



web: pr-dc.com, email: info@pr-dc.com, github: github.com/PR-DC



JSLAB v1.0.3 SOURCE CODE

github.com/PR-DC/JSLAB

Contents

1	root	2
2	config	6
3	cpp	34
4	css	66
5	html	178
6	js	217
6.1	code	242
6.2	dev	255
6.3	editor	275
6.4	main	296
6.5	sandbox	347
6.6	windows	852

1 root

```

1  {
2      "name": "JSLAB",
3      "version": "1.0.3",
4      "description": "JavaScript LABoratory environment",
5      "main": "js/init.js",
6      "repository": "https://github.com/PR-DC/JSLAB",
7      "license": "GPL-3.0-or-later",
8      "author": "Milos Petrasinovic <mpetrasinovic@prdc.rs>",
9      "homepage": "https://pr-dc.com",
10     "scripts": {
11         "preinstall": "npm install rimraf & npm install node-7z & npm install 7zip
12             -bin & node js/dev/prepare-libs.js & npm install -g node-gyp",
13         "postinstall": "node js/dev/build-configure.js & node-gyp rebuild & node
14             js/dev/make-doc.js & electron .",
15         "start": "electron .",
16         "debug": "electron . --debug-app",
17         "test": "electron . --test-app",
18         "build": "node js/dev/build-configure.js & node-gyp rebuild & electron .",
19         "pack": "node js/dev/build-configure.js --action pack & node-gyp rebuild &
20             electron-builder --dir",
21         "dist": "node js/dev/build-configure.js --action dist & node-gyp rebuild &
22             electron-builder --win",
23         "dist-portable": "node js/dev/build-configure.js --action dist-portable &
24             node-gyp rebuild & electron-builder --win portable",
25         "dist-signed": "node js/dev/build-configure.js --action dist --sign-build
26             & node-gyp rebuild & electron-builder --win",
27         "clear-app-data": "node js/dev/clear-app-data.js --confirm",
28         "make-doc": "node js/dev/make-doc.js",
29         "make-source-code-book": "node js/dev/make-source-code-book.js",
30         "update-libs": "node js/dev/download-libs.js --force --confirm",
31         "upload-source": "node js/dev/upload-source-code.js"
32     },
33     "build": {
34         "asar": false,
35         "appId": "com.pr-dc.jslab",
36         "productName": "PR-DC JSLAB",
37         "copyright": "Copyright 2024 @ PR-DC",
38         "nodeGypRebuild": true,
39         "directories": {
40             "buildResources": "build",
41             "output": "dist"
42         },
43         "files": [
44             "**",
45             "!cpp",
46             "!dev",
47             "!dist",
48             "!build",
49             "!bin",
50             "!binding.gyp",
51             "!js/dev",
52             "!lib/boost-1.86.0",
53             "!lib/cgal-6.0.1",
54             "!node-gyp",
55             "!node-gyp-build"
56         ]
57     }
58 }
```



```
48      " ! lib/eigen -3.4.0 "
49  ],
50  "fileAssociations": [
51    {
52      "ext": "jsl",
53      "name": "JSL",
54      "description": "JavaScript LABoratory script",
55      "role": "Editor",
56      "icon": "icons/icon.ico"
57    }
58  ],
59  "extraResources": [
60    {
61      "from": "./build/Release/",
62      "to": "./app/build/Release/",
63      "filter": [
64        "*.*node",
65        "*.*dll"
66      ]
67    },
68    {
69      "from": "./node_modules/npm/",
70      "to": "./app/node_modules/npm/"
71    },
72    {
73      "from": "./node_modules/node-gyp/",
74      "to": "./app/node_modules/node-gyp/"
75    }
76  ],
77  "compression": "maximum",
78  "mac": {
79    "category": "public.app-category.utilities",
80    "icon": "icons/icon.icns",
81    "target": "dmg"
82  },
83  "win": {
84    "icon": "icons/icon.ico",
85    "target": "nsis",
86    "signtoolOptions": {
87      "publisherName": [
88        "PR-DC"
89      ],
90      "signingHashAlgorithms": [
91        "sha256"
92      ]
93    }
94  },
95  "linux": {
96    "icon": "icons/",
97    "target": "AppImage"
98  },
99  "portable": {
100    "splashImage": "icons/splash.bmp"
101  },
102  "nsis": {
```

```
103     "oneClick": false,
104     "perMachine": true,
105     "allowToChangeInstallationDirectory": true,
106     "installerIcon": "icons/nsis_in_icon.ico",
107     "uninstallerIcon": "icons/nsis_un_icon.ico",
108     "installerHeader": "img/nsis_in_header.bmp",
109     "installerHeaderIcon": "icons/icon.ico",
110     "installerSidebar": "img/nsis_in_welcom.bmp",
111     "uninstallerSidebar": "img/nsis_un_welcom.bmp",
112     "uninstallDisplayName": "JSLAB ${version}",
113     "shortcutName": "JSLAB",
114     "language": "1033",
115     "displayLanguageSelector": true,
116     "installerLanguages": [
117       "en_US",
118       "sr_RS"
119     ],
120     "multiLanguageInstaller": true,
121     "warningsAsErrors": false
122   },
123   "artifactName": "JSLAB_1.0.3.${ext}"
124 },
125   "devDependencies": {
126     "electron": "36.4.0",
127     "electron-builder": "26.0.12"
128   },
129   "dependencies": {
130     "@babel/parser": "7.27.5",
131     "@babel/plugin-syntax-top-level-await": "7.14.5",
132     "7zip-bin": "^5.2.0",
133     "big-json-viewer": "0.1.7",
134     "bytenode": "1.5.7",
135     "dir-compare": "5.0.0",
136     "electron-context-menu": "3.6.1",
137     "electron-store": "8.2.0",
138     "eslint": "9.29.0",
139     "fast-xml-parser": "5.2.5",
140     "fmin": "0.0.4",
141     "glob": "11.0.3",
142     "jsdoc-api": "9.3.4",
143     "ml-regression-polynomial": "3.0.2",
144     "node-7z": "^3.0.0",
145     "node-addon-api": "8.4.0",
146     "node-gyp": "11.2.0",
147     "node-mavlink": "2.1.0",
148     "npm": "11.4.2",
149     "path-equal": "1.2.5",
150     "pdfkit": "0.17.1",
151     "recast": "0.23.11",
152     "rimraf": "^5.0.10",
153     "seedrandom": "3.0.5",
154     "serialport": "13.0.0",
155     "source-map": "0.7.4",
156     "svg-to-pdfkit": "0.1.8",
157     "tcp-port-used": "1.0.2",
```

```

158     "usb": "2.15.0",
159     "zeromq": "6.0.0-beta.20"
160   }
161 }
```

Listing 1 - package.json

```

1  {
2   "targets": [
3     {
4       "target_name": "native_module",
5       "sources": [
6         "cpp/native-module.cpp"
7       ],
8       "include_dirs": [
9         "<!@(node -p \'require('node-addon-api').include\")",
10        "<(module_root_dir)/lib/eigen-3.4.0/"
11      ],
12      "cflags!": [
13        "-fno-exceptions"
14      ],
15      "cflags_cc!": [
16        "-fno-exceptions"
17      ],
18      "defines": [
19        "NAPI_DISABLE_CPP_EXCEPTIONS"
20      ],
21      "msvs_settings": {
22        "VCCLCompilerTool": {
23          "AdditionalOptions": [
24            "-std:c++17"
25          ]
26        }
27      }
28    },
29    {
30      "target_name": "alpha_shape_3d",
31      "sources": [
32        "cpp/alpha-shape-3d.cpp"
33      ],
34      "include_dirs": [
35        "<!@(node -p \'require('node-addon-api').include\")",
36        "<(module_root_dir)/lib/cgal-6.0.1/include/",
37        "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/include",
38        "<(module_root_dir)/lib/boost-1.86.0/"
39      ],
40      "libraries": [
41        "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/lib/gmp.lib",
42        "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/lib/mpfr.lib"
43      ],
44      "cflags!": [
45        "-fno-exceptions"
46      ],
47      "cflags_cc!": [
48        "-fno-exceptions",
49        "-O3",
50      ],
51      "defines": [
52        "NAPI_DISABLE_CPP_EXCEPTIONS"
53      ],
54      "msvs_settings": {
55        "VCCLCompilerTool": {
56          "AdditionalOptions": [
57            "-std:c++17"
58          ]
59        }
60      }
61    }
62  }
63 }
```

```

50         "-DNDEBUG"
51     ],
52     "defines": [
53       "NAPI_DISABLE_CPP_EXCEPTIONS"
54     ],
55     "copies": [
56       {
57         "destination": "<(module_root_dir)/build/Release",
58         "files": [
59           "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/gmp-10.dll",
60           "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/mpfr-6.dll"
61         ]
62       }
63     ],
64     "msvs_settings": {
65       "VCCLCompilerTool": {
66         "AdditionalOptions": [
67           "-std:c++17",
68           "/GR",
69           "/EHsc"
70         ]
71       }
72     }
73   }
74 }
75 }
```

Listing 2 - binding.gyp

2 config

```

1 /**
2  * @file JSLAB global configuration
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for application configuration.
10 */
11 class PRDC_APP_CONFIG {
12
13 /**
14  * Create JSLAB configuration object.
15 */
16 constructor() {
17   this.PRODUCTION = false;
18   this.DEBUG = false;
19   this.TEST = false;
20   this.SIGN_BUILD = false;
21
22   this.GROUP_RAF = true;
23   this.OUTPUT_COMPLETE_JSDOC = false;
```

```

24
25   this.DEBUG_FUN_SHADOW = false;
26   this.DEBUG_NEW_FUN = false;
27   this.DEBUG_RENDER_GONE_ERROR = true;
28   this.DEBUG_SYM_PYTHON_EVAL_CODE = false;
29   this.DEBUG_PRE_TRANSFORMED_CODE = false;
30   this.DEBUG_TRANSFORMED_CODE = false;
31   this.DEBUG_PARALLEL_WORKER_SETUP_FUN = false;
32   this.DEBUG_PARALLEL_WORKER_WORK_FUN = false;
33
34   this.LOG_RENDER_GONE_ERROR = true;
35
36 // Log codes
37 this.LOG_CODES = {
38   'other': 0,
39   'render-gone-error': 1,
40 };
41
42 // JSLAB settings
43 this.FORBIDDEN_NAMES = [ 'jsl', 'config', 'language', 'app_path', 'packed' ];
44 this.MATHJS_PREVENT_OVERRIDE = [ 'config', 'print', 'Infinity', 'NaN', 'isNaN', 'Node' ];
45 this.SUBMODULES = {
46   'builtin': [
47     {name: 'basic', file: 'basic', class_name: 'PRDC_JSLAB_LIB_BASIC'},
48     {name: 'math', file: 'math', class_name: 'PRDC_JSLAB_LIB_MATH'},
49     {name: 'non_blocking', file: 'non-blocking', class_name: 'PRDC_JSLAB_LIB_NON_BLOCKING'},
50     {name: 'path', file: 'path', class_name: 'PRDC_JSLAB_LIB_PATH'},
51     {name: 'windows', file: 'windows', class_name: 'PRDC_JSLAB_LIB_WINDOWS'},
52     {name: 'figures', file: 'figures', class_name: 'PRDC_JSLAB_LIB FIGURES'},
53     {name: 'time', file: 'time', class_name: 'PRDC_JSLAB_LIB_TIME'},
54     {name: 'array', file: 'array', class_name: 'PRDC_JSLAB_LIB_ARRAY'},
55     {name: 'color', file: 'color', class_name: 'PRDC_JSLAB_LIB_COLOR'},
56     {name: 'conversion', file: 'conversion', class_name: 'PRDC_JSLAB_LIB_CONVERSION'},
57     {name: 'device', file: 'device', class_name: 'PRDC_JSLAB_LIB_DEVICE'},
58     {name: 'serial_device', file: 'serial-device', class_name: 'PRDC_JSLAB_LIB_SERIAL_DEVICE'},
59     {name: 'file_system', file: 'file-system', class_name: 'PRDC_JSLAB_LIB_FILE_SYSTEM'},
60     {name: 'system', file: 'system', class_name: 'PRDC_JSLAB_LIB_SYSTEM'},
61     {name: 'geography', file: 'geography', class_name: 'PRDC_JSLAB_LIB_GEOGRAPHY'},
62     {name: 'networking', file: 'networking', class_name: 'PRDC_JSLAB_LIB_NETWORKING'},
63     {name: 'format', file: 'format', class_name: 'PRDC_JSLAB_LIB_FORMAT'},
64     {name: 'render', file: 'render', class_name: 'PRDC_JSLAB_LIB_RENDER'},
65     {name: 'geometry', file: 'geometry', class_name: 'PRDC_JSLAB_LIB_GEOMETRY'},
66     {name: 'control', file: 'control', class_name: 'PRDC_JSLAB_LIB_CONTROL'}
  ],
}

```

```

67     {name: 'optim', file: 'optim', class_name: 'PRDC_JSLAB_LIB_OPTIM'},
68     {name: 'presentation', file: 'presentation', class_name: 'PRDC_JSLAB_LIB_PRESENTATION'},
69     {name: 'mechanics', file: 'mechanics', class_name: 'PRDC_JSLAB_LIB_MECHANICS'},
70     {name: 'gui', file: 'gui', class_name: 'PRDC_JSLAB_LIB_GUI'},
71   ],
72   'lib': [
73     {name: 'parallel', file: 'parallel', class_name: 'PRDC_JSLAB_PARALLEL'},
74     {name: 'mat', file: 'matrix-math', class_name: 'PRDC_JSLAB_MATRIX_MATH'},
75     {name: 'vec', file: 'vector-math', class_name: 'PRDC_JSLAB_VECTOR_MATH'},
76     {name: 'sym', file: 'sym-math', class_name: 'PRDC_JSLAB_SYMBOLIC_MATH'},
77   ],
78 };
79
80 this.DOC_SUBMODULES_ADDITIONAL = [
81   {name: 'Matrix', file: 'matrix-math', class_name: 'PRDC_JSLAB_MATRIX'},
82   {name: 'Vector', file: 'vector-math', class_name: 'PRDC_JSLAB_VECTOR'},
83   {name: 'Symbolic', file: 'sym-math', class_name: 'PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL'},
84   {name: 'Window', file: 'windows', class_name: 'PRDC_JSLAB_WINDOW'},
85   {name: 'Figure', file: 'figures', class_name: 'PRDC_JSLAB_PICTURE'},
86   {name: 'Plot', file: 'figures', class_name: 'PRDC_JSLAB_PLOT'},
87   {name: 'freecad_link', file: 'freecad-link', class_name: 'PRDC_JSLAB_FREECAD_LINK'},
88   {name: 'om_link', file: 'om-link', class_name: 'PRDC_JSLAB_OPENMODELICA_LINK'},
89   {name: 'tcp_client', file: 'networking-tcp', class_name: 'PRDC_JSLAB_TCP_CLIENT'},
90   {name: 'tcp_server', file: 'networking-tcp', class_name: 'PRDC_JSLAB_TCP_SERVER'},
91   {name: 'udp_client', file: 'networking-udp', class_name: 'PRDC_JSLAB_UDP'},
92   {name: 'udp_server', file: 'networking-udp', class_name: 'PRDC_JSLAB_UDP_SERVER'},
93   {name: 'video_call', file: 'networking-video-call', class_name: 'PRDC_JSLAB_VIDEOCALL'},
94   {name: 'mathjs', file: 'mathjs-doc', class_name: 'PRDC_JSLAB_MATHJS_DOC'},
95   {name: 'rcmiga', file: 'optim-rcmiga', class_name: 'PRDC_JSLAB_OPTIM_RCMIGA'},
96   {name: 'space_search', file: 'geometry-spacesearch', class_name: 'PRDC_JSLAB_GEOMETRY_SPACE_SEARCH'},
97   {name: 'map', file: 'geography-map', class_name: 'PRDC_JSLAB_GEOGRAPHY_MAP'},
98   {name: 'map_3d', file: 'geography-map-3d', class_name: 'PRDC_JSLAB_GEOGRAPHY_MAP_3D'},
99   {name: 'Gamepad', file: 'device-gamepad', class_name: 'PRDC_JSLAB_DEVICE_GAMEPAD'},
100 ];
101

```

```

102     this.SOURCE_CODE_BOOK_FILES = [
103         'package.json',
104         'binding.gyp',
105         'config',
106         'cpp',
107         'css',
108         'html',
109         'js'
110     ];
111
112     this.SOURCE_CODE_BOOK_FILES_EXCLUDE = [
113         'html/io_html_figure.html',
114     ];
115
116     this.LINT_OPTIONS = {
117         overrideConfigFile: true,
118         overrideConfig: {
119             languageOptions: {
120                 ecmaVersion: "latest",
121                 sourceType: "module"
122             },
123             rules: {
124                 "no-unused-vars": "warn",
125                 "semi": ["warn", "always"],
126                 "no-extra-semi": "warn",
127                 "no-unreachable": "warn",
128                 "consistent-return": "warn",
129                 "no-shadow": "warn",
130                 "no-use-before-define": "warn"
131             }
132         }
133     };
134
135     this.COMPRESSSED_LIBS = [
136         'leaflet-1.9.4',
137         'sympy-0.26.2',
138         'cgal-6.0.1',
139         'boost-1.86.0',
140         'codemirror-5.49.2',
141         'eigen-3.4.0',
142         'three.js-r162',
143         'Cesium-1.124'
144     ];
145     this.COMPILE_LIBS = [];
146
147     // Language
148     this.langs = ["en", "rs", "rsc"];
149
150     // Windows
151     this.WIN_SAVE_DEBOUNCE_TIME = 50; // [ms]
152
153     // Other
154     this.PLOTER = ['plotly', 'echarts'][0];
155     this.DOC_LATEX_RERUNS_NUMBER = 3;
156     this.SOURCE_CODE_BOOK_LATEX_RERUNS_NUMBER = 3;

```

```

157   this.MAX_ACTIVE_WEBGL_CONTEXTS = '128';
158   this.MAX_JSON_STRING_LENGTH = 1000;
159
160   // Build sign
161   this.COMPANY_NAME = process.env.COMPANY_NAME;
162   this.TIMESTAMP_SERVER = process.env.TIMESTAMP_SERVER;
163   this.SIGN_TOOL_PATH = process.env.SIGN_TOOL_PATH;
164
165   // Upload and download libs from server
166   this.SERVER_SOURCE_PATH = process.env.SERVER_PATH + "JSLAB/";
167   this.SOURCE_UPLOAD_EXCLUDE = [ '/bin', '/build', '/dist', '/node_modules',
168     '/package-lock.json', '/binding.gyp', '/lib', '*.obj' ];
169   this.SERVER_LIBS_PATH = process.env.SERVER_LIBS_PATH;
170
171   this.USED_LIBS = [
172     'sprintf-1.1.3',
173     'sympy-0.26.2',
174     'cgal-6.0.1',
175     'boost-1.86.0',
176     'codemirror-5.49.2',
177     'complete.ly.1.0.1',
178     'd3-7.8.5',
179     'draggabilly-2.3.0',
180     'eigen-3.4.0',
181     'highlight-11.0.1',
182     'jquery-3.7.0',
183     'jshint-2.13.0',
184     'math-11.8.2',
185     'tex-mml-ctml-3.2.0',
186     'luxon-3.4.4',
187     'plotly-2.24.2',
188     'three.js-r162',
189     'inflate-0.3.1',
190     'hammer-2.0.8',
191     'anime-3.2.1',
192     'tween.js-23.1.1',
193     'leaflet-1.9.4',
194     'leaflet.rotatedMarker-0.2.0',
195     'Cesium-1.124',
196     'mermaid-11.4.1',
197     'jstree-3.3.17',
198     'PRDC_APP_LOGGER',
199     'PRDC_PANEL',
200     'PRDC_TABS',
201     'PRDC_POPUP',
202     'PRDC_SVG_VIEWER'
203   ];
204
205   this.UPLOAD_COMPARE_SIZE = false;
206   this.UPLOAD_COMPARE_CONTENT = true;
207   this.UPLOAD_COMPARE_DATE = false;
208   this.UPLOAD_COMPARE_SIZE_ON_DISTINCT = false;
209
210   this.PANEL_RESIZER_WIDTH = 10;
211   this.PANEL_MIN_SIZE = 10;

```

```

211     this.PANEL_DEFAULT_COLUMNS = [20, 80];
212     this.PANEL_DEFAULT_LEFT_ROWS = [100/3, 100/3, 100/3];
213     this.PANEL_DEFAULT_WORKSPACE_COLUMNS = [50, 25, 25];
214   }
215 }
216
217 exports.PRDC_APP_CONFIG = PRDC_APP_CONFIG;

```

Listing 3 - config.js

```

1  {
2    "1": {
3      "en": "Editor",
4      "rs": "Uredavač",
5      "rsc": "Уређивач"
6    },
7    "2": {
8      "en": "Help",
9      "rs": "Pomoć",
10     "rsc": "Помоћ"
11   },
12   "3": {
13     "en": "Info",
14     "rs": "Info",
15     "rsc": "Инфо"
16   },
17   "4": {
18     "en": "File Browser",
19     "rs": "Pretraživač fajlova",
20     "rsc": "Претраживач фајлова"
21   },
22   "5": {
23     "en": "Workspace",
24     "rs": "Radni prostor",
25     "rsc": "Радни простор"
26   },
27   "6": {
28     "en": "Command History",
29     "rs": "Istorija Komandi",
30     "rsc": "Историја Команди"
31   },
32   "7": {
33     "en": "Command Window",
34     "rs": "Komandni Prozor",
35     "rsc": "Командни Прозор"
36   },
37   "8": {
38     "en": "version",
39     "rs": "verzija",
40     "rsc": "верзија"
41   },
42   "9": {
43     "en": "Settings",
44     "rs": "Podešavanja",
45     "rsc": "Подешавања"
46   },

```

```
47     "10": {
48         "en": "New",
49         "rs": "Nova",
50         "rsc": "Нова"
51     },
52     "11": {
53         "en": "Open",
54         "rs": "Otvoři",
55         "rsc": "Отвори"
56     },
57     "12": {
58         "en": "Save",
59         "rs": "Sačuvaj",
60         "rsc": "Сачувай"
61     },
62     "13": {
63         "en": "Save As",
64         "rs": "Sačuvaj Kao",
65         "rsc": "Сачувай КАО"
66     },
67     "14": {
68         "en": "Run",
69         "rs": "Pokreni",
70         "rsc": "Покрени"
71     },
72     "15": {
73         "en": "Sandbox paused",
74         "rs": "Radni prostor pauziran",
75         "rsc": "Радни простор паузиран"
76     },
77     "16": {
78         "en": "Language",
79         "rs": "Jezik",
80         "rsc": "Језик"
81     },
82     "17": {
83         "en": "Number of latest messages",
84         "rs": "Broj najnovijih poruka",
85         "rsc": "Број најновијих порука"
86     },
87     "18": {
88         "en": "Close log save dialog",
89         "rs": "Zatvorite dijalog za čuvanje dnevnika",
90         "rsc": "Затворите дијалог за чување дневника"
91     },
92     "19": {
93         "en": "Close command history",
94         "rs": "Zatvorite istoriju komandi",
95         "rsc": "Затвори историју команди"
96     },
97     "20": {
98         "en": "Close script directory dialog",
99         "rs": "Zatvorite dijalog direktorijuma skripte",
100        "rsc": "Затворите дијалог директоријума скрипте"
101    },
102}
```

```
102  "21": {
103    "en": "Close paths menu",
104    "rs": "Zatvorite meni putanja",
105    "rsc": "Затвори мени путања"
106  },
107  "22": {
108    "en": "Close help",
109    "rs": "Zatvori pomoć",
110    "rsc": "Затвори помоћ"
111  },
112  "23": {
113    "en": "Close info",
114    "rs": "Zatvori informacije",
115    "rsc": "Затворите информације"
116  },
117  "24": {
118    "en": "Close settings",
119    "rs": "Zatvorite podešavanja",
120    "rsc": "Затворите подешавања"
121  },
122  "25": {
123    "en": "Select language",
124    "rs": "Izaberite jezik",
125    "rsc": "Изаберите језик"
126  },
127  "26": {
128    "en": "Open new script",
129    "rs": "Otvorite novu skriptu",
130    "rsc": "Отворите нову скрипту"
131  },
132  "27": {
133    "en": "Open file",
134    "rs": "Otvorite datoteku",
135    "rsc": "Отворите датотеку"
136  },
137  "28": {
138    "en": "Save file",
139    "rs": "Sačuvaj datoteku",
140    "rsc": "Сачувавј датотеку"
141  },
142  "29": {
143    "en": "File save as",
144    "rs": "Datoteku sačuvaj kao",
145    "rsc": "Датотеку сачувавј као"
146  },
147  "30": {
148    "en": "Save script and run",
149    "rs": "Sačuvajte skriptu i pokreni",
150    "rsc": "Сачувавте скрипту и покрени"
151  },
152  "31": {
153    "en": "Click to go back",
154    "rs": "Kliknite da biste se vratili",
155    "rsc": "Кликните да бисте се вратили"
156  },
```

```
157 "32": {
158     "en": "Click to go forward",
159     "rs": "Kliknite da idete napred",
160     "rsc": "Кликните да идете напред"
161 },
162 "33": {
163     "en": "Click to go up",
164     "rs": "Kliknite da idete gore",
165     "rsc": "Кликните да идете горе"
166 },
167 "34": {
168     "en": "Select folder",
169     "rs": "Izaberi direktorijum",
170     "rsc": "Изабери директоријум"
171 },
172 "35": {
173     "en": "Save this path",
174     "rs": "Sačuvajte ovu putanju",
175     "rsc": "Сачувайте ову путању"
176 },
177 "36": {
178     "en": "More folder options",
179     "rs": "Više opcija direktorijuma",
180     "rsc": "Више опција директоријума"
181 },
182 "37": {
183     "en": "Refresh file browser",
184     "rs": "Osvežite pregledač datoteka",
185     "rsc": "Освежите прегледач датотека"
186 },
187 "38": {
188     "en": "Clear workspace",
189     "rs": "Očistite radni prostor",
190     "rsc": "Очистите радни простор"
191 },
192 "39": {
193     "en": "Clear command history",
194     "rs": "Obrišite istoriju komandi",
195     "rsc": "Обришите историју команди"
196 },
197 "40": {
198     "en": "Open settings",
199     "rs": "Otvorite podešavanja",
200     "rsc": "Отворите подешавања"
201 },
202 "41": {
203     "en": "Hide timestamp",
204     "rs": "Sakrij vremensku oznaku",
205     "rsc": "Сакриј временску ознаку"
206 },
207 "42": {
208     "en": "Turn off auto scroll",
209     "rs": "Isključite automatsko pomeranje",
210     "rsc": "Искључите аутоматско померање"
211 },
```

```
212 "43": {
213     "en": "Clear command window",
214     "rs": "Očistite komandni prozor",
215     "rsc": "Очистите командни прозор"
216 },
217 "44": {
218     "en": "Save log to file",
219     "rs": "Sačuvajte dnevnik u datoteku",
220     "rsc": "Сачувавте дневник у датотеку"
221 },
222 "45": {
223     "en": "Scroll to bottom",
224     "rs": "Pomerite se do dna",
225     "rsc": "Померите се до дна"
226 },
227 "46": {
228     "en": "Close settings dialog",
229     "rs": "Zatvorite dijalog podešavanja",
230     "rsc": "Затворите дијалог подешавања"
231 },
232 "47": {
233     "en": "File modified",
234     "rs": "Fajl je izmenjen",
235     "rsc": "Фајл је изменјен"
236 },
237 "48": {
238     "en": "The file",
239     "rs": "Fajl",
240     "rsc": "Фајл"
241 },
242 "49": {
243     "en": "is about to be closed but has been modified. Do you want to save or
244         discard the changes?",
245     "rs": "će biti zatvoren ali je izmenjen. Da li želite da sačuvate ili
246         odbacite izmene?",
247     "rsc": "ће бити затворен али је изменјен. Да ли желите да сачувате или одба-
248         ците измене?"
249 },
250 "50": {
251     "en": "Save",
252     "rs": "Sačuvaj",
253     "rsc": "Сачувай"
254 },
255 "51": {
256     "en": "Discard",
257     "rs": "Odbaci",
258     "rsc": "Одбаци"
259 },
260 "52": {
261     "en": "Cancel",
262     "rs": "Otkaži",
263     "rsc": "Откажи"
264 },
265 "53": {
266     "en": "Variable",
267 }
```

```
264      "rs": "Promenljiva",
265      "rsc": "Променљива"
266    },
267    "54": {
268      "en": "Type",
269      "rs": "Tip",
270      "rsc": "Тип"
271    },
272    "55": {
273      "en": "Class",
274      "rs": "Klasa",
275      "rsc": "Класа"
276    },
277    "56": {
278      "en": "Command window settings",
279      "rs": "Podešavanja komandnog prozora",
280      "rsc": "Подешавања командног прозора"
281    },
282    "57": {
283      "en": "Change settings",
284      "rs": "Promenite podešavanja",
285      "rsc": "Промените подешавања"
286    },
287    "58": {
288      "en": "Save log",
289      "rs": "Sačuvaj dnevnik",
290      "rsc": "Сачувај дневник"
291    },
292    "59": {
293      "en": "Write timestamps",
294      "rs": "Upisuj vremenske oznake",
295      "rsc": "Уписуј временске ознаке"
296    },
297    "60": {
298      "en": "Save log",
299      "rs": "Sačuvaj dnevnik",
300      "rsc": "Сачувај дневник"
301    },
302    "61": {
303      "en": "Session command history",
304      "rs": "Istorijska komandi",
305      "rsc": "Историја команди"
306    },
307    "62": {
308      "en": "Script directory",
309      "rs": "Direktorijum skripte",
310      "rsc": "Директоријум скрипте"
311    },
312    "63": {
313      "en": "The directory",
314      "rs": "Direktorijum",
315      "rsc": "Директоријум"
316    },
317    "64": {
318      "en": "is not a working directory nor a saved directory, would you like to
```

```
change a working directory or save this directory?",  
319 "rs": "nije radni direktorijum niti sačuvani direktorijum, da li želite da  
320     promenite radni direktorijum ili sačuvate ovaj direktorijum?",  
321     "rsc": "није радни директоријум нити сачувани директоријум, да ли желите да  
322         промените радни директоријум или сачувате овај директоријум?"  
323     },  
324     "65": {  
325         "en": "Change working directory",  
326         "rs": "Promeni radni direktorijum",  
327         "rsc": "Промени радни директоријум"  
328     },  
329     "66": {  
330         "en": "Save directory",  
331         "rs": "Sačuvaj direktorijum",  
332         "rsc": "Сачувај директоријум"  
333     },  
334     "67": {  
335         "en": "Run",  
336         "rs": "Pokreni",  
337         "rsc": "Покрени"  
338     },  
339     "68": {  
340         "en": "Saved paths",  
341         "rs": "Sačuvane putanje",  
342         "rsc": "Сачуване путање"  
343     },  
344     "69": {  
345         "en": "Command window keyboard shortcuts",  
346         "rs": "Prečice tastature komandnog prozora",  
347         "rsc": "Пречице тастатуре командног прозора"  
348     },  
349     "70": {  
350         "en": "Shortcut",  
351         "rs": "Prečica",  
352         "rsc": "Пречица"  
353     },  
354     "71": {  
355         "en": "Action",  
356         "rs": "Akcija",  
357         "rsc": "Акција"  
358     },  
359     "72": {  
360         "en": "Clear input / Close dialog",  
361         "rs": "Obriši unos / Zatvori dijalog",  
362         "rsc": "Обриши унос / Затвори дијалог"  
363     },  
364     "73": {  
365         "en": "Go backward through the command history",  
366         "rs": "Idi unazad kroz istoriju komandi",  
367         "rsc": "Иди уназад кроз историју команди"  
368     },  
369     "74": {  
370         "en": "Go forward through the command history",  
371         "rs": "Idi unapred kroz istoriju komandi",  
372         "rsc": "Иди унапред кроз историју команди"
```

```
371 },
372 "75": {
373     "en": "First command in the command history",
374     "rs": "Prva komanda u istoriji komandi",
375     "rsc": "Прва команда у историји команди"
376 },
377 "76": {
378     "en": "Last command in the command history",
379     "rs": "Poslednja komanda u istoriji komandi",
380     "rsc": "Последња команда у историји команди"
381 },
382 "77": {
383     "en": "Break the line",
384     "rs": "Prekini liniju",
385     "rsc": "Прекини линију"
386 },
387 "78": {
388     "en": "Repeat the last command",
389     "rs": "Ponovi poslednju komandu",
390     "rsc": "Понови последњу команду"
391 },
392 "79": {
393     "en": "Show the command history",
394     "rs": "Prikaži istoriju komandi",
395     "rsc": "Прикажи историју команди"
396 },
397 "80": {
398     "en": "Clear the command history",
399     "rs": "Obriši istoriju komandi",
400     "rsc": "Обриши историју команди"
401 },
402 "81": {
403     "en": "Complete current command from the command history",
404     "rs": "Završi trenutnu komandu iz istorije komandi",
405     "rsc": "Заврши тренутну команду из историје команди"
406 },
407 "82": {
408     "en": "Focus command input",
409     "rs": "Fokusiraj na unos komandi",
410     "rsc": "Фокусирај на унос команди"
411 },
412 "83": {
413     "en": "Open help",
414     "rs": "Otvorite pomoć",
415     "rsc": "Отворите помоћ"
416 },
417 "84": {
418     "en": "Open settings dialog",
419     "rs": "Otvorite dijalog podešavanja",
420     "rsc": "Отворите дијалог подешавања"
421 },
422 "85": {
423     "en": "Open log save dialog",
424     "rs": "Otvorite dijalog za čuvanje dnevnika",
425     "rsc": "Отворите дијалог за чување дневника"
```

```
426 },
427 "86": {
428     "en": "This software uses open-source components stated below, copyright
        and other proprietary rights of these components belong to their
        respective owners.",
429     "rs": "Ovaj softver koristi open-source komponente navedene ispod,
        autorska prava i druga prava intelektualne svojine ovih komponenti
        pripadaju njihovim vlasnicima.",
430     "rsc": "Овај софтвер користи open-source компоненте наведене испод, ауторс-
        ка права и друга права интелектуалне својине ових компоненти припадају
        њиховим власницима."
431 },
432 "87": {
433     "en": "Ready...",
434     "rs": "Spremno...",
435     "rsc": "Спремно..."
436 },
437 "88": {
438     "en": "Evaluating...",
439     "rs": "Evaluacija...",
440     "rsc": "Евалуација..."
441 },
442 "89": {
443     "en": "Stop request sent by user...",
444     "rs": "Korisnik je poslao zahtev za zaustavljanje...",
445     "rsc": "Корисник је послao захтев за заустављање..."
446 },
447 "90": {
448     "en": "Stop loop triggered...",
449     "rs": "Petlja izvršavanja je zaustavljena...",
450     "rsc": "Петља извршавања је заустављена..."
451 },
452 "91": {
453     "en": "Sandbox activity",
454     "rs": "Aktivnost radnog prostora",
455     "rsc": "Активност радног простора"
456 },
457 "92": {
458     "en": "Unable to scan directory",
459     "rs": "Nije moguće skenirati direktorijum",
460     "rsc": "Није могуће скенирати директоријум"
461 },
462 "93": {
463     "en": "Running total of",
464     "rs": "Ukupno pokrenuto",
465     "rsc": "Укупно покренуто"
466 },
467 "94": {
468     "en": "tests",
469     "rs": "testova",
470     "rsc": "тестова"
471 },
472 "95": {
473     "en": "Test",
474     "rs": "Test",
```

```
475     "rsc": "Test"
476 },
477 "96": {
478     "en": "failed with error",
479     "rs": "nije uspeo zbog greske",
480     "rsc": "није успео због грешке"
481 },
482 "97": {
483     "en": "failed",
484     "rs": "nije uspeo",
485     "rsc": "није успео"
486 },
487 "98": {
488     "en": "passed",
489     "rs": "uspesan",
490     "rsc": "успешан"
491 },
492 "99": {
493     "en": "Final results",
494     "rs": "Konačni rezultati",
495     "rsc": "Коначни резултати"
496 },
497 "100": {
498     "en": "Tests passed",
499     "rs": "Testovi uspešni",
500     "rsc": "Тестови успешни"
501 },
502 "101": {
503     "en": "Tests failed",
504     "rs": "Testovi nisu uspešni",
505     "rsc": "Тестови нису успешни"
506 },
507 "102": {
508     "en": "There is no implemented tests.",
509     "rs": "Nema implementiranih testova.",
510     "rsc": "Нема имплементираних тестова."
511 },
512 "103": {
513     "en": "Script not found at",
514     "rs": "Skripta nije pronađena na",
515     "rsc": "Скрипта није пронађена на"
516 },
517 "104": {
518     "en": "Not enough lines in file",
519     "rs": "Nema dovoljno linija u fajlu",
520     "rsc": "Нема довољно линија у фајлу"
521 },
522 "105": {
523     "en": "script",
524     "rs": "skripta",
525     "rsc": "скрипта"
526 },
527 "106": {
528     "en": "File",
529     "rs": "Fajl",
```

```
530      "rsc": "Фајл"
531  },
532  "107": {
533    "en": "is selected. There are also",
534    "rs": "је одабран. Такође постоје",
535    "rsc": "је одабран. Такође постоје"
536  },
537  "108": {
538    "en": "is selected. There is also",
539    "rs": "је одабран. Такође постоји",
540    "rsc": "је одабран. Такође постоји"
541  },
542  "109": {
543    "en": "Failed to find",
544    "rs": "Није пронађено",
545    "rsc": "Није пронађено"
546  },
547  "110": {
548    "en": "module",
549    "rs": "modul",
550    "rsc": "модул"
551  },
552  "111": {
553    "en": "Provided path is not string",
554    "rs": "Дата путања није низ карактера",
555    "rsc": "Дата путања није низ карактера"
556  },
557  "112": {
558    "en": "line",
559    "rs": "linija",
560    "rsc": "линија"
561  },
562  "113": {
563    "en": "column",
564    "rs": "kolona",
565    "rsc": "колона"
566  },
567  "114": {
568    "en": "at",
569    "rs": "на",
570    "rsc": "на"
571  },
572  "115": {
573    "en": "Not implemented!",
574    "rs": "Није implementirano!",
575    "rsc": "Није имплементирано!"
576  },
577  "116": {
578    "en": "Code arrived to user defined end point at line",
579    "rs": "Код је дошао до корисниčki definisane tačke zaustavaljanja na
580        линији",
581    "rsc": "Код је дошао до кориснички дефинисане тачке заустављања на линији"
582  },
583  "117": {
584    "en": "Unable to write to file",
```

```
584     "rs": "Nije moguće pisati u fajl",
585     "rsc": "Није могуће писати у фајл"
586   },
587   "118": {
588     "en": "File content is not valid JSON",
589     "rs": "Sadržaj fajla nije ispravan JSON",
590     "rsc": "Садржај фајла није исправан JSON"
591   },
592   "119": {
593     "en": "Folder selection canceled",
594     "rs": "Odabir direktorijuma je otkazan",
595     "rsc": "Одабир директоријума је отказан"
596   },
597   "120": {
598     "en": "Content of binding.gyp is not valid JSON",
599     "rs": "Sadržaj binding.gyp nije ispravan JSON",
600     "rsc": "Садржај binding.gyp није исправан JSON"
601   },
602   "121": {
603     "en": "Unknown response",
604     "rs": "Nepoznat odgovor",
605     "rsc": "Непознат одговор"
606   },
607   "122": {
608     "en": "No targets defined",
609     "rs": "Nisu definisani ciljevi",
610     "rsc": "Нису дефинисани циљеви"
611   },
612   "123": {
613     "en": "File binding.gyp not found",
614     "rs": "Fajl binding.gyp nije pronađen",
615     "rsc": "Фајл binding.gyp није пронађен"
616   },
617   "124": {
618     "en": "Format not supported.",
619     "rs": "Format nije podržan.",
620     "rsc": "Формат није подржан."
621   },
622   "125": {
623     "en": "User requested loop stop!",
624     "rs": "Korisnik je zatražio zaustavljanje petlje!",
625     "rsc": "Корисник је захтевао заустављање петље!"
626   },
627   "126": {
628     "en": "Open canceled",
629     "rs": "Otvaranje je otkazano",
630     "rsc": "Отварање је отказано"
631   },
632   "127": {
633     "en": "Callback is missing",
634     "rs": "Callback nedostaje",
635     "rsc": "Callback недостаје"
636   },
637   "128": {
638     "en": "Unable to find files in folder",
```

```

639      "rs": "Nije moguće pronaći fajlove u direktorijumu",
640      "rsc": "Није могуће пронаћи фајлове у директоријуму"
641  },
642  "129": {
643      "en": "Save canceled",
644      "rs": "Čuvanje je otkazano",
645      "rsc": "Чување је отказано"
646  },
647  "130": {
648      "en": "Path override",
649      "rs": "Pregaćena putanja",
650      "rsc": "Прегажена путања"
651  },
652  "131": {
653      "en": "Script run aborted, file not saved.",
654      "rs": "Izvršavanje skripte prekinuto, fajl nije sačuvan.",
655      "rsc": "Извршавање скрипте прекинуто, фајл није сачуван."
656  },
657  "132": {
658      "en": "File open canceled",
659      "rs": "Otvaranje fajla je otkazano",
660      "rsc": "Отварање фајла је отказано"
661  },
662  "133": {
663      "en": "Script",
664      "rs": "Skripta",
665      "rsc": "Скрипта"
666  },
667  "134": {
668      "en": "already open",
669      "rs": "je već otvoren",
670      "rsc": "је већ отворен"
671  },
672  "135": {
673      "en": "For more information, visit",
674      "rs": "Za više informacija, posetite",
675      "rsc": "За више информација, посетите"
676  },
677  "136": {
678      "en": "Copyright",
679      "rs": "Autorsko pravo",
680      "rsc": "Ауторско право"
681  },
682  "137": {
683      "en": "This program is free software: you can redistribute it and/or
           modify it under the terms of the GNU Lesser General Public License as
           published by the Free Software Foundation, either version 3 of the
           License, or (at your option) any later version.",
684      "rs": "Ovaj program je besplatan softver: можете га redistribuirati i/ili
           modifikovati pod uslovima GNU manje opšte javne licence koju je
           objavila Fondacija za slobodni softver, bilo verzije 3 licence, ili (
           po vašem izboru) bilo koje kasnije verzije.",
685      "rsc": "Овај програм је бесплатан софтвер: можете га редистрибуирати и/или
           модификовати под условима ГНУ мање опште јавне лиценце коју је објави
           ла Фондација за слободни софтвер, било верзије 3 лиценце, или (по ваше

```

м избору) било које касније верзије."

686 },
687 "138": {
688 "en": "This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.",
689 "rs": "Ovaj program se distribuira u nadi da će biti koristan, ali BEZ
ИКАВЕ ГАРАНЦИЈЕ; чак и без implicitne garancije o ПРОДАЈНОСТИ или ПРИКЛ
АДНОСТИ ЗА ОДРЕЂЕНУ НАМЕНУ. Погледајте GNU мање општу јавну лиценцу за
више детаља.",
690 "rsc": "Овај програм се дистрибуира у нади да ће бити користан, али БЕЗ ИК
АКВЕ ГАРАНЦИЈЕ; чак и без имплицитне гаранције о ПРОДАЈНОСТИ или ПРИКЛ
АДНОСТИ ЗА ОДРЕЂЕНУ НАМЕНУ. Погледајте ГНУ мање општу јавну лиценцу за
више детаља."
691 },
692 "139": {
693 "en": "For help type",
694 "rs": "Za pomoć unesite",
695 "rsc": "За помоћ унесите"
696 },
697 "140": {
698 "en": "No saved paths.",
699 "rs": "Nema sačuvanih putanja.",
700 "rsc": "Нема сачуваних путања."
701 },
702 "141": {
703 "en": "Choose N-API module folder",
704 "rs": "Izaberite direktorijum N-API modula",
705 "rsc": "Изаберите директоријум N-API модула"
706 },
707 "142": {
708 "en": "Choose folder",
709 "rs": "Izaberite direktorijum",
710 "rsc": "Изаберите директоријум"
711 },
712 "143": {
713 "en": "Save figure",
714 "rs": "Sačuvajte grafik",
715 "rsc": "Сачувайте график"
716 },
717 "144": {
718 "en": "Save file",
719 "rs": "Sačuvajte datoteku",
720 "rsc": "Сачувайте датотеку"
721 },
722 "145": {
723 "en": "Save file",
724 "rs": "Sačuvajte datoteku",
725 "rsc": "Сачувайте датотеку"
726 },
727 "146": {
728 "en": "Load from file",
729 "rs": "Učitajte iz datoteke",
730 "rsc": "Учитајте из датотеке"

```
731 },
732 "147": {
733     "en": "Load file",
734     "rs": "Učitajte datoteku",
735     "rsc": "Учитајте датотеку"
736 },
737 "148": {
738     "en": "Choose working directory",
739     "rs": "Izaberite radni direktorijum",
740     "rsc": "Изаберите радни директоријум"
741 },
742 "149": {
743     "en": "Set",
744     "rs": "Postavi",
745     "rsc": "Постави"
746 },
747 "150": {
748     "en": "Save log file",
749     "rs": "Sačuvajte datoteku dnevnika",
750     "rsc": "Сачувајте датотеку дневника"
751 },
752 "151": {
753     "en": "Save log",
754     "rs": "Sačuvajte dnevnik",
755     "rsc": "Сачувајте дневник"
756 },
757 "152": {
758     "en": "Plot save as",
759     "rs": "Sačuvaj grafikon kao",
760     "rsc": "Сачувај графикон као"
761 },
762 "153": {
763     "en": "Zoom part of the plot",
764     "rs": "Zumiraj deo grafikona",
765     "rsc": "Зумирај део графикона"
766 },
767 "154": {
768     "en": "Pan plot",
769     "rs": "Pomeranje grafikona",
770     "rsc": "Померање графикона"
771 },
772 "155": {
773     "en": "Rotate plot",
774     "rs": "Rotiraj grafikon",
775     "rsc": "Ротирај графикон"
776 },
777 "156": {
778     "en": "Zoom in",
779     "rs": "Uvećaj",
780     "rsc": "Увећај"
781 },
782 "157": {
783     "en": "Zoom out",
784     "rs": "Umanji",
785     "rsc": "Умањи"
```

```
786 },
787 "158": {
788     "en": "Fit",
789     "rs": "Prilagodi",
790     "rsc": "Прилагоди"
791 },
792 "159": {
793     "en": "Save As",
794     "rs": "Sačuvaj kao",
795     "rsc": "Сачувај као"
796 },
797 "160": {
798     "en": "Zoom",
799     "rs": "Zumiraj",
800     "rsc": "Зумирај"
801 },
802 "161": {
803     "en": "Pan",
804     "rs": "Pomeranje",
805     "rsc": "Померање"
806 },
807 "162": {
808     "en": "Rotate",
809     "rs": "Rotiraj",
810     "rsc": "Ротирај"
811 },
812 "163": {
813     "en": "Zoom in",
814     "rs": "Uvećaj",
815     "rsc": "Увећај"
816 },
817 "164": {
818     "en": "Zoom out",
819     "rs": "Umanji",
820     "rsc": "Умањи"
821 },
822 "165": {
823     "en": "Fit",
824     "rs": "Prilagodi",
825     "rsc": "Прилагоди"
826 },
827 "166": {
828     "en": "Show timestamp",
829     "rs": "Prikaži vremensku oznaku",
830     "rsc": "Прикажи временску ознаку"
831 },
832 "167": {
833     "en": "Turn on auto scroll",
834     "rs": "Uključite automatsko pomeranje",
835     "rsc": "Укључите аутоматско померање"
836 },
837 "168": {
838     "en": "Search",
839     "rs": "Pretraga",
840     "rsc": "Претрага"
```

```
841 },
842 "169": {
843     "en": "Show search dialog",
844     "rs": "Prikaži dijalog za pretragu",
845     "rsc": "Прикажи дијалог за претрагу"
846 },
847 "170": {
848     "en": "Failed to compile module!",
849     "rs": "Neuspešno kompajliranje modula!",
850     "rsc": "Неуспешно компајлирање модула!"
851 },
852 "171": {
853     "en": "Unable to recompile module, try reseting sandbox with <span class='eval-code'> resetSandbox()</span> function.",
854     "rs": "Nije moguće rekompajlirati modul, pokušajte reset sandbox-a sa <span class='eval-code'>resetSandbox()</span> funkcijom.",
855     "rsc": "Није могуће рекомпајлирати модул, покушајте ресет sandbox-а са <span class='eval-code'>resetSandbox()</span> функцијом."
856 },
857 "172": {
858     "en": "Figure is not opened.",
859     "rs": "Grafik nije otvoren.",
860     "rsc": "График није отворен."
861 },
862 "173": {
863     "en": "Source directory does not exist.",
864     "rs": "Izvorni direktorijum ne postoji.",
865     "rsc": "Изворни директоријум не постоји."
866 },
867 "174": {
868     "en": "Failed to open specified file in new window!",
869     "rs": "Neuspešno otvaranje specificirane datoteke u novom prozoru!",
870     "rsc": "Неуспешно отварање специфициране датотеке у новом прозору!"
871 },
872 "175": {
873     "en": "Wait for lib to be loaded!",
874     "rs": "Sačekajte da se biblioteka učita!",
875     "rsc": "Сачекајте да се библиотека учита!"
876 },
877 "176": {
878     "en": "Arrays must be of the same length!",
879     "rs": "Nizovi moraju biti iste dužine!",
880     "rsc": "Низови морају бити исте дужине!"
881 },
882 "177": {
883     "en": "Invalid input types!",
884     "rs": "Nevažeći tipovi ulaza!",
885     "rsc": "Неважећи типови улаза!"
886 },
887 "178": {
888     "en": "Array size does not match the dimensions provided.",
889     "rs": "Veličina niza ne odgovara datim dimenzijama.",
890     "rsc": "Величина низа не одговара датим димензијама."
891 },
892 "179": {
```

```
893     "en": "FreeCAD instance already active!",
894     "rs": "FreeCAD instanca je već aktivna!",
895     "rsc": "FreeCAD инстанца је већ активна!"
896 },
897 "180": {
898     "en": "Could not start FreeCAD!",
899     "rs": "Nije moguće pokrenuti FreeCAD!",
900     "rsc": "Није могуће покренути FreeCAD!"
901 },
902 "181": {
903     "en": "Could not find FreeCAD TCP Server!",
904     "rs": "Nije moguće pronaći FreeCAD TCP server!",
905     "rsc": "Није могуће пронаћи FreeCAD TCP сервер!"
906 },
907 "182": {
908     "en": "No FreeCAD instance!",
909     "rs": "Nema FreeCAD instance!",
910     "rsc": "Нема FreeCAD инстанце!"
911 },
912 "183": {
913     "en": "File does not exist!",
914     "rs": "Datoteka ne postoji!",
915     "rsc": "Датотека не постоји!"
916 },
917 "184": {
918     "en": "is a forbidden name.",
919     "rs": "je zabranjeno ime.",
920     "rsc": "је забрањено име."
921 },
922 "185": {
923     "en": "Invalid variable declaration",
924     "rs": "Nevažeća deklaracija promenljive",
925     "rsc": "Неважећа декларација променљиве"
926 },
927 "186": {
928     "en": "Invalid function declaration",
929     "rs": "Nevažeća deklaracija funkcije",
930     "rsc": "Неважећа декларација функције"
931 },
932 "187": {
933     "en": "Invalid class declaration",
934     "rs": "Nevažeća deklaracija klase",
935     "rsc": "Неважећа декларација класе"
936 },
937 "188": {
938     "en": "Invalid import",
939     "rs": "Nevažeći import",
940     "rsc": "Неважећи импорт"
941 },
942 "189": {
943     "en": "Increment dx cannot be zero.",
944     "rs": "Povećanje dx ne može biti nula.",
945     "rsc": "Повећање dx не може бити нула."
946 },
947 "190": {
```

```
948     "en": "Input must be an array.",
949     "rs": "Ulag mora biti niz.",
950     "rsc": "Улаз мора бити низ."
951 },
952 "191": {
953     "en": "Array is empty or contains only NaN values.",
954     "rs": "Niz je prazan ili sadrži samo NaN vrednosti.",
955     "rsc": "Низ је празан или садржи само NaN вредности."
956 },
957 "192": {
958     "en": "Input must be a number, a complex number object, or an array
959     thereof.",
959     "rs": "Ulag mora biti broj, objekat kompleksnog broja ili niz takvih
960     objekata.",
960     "rsc": "Улаз мора бити број, објекат комплексног броја или низ таквих обје
961     ката."
961 },
962 "193": {
963     "en": "The file is not a valid OFF one.",
964     "rs": "Datoteka nije validna OFF datoteka.",
965     "rsc": "Датотека није валидна OFF датотека."
966 },
967 "194": {
968     "en": "Incomplete header information in OFF file.",
969     "rs": "Nepotpune informacije u zaglavlju OFF datoteke.",
970     "rsc": "Непотпуне информације у заглављу OFF датотеке."
971 },
972 "195": {
973     "en": "Problem in reading vertices.",
974     "rs": "Problem pri čitanju tačaka.",
975     "rsc": "Проблем при читању тачака."
976 },
977 "196": {
978     "en": "Problem in reading faces.",
979     "rs": "Problem pri čitanju stranica.",
980     "rsc": "Проблем при читању страница."
981 },
982 "197": {
983     "en": "Step size cannot be zero.",
984     "rs": "Veličina koraka ne može biti nula.",
985     "rsc": "Величина корака не може бити нула."
986 },
987 "198": {
988     "en": "Step size does not align with start and end values.",
989     "rs": "Veličina koraka se ne slaže sa početnim i krajnjim vrednostima.",
990     "rsc": "Величина корака се не слаже са почетним и крајњим вредностима."
991 },
992 "199": {
993     "en": "No file for window!",
994     "rs": "Nema datoteke za prozor!",
995     "rsc": "Нема датотеке за прозор!"
996 },
997 "200": {
998     "en": "Alias 'as' is required when import is '*' in module",
999     "rs": "Alias 'as' je potreban kada je import '*' u modulu",
```

```
1000     "rsc": "Alias 'as' је потребан када је import '*' у модулу "
1001 },
1002 "201": {
1003     "en": "No connection with OMC. Create a new instance of OpenModelicaLink
1004         session.",
1005     "rs": "Nema veze sa OMC-om. Kreirajte novu instancu OpenModelicaLink
1006         sesije.",
1007     "rsc": "Нема везе са OMC-ом. Креирајте нову инстанцу OpenModelicaLink сеси
1008         је."
1009 },
1010 "202": {
1011     "en": "Model is not Linearized",
1012     "rs": "Model nije linearizovan",
1013     "rsc": "Модел није линеаризован"
1014 },
1015 "203": {
1016     "en": "Filename and modelname are required.",
1017     "rs": "Ime datoteke i ime modela su obavezni.",
1018     "rsc": "Име датотеке и име модела су обавезни."
1019 },
1020 "204": {
1021     "en": "XML file is not generated.",
1022     "rs": "XML datoteka nije generisana.",
1023     "rsc": "XML датотека није генерисана."
1024 },
1025 "205": {
1026     "en": "Model cannot be Simulated: executable not found.",
1027     "rs": "Model se ne može simulirati: izvršna datoteka nije pronađena.",
1028     "rsc": "Модел се не може симулирати: извршна датотека није пронађена."
1029 },
1030 "206": {
1031     "en": "Model cannot be Simulated: xmlfile not found.",
1032     "rs": "Model se ne može simulirati: XML datoteka nije pronađena.",
1033     "rsc": "Модел се не може симулирати: XML датотека није пронађена."
1034 },
1035 "207": {
1036     "en": "Linearization cannot be performed: ",
1037     "rs": "Linearizacija se ne može izvršiti: ",
1038     "rsc": "Линеаризација се не може извршити: "
1039 },
1040 "208": {
1041     "en": "Result File does not exist!",
1042     "rs": "Datoteka sa rezultatima ne postoji!",
1043     "rsc": "Датотека са резултатима не постоји!"
1044 },
1045 "209": {
1046     "en": " is not a parameter",
1047     "rs": " nije parametar",
1048     "rsc": " није параметар"
1049 },
1050 "210": {
1051     "en": " is not a Simulation Option",
1052     "rs": " nije opcija za simulaciju",
1053     "rsc": " није опција за симулацију"
1054 },
1055 }
```

```
1052 "211": {
1053     "en": " is not a Linearization Option",
1054     "rs": " nije opcija za linearizaciju",
1055     "rsc": " није опција за линеаризацију"
1056 },
1057 "212": {
1058     "en": " is not an Input",
1059     "rs": " nije ulaz",
1060     "rsc": " није улаз"
1061 },
1062 "213": {
1063     "en": "You can see current workspace in main window. Would you like to
1064         continue?",
1064     "rs": "Možete videti trenutni radni prostor u glavnom prozoru. Da li želite
1065         da nastavite?",  

1065     "rsc": "Можете видети тренутни радни простор у главном прозору. Да ли желите да наставите?"
1066 },
1067 "214": {
1068     "en": "yes",
1069     "rs": "da",
1070     "rsc": "да"
1071 },
1072 "215": {
1073     "en": "no",
1074     "rs": "ne",
1075     "rsc": "не"
1076 },
1077 "216": {
1078     "en": "Code arrived to breakpoint at line",
1079     "rs": "Kod je dostigao breakpoint na liniji",
1080     "rsc": "Код је досегао breakpoint на линији"
1081 },
1082 "217": {
1083     "en": "Code paused at line",
1084     "rs": "Kod je pauziran na liniji",
1085     "rsc": "Код је паузиран на линији"
1086 },
1087 "218": {
1088     "en": "No help found for ",
1089     "rs": "Nije pronadjena dokumentacija za ",
1090     "rsc": "Није пронађена документација за "
1091 },
1092 "219": {
1093     "en": "Open documentation",
1094     "rs": "Otvorite dokumentaciju",
1095     "rsc": "Отворите документацију"
1096 },
1097 "220": {
1098     "en": "No source code found for ",
1099     "rs": "Nije pronadjen kod za ",
1100     "rsc": "Није пронађен код за "
1101 },
1102 "221": {
1103     "en": "Reset",
```

```
1104     "rs": "Reset",
1105     "rsc": "Песет"
1106   },
1107   "222": {
1108     "en": "Unable to find audio/video device!",
1109     "rs": "Nije moguće pronaći audio/video uređaj!",
1110     "rsc": "Није могуће пронаћи аудио/видео уређај!"
1111   },
1112   "223": {
1113     "en": "Number of commands in history",
1114     "rs": "Broj komandi u istoriji",
1115     "rsc": "Број команди у историји"
1116   },
1117   "224": {
1118     "en": "Program arduino-cli not found.",
1119     "rs": "Program arduino-cli nije pronadjen.",
1120     "rsc": "Програм arduino-cli није пронађен."
1121   },
1122   "225": {
1123     "en": "Configuration file for arduino-cli not found.",
1124     "rs": "Konfiguracioni fajl za arduino-cli nije pronadjen.",
1125     "rsc": "Конфигурациони фајл за arduino-cli није пронађен."
1126   },
1127   "226": {
1128     "en": "Submit command to workspace.",
1129     "rs": "Pošalji komandu radnom prostoru.",
1130     "rsc": "Пошаљи команду радном простору."
1131   },
1132   "227": {
1133     "en": "Compile",
1134     "rs": "Kompajliraj",
1135     "rsc": "Компајлирај"
1136   },
1137   "228": {
1138     "en": "Upload",
1139     "rs": "Otpremi",
1140     "rsc": "Отпреми"
1141   },
1142   "229": {
1143     "en": "Program file for arduino-cli not found.",
1144     "rs": "Fajl programa za arduino-cli nije pronadjen.",
1145     "rsc": "Фајл програма за arduino-cli није пронађен."
1146   },
1147   "230": {
1148     "en": "No available ports.",
1149     "rs": "Nema dostupnih portova.",
1150     "rsc": "Нема доступних портова."
1151   },
1152   "231": {
1153     "en": "Choose",
1154     "rs": "Izaberi",
1155     "rsc": "Изабери"
1156   },
1157   "232": {
1158     "en": "Port",
```

```
1159      "rs": "Port",
1160      "rsc": "Порт"
1161    },
1162    "233": {
1163      "en": "Baudrate",
1164      "rs": "Brzina",
1165      "rsc": "Брзина"
1166    },
1167    "234": {
1168      "en": "Type your message...",
1169      "rs": "Unesite svoju poruku...",
1170      "rsc": "Унесите своју поруку..."
1171    },
1172    "235": {
1173      "en": "Unknown method!",
1174      "rs": "Nepoznati metod!",
1175      "rsc": "Непознати метод!"
1176    },
1177    "236": {
1178      "en": "Save video",
1179      "rs": "Sačuvajte video",
1180      "rsc": "Сачувавте видео"
1181    },
1182    "237": {
1183      "en": "Unable to check for update without internet.",
1184      "rs": "Nije moguće proveriti da li postoji novija verzija bez interneta.",
1185      "rsc": "Није могуће проверити да ли постоји новија верзија без интернета."
1186    },
1187    "238": {
1188      "en": "Update available at: ",
1189      "rs": "Novija verzija dostupna na: ",
1190      "rsc": "Новија верзија доступна на: "
1191    },
1192    "239": {
1193      "en": "Choose presentation folder",
1194      "rs": "Izaberi folder prezentacije",
1195      "rsc": "Изабери фолдер презентације"
1196    },
1197    "240": {
1198      "en": "Invalid presentation folder!",
1199      "rs": "Neispravan folder prezentacije!",
1200      "rsc": "Неисправан фолдер презентације!"
1201    },
1202    "241": {
1203      "en": "Packed presentation available at: ",
1204      "rs": "Spakovana prezentacija je dostupna na adresi: ",
1205      "rsc": "Спакована презентација је доступна на адреси: "
1206    },
1207    "242": {
1208      "en": "Presentation display",
1209      "rs": "Prikaz prezentacije",
1210      "rsc": "Приказ презентације"
1211    },
1212    "243": {
1213      "en": "Presentation code",
```

```

1214      "rs": "Kod prezentacije",
1215      "rsc": "Код презентације"
1216    },
1217    "244": {
1218      "en": "PDF presentation available at: ",
1219      "rs": "PDF prezентација је доступна на адреси: ",
1220      "rsc": "PDF презентација је доступна на адреси: "
1221    },
1222    "245": {
1223      "en": "Fold slides",
1224      "rs": "Preklopi slajdove",
1225      "rsc": "Преклопи слайдове"
1226    },
1227    "246": {
1228      "en": "Unfold slides",
1229      "rs": "Otklopi slajdove",
1230      "rsc": "Отклопи слайдове"
1231    },
1232    "247": {
1233      "en": "Choose JSON file",
1234      "rs": "Izaberite JSON dokument",
1235      "rsc": "Изаберите JSON документ"
1236    },
1237    "248": {
1238      "en": "Invalid file!",
1239      "rs": "Neispravan dokument!",
1240      "rsc": "Неисправан документ!"
1241  }
1242 }
```

Listing 4 - lang.json

3 cpp

```

1 // AlphaShape3D - alpha-shape-3d.cpp
2 // Author: Milos Petrasinovic <mpetrasinovic@prdc.rs>
3 // PR-DC, Republic of Serbia
4 // info@prdc.rs
5 // -----
6
7 #include "alpha-shape-3d.h"
8
9 namespace alpha_shape_3d_ns {
10
11 #ifdef PROFILE_SOLVER_MP_6RSS
12 // Function to start the timer and return the start time
13 time_point<steady_clock> tic() {
14   return steady_clock::now();
15 }
16
17 // Function to stop the timer and return the elapsed time
18 long toc(const time_point<steady_clock>& startTime) {
19   return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
20 }
```

```

21 #endif
22
23 // Function to get current time
24 std::string getCurrentTime() {
25   // get current time
26   auto now = system_clock::now();
27
28   // get number of milliseconds for the current second
29   // (remainder after division into seconds)
30   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
31
32   // convert to std::time_t in order to convert to std::tm (broken time)
33   auto timer = system_clock::to_time_t(now);
34
35   // convert to broken time
36   std::tm bt = *std::localtime(&timer);
37
38   std::ostringstream oss;
39
40   oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
41   oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
42
43   return oss.str();
44 }
45
46 // Function to console log data
47 int consoleLog(uint8_t level, const char* format, ...) {
48 #ifdef DEBUG_SOLVER_MP_6RSS_LEVEL
49   if(level <= DEBUG_SOLVER_MP_6RSS_LEVEL) {
50     printf("\033[0;33m[%s SolverMP6RSS]\033[0m ", getCurrentTime().c_str());
51     va_list vl;
52     va_start(vl, format);
53     auto ret = vprintf(format, vl);
54     va_end(vl);
55     printf("\n");
56     return ret;
57   }
58 #endif
59   return 0;
60 }
61
62 // AlphaShape3D()
63 // Object constructor
64 // -----
65 AlphaShape3D::AlphaShape3D(const Napi::CallbackInfo& info) : Napi::ObjectWrap<
66   AlphaShape3D>(info) {
67 #ifdef DEBUG_ALPHA_SHAPE_3D
68   consoleLog(0, "Called constructor");
69 #endif
70   this->alphaShape = nullptr;
71   this->delaunayTriangulation = nullptr;
72 }
73 // ~AlphaShape3D()
74 // Object destructor

```

```

75 // -----
76 AlphaShape3D::~AlphaShape3D( void ) {
77     if( this->delaunayTriangulation ){
78         delete this->delaunayTriangulation;
79         this->delaunayTriangulation = nullptr;
80     }
81     if( this->alphaShape ){
82         delete this->alphaShape;
83         this->alphaShape = nullptr;
84     }
85 }
86
87 // Init() function
88 // -----
89 Napi::Object AlphaShape3D::Init(Napi::Env env, Napi::Object exports) {
90     Napi::Function func = DefineClass(env, "AlphaShape3D", {
91         InstanceMethod("newShape", &AlphaShape3D::NewShapeJS),
92         InstanceMethod("getAlpha", &AlphaShape3D::GetAlphaJS),
93         InstanceMethod("setAlpha", &AlphaShape3D::SetAlphaJS),
94         InstanceMethod("getNumRegions", &AlphaShape3D::GetNumRegionsJS),
95         InstanceMethod("getAlphaSpectrum", &AlphaShape3D::GetAlphaSpectrumJS),
96         InstanceMethod("getCriticalAlpha", &AlphaShape3D::GetCriticalAlphaJS),
97         InstanceMethod("getSurfaceArea", &AlphaShape3D::GetSurfaceAreaJS),
98         InstanceMethod("getVolume", &AlphaShape3D::GetVolumeJS),
99         InstanceMethod("getBoundaryFacets", &AlphaShape3D::GetBoundaryFacetsJS),
100        InstanceMethod("writeBoundaryFacets", &AlphaShape3D::WriteBoundaryFacetsJS
101            ),
102        InstanceMethod("checkInShape", &AlphaShape3D::CheckInShapeJS),
103        InstanceMethod("writeOff", &AlphaShape3D::WriteOffJS),
104        InstanceMethod("getTriangulation", &AlphaShape3D::GetTriangulationJS),
105        InstanceMethod("getNearestNeighbor", &AlphaShape3D::GetNearestNeighborJS),
106        InstanceMethod("getSimplifiedShape", &AlphaShape3D::GetSimplifiedShapeJS),
107        InstanceMethod("removeUnusedPoints", &AlphaShape3D::RemoveUnusedPointsJS)
108    });
109    Napi::FunctionReference* constructor = new Napi::FunctionReference();
110    *constructor = Napi::Persistent(func);
111    env.SetInstanceData(constructor);
112
113    exports.Set("AlphaShape3D", func);
114    return exports;
115}
116
117 // NewShapeJS() function
118 // -----
119 void AlphaShape3D::NewShapeJS(const Napi::CallbackInfo& info) {
120 #ifdef DEBUG_ALPHA_SHAPE_3D
121     consoleLog(0, "Called NewShapeJS()");
122 #endif
123
124     Napi::Env env = info.Env();
125     if( info.Length() < 1 || !info[0].IsArray() ){
126         Napi::TypeError::New(env, "Array of points expected").
127             ThrowAsJavaScriptException();
128     }
129 }
```

```

128
129 Napi::Array jsPoints = info[0].As<Napi::Array>();
130 uint32_t numPoints = jsPoints.Length();
131
132 // Clear existing data to prevent accumulation
133 this->inputPoints.resize(0, 0);
134 this->Points.clear();
135 this->Vertices.clear();
136
137 // Initialize inputPoints matrix
138 this->inputPoints.resize(numPoints, 3);
139
140 for(uint32_t i = 0; i < numPoints; i++){
141   Napi::Value point = jsPoints[i];
142   if(!point.isArray()){
143     Napi::TypeError::New(env, "Each point should be an array of 3 numbers")
144       .ThrowAsJavaScriptException();
145     return;
146   }
147   Napi::Array jsPoint = point.As<Napi::Array>();
148   if(jsPoint.Length() != 3){
149     Napi::TypeError::New(env, "Each point should have exactly 3 coordinates")
150       .ThrowAsJavaScriptException();
151     return;
152   }
153   for(uint32_t j = 0; j < 3; j++){
154     this->inputPoints(i, j) = jsPoint.Get(j).As<Napi::Number>().DoubleValue();
155   }
156 }
157 #ifdef DEBUG_ALPHA_SHAPE_3D
158   std::chrono::steady_clock::time_point begin =
159     std::chrono::steady_clock::now();
160 #endif
161
162 uint32_t n = this->inputPoints.numRows();
163
164 #ifdef DEBUG_ALPHA_SHAPE_3D
165   std::cout << "Reading " << n << " points " << std::endl;
166 #endif
167
168 for(std::size_t i = 0; i < n; i++){
169   this->Points.emplace_back(
170     Point(this->inputPoints(i, 0), this->inputPoints(i, 1), this->
171       inputPoints(i, 2)));
172   this->Vertices.emplace_back(std::make_pair(this->Points.back(), i));
173 }
174 #ifdef DEBUG_ALPHA_SHAPE_3D
175   std::cout << "Computing delaunay triangulation." << std::endl;
176 #endif
177
178 // Delete existing triangulation and alphaShape to prevent memory leaks

```

```

179 if (this->delaunayTriangulation) {
180     delete this->delaunayTriangulation;
181     this->delaunayTriangulation = nullptr;
182 }
183 if (this->alphaShape){
184     delete this->alphaShape;
185     this->alphaShape = nullptr;
186 }
187
188 this->delaunayTriangulation = new Dt(this->Vertices.begin() , this->Vertices .
189 end());
190
191 #ifdef DEBUG_ALPHA_SHAPE_3D
192     std::cout << "Number of triangulation cells is "
193         << this->delaunayTriangulation->number_of_finite_cells() << std::
194         endl;
195 #endif
196
197 this->triangulationMatrix.resize(this->delaunayTriangulation->
198     number_of_finite_cells() *4, 3);
199 uint64_t i_idx = 0;
200 for(Dt::Finite_cells_iterator cit = this->delaunayTriangulation->
201     finite_cells_begin();
202     cit != this->delaunayTriangulation->finite_cells_end(); cit++){
203     this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
204     this->triangulationMatrix(i_idx, 1) = cit->vertex(1)->info();
205     this->triangulationMatrix(i_idx, 2) = cit->vertex(2)->info();
206     i_idx++;
207     this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
208     this->triangulationMatrix(i_idx, 1) = cit->vertex(2)->info();
209     this->triangulationMatrix(i_idx, 2) = cit->vertex(3)->info();
210     i_idx++;
211     this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
212     this->triangulationMatrix(i_idx, 1) = cit->vertex(1)->info();
213     this->triangulationMatrix(i_idx, 2) = cit->vertex(3)->info();
214     i_idx++;
215 }
216
217 #ifdef DEBUG_ALPHA_SHAPE_3D
218     std::cout << "Computing alpha shapes." << std::endl;
219 #endif
220     this->alphaShape = new As3(*this->delaunayTriangulation , As3::GENERAL);
221
222     this->numAlphaValues = this->alphaShape->number_of_alphas();
223
224 #ifdef DEBUG_ALPHA_SHAPE_3D
225     std::cout << "Number of alpha values is "
226         << this->numAlphaValues << std::endl;
227     std::cout << "Max alpha value is "
228         << this->alphaShape->get_nth_alpha(this->numAlphaValues) << std::
229         endl;

```

```

229     std::cout << "Min of alpha value is "
230             << this->alphaShape->get_nth_alpha(1) << std::endl;
231 #ifdef PROFILE_SOLVER_MP_6RSS
232     std::chrono::steady_clock::time_point end =
233         std::chrono::steady_clock::now();
234     std::cout << "Time elapsed = "
235             << std::chrono::duration_cast<std::chrono::milliseconds>
236                 (end - begin).count()
237             << " ms" << std::endl;
238 #endif
239 #endif
240 }
241
242 // GetAlphaJS() function
243 // -----
244 Napi::Value AlphaShape3D::GetAlphaJS(const Napi::CallbackInfo& info) {
245 #ifdef DEBUG_ALPHA_SHAPE_3D
246     consoleLog(0, "Called GetAlphaJS()");
247 #endif
248
249     Napi::Env env = info.Env();
250     double result = this->getAlpha();
251     return Napi::Number::New(env, result);
252 }
253
254 // SetAlphaJS() function
255 // -----
256 void AlphaShape3D::SetAlphaJS(const Napi::CallbackInfo& info) {
257 #ifdef DEBUG_ALPHA_SHAPE_3D
258     consoleLog(0, "Called SetAlphaJS()");
259 #endif
260
261     Napi::Env env = info.Env();
262     if(info.Length() < 1 || !info[0].IsNumber()){
263         Napi::TypeError::New(env, "Number expected").ThrowAsJavaScriptException();
264     }
265     double alpha = info[0].As<Napi::Number>().DoubleValue();
266     this->setAlpha(alpha);
267 }
268
269 // GetNumRegionsJS() function
270 // -----
271 Napi::Value AlphaShape3D::GetNumRegionsJS(const Napi::CallbackInfo& info) {
272 #ifdef DEBUG_ALPHA_SHAPE_3D
273     consoleLog(0, "Called GetNumRegionsJS()");
274 #endif
275
276     Napi::Env env = info.Env();
277     double result = this->numRegions();
278     return Napi::Number::New(env, result);
279 }
280
281 // GetAlphaSpectrumJS() function
282 // -----
283 Napi::Value AlphaShape3D::GetAlphaSpectrumJS(const Napi::CallbackInfo& info) {

```

```

284 #ifdef DEBUG_ALPHA_SHAPE_3D
285   consoleLog(0, "Called GetAlphaSpectrumJS()");
286 #endif
287
288 Napi::Env env = info.Env();
289 Matrix spectrum = this->getAlphaSpectrum();
290 Napi::Array result = Napi::Array::New(env, spectrum.numCols());
291 for(size_t i = 0; i < spectrum.numCols(); i++){
292   result.Set(i, Napi::Number::New(env, spectrum(0, i)));
293 }
294 return result;
295 }
296
297 // GetCriticalAlphaJS() function
298 // -----
299 Napi::Value AlphaShape3D::GetCriticalAlphaJS(const Napi::CallbackInfo& info) {
300 #ifdef DEBUG_ALPHA_SHAPE_3D
301   consoleLog(0, "Called GetCriticalAlphaJS()");
302 #endif
303
304 Napi::Env env = info.Env();
305 if(info.Length() < 1 || !info[0].IsString()){
306   Napi::TypeError::New(env, "String expected").ThrowAsJavaScriptException();
307   return env.Null();
308 }
309 std::string type = info[0].As<Napi::String>().Utf8Value();
310 double result = this->getCriticalAlpha(type);
311 return Napi::Number::New(env, result);
312 }
313
314 // GetSurfaceAreaJS() function
315 // -----
316 Napi::Value AlphaShape3D::GetSurfaceAreaJS(const Napi::CallbackInfo& info) {
317 #ifdef DEBUG_ALPHA_SHAPE_3D
318   consoleLog(0, "Called GetSurfaceAreaJS()");
319 #endif
320
321 Napi::Env env = info.Env();
322 double result = this->getSurfaceArea();
323 return Napi::Number::New(env, result);
324 }
325
326 // GetVolumeJS() function
327 // -----
328 Napi::Value AlphaShape3D::GetVolumeJS(const Napi::CallbackInfo& info) {
329 #ifdef DEBUG_ALPHA_SHAPE_3D
330   consoleLog(0, "Called GetVolumeJS()");
331 #endif
332
333 Napi::Env env = info.Env();
334 double result = this->getVolume();
335 return Napi::Number::New(env, result);
336 }
337
338 // GetBoundaryFacetsJS() function

```

```

339 // -----
340 Napi::Value AlphaShape3D::GetBoundaryFacetsJS(const Napi::CallbackInfo& info)
341 {
342 #ifdef DEBUG_ALPHA_SHAPE_3D
343   consoleLog(0, "Called GetBoundaryFacetsJS ()");
344 #endif
345
346   Napi::Env env = info.Env();
347   Matrix facets = this->getBoundaryFacets();
348   Napi::Array result = Napi::Array::New(env, facets.numRows());
349   for (size_t i = 0; i < facets.numRows(); i++) {
350     Napi::Array facet = Napi::Array::New(env, 3);
351     for (size_t j = 0; j < 3; j++) {
352       facet.Set(j, Napi::Number::New(env, facets(i, j)));
353     }
354     result.Set(i, facet);
355   }
356   return result;
357 }
358 // WriteBoundaryFacetsJS() function
359 // -----
360 void AlphaShape3D::WriteBoundaryFacetsJS(const Napi::CallbackInfo& info) {
361 #ifdef DEBUG_ALPHA_SHAPE_3D
362   consoleLog(0, "Called WriteBoundaryFacetsJS ()");
363 #endif
364
365   Napi::Env env = info.Env();
366   if (info.Length() < 1 || !info[0].IsString()) {
367     Napi::TypeError::New(env, "String expected").ThrowAsJavaScriptException();
368   }
369   std::string filename = info[0].As<Napi::String>().Utf8Value();
370   this->writeBoundaryFacets(filename);
371 }
372 // CheckInShapeJS() function
373 // -----
374 Napi::Value AlphaShape3D::CheckInShapeJS(const Napi::CallbackInfo& info) {
375 #ifdef DEBUG_ALPHA_SHAPE_3D
376   consoleLog(0, "Called CheckInShapeJS ()");
377 #endif
378
379   Napi::Env env = info.Env();
380   if (info.Length() < 1 || !info[0].IsArray()) {
381     Napi::TypeError::New(env, "Array expected").ThrowAsJavaScriptException();
382     return env.Null();
383   }
384   Napi::Array jsArray = info[0].As<Napi::Array>();
385   Matrix QP(jsArray.Length(), 3);
386   for (size_t i = 0; i < jsArray.Length(); i++) {
387     Napi::Array point = jsArray.Get(i).As<Napi::Array>();
388     for (size_t j = 0; j < 3; j++) {
389       QP(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
390     }
391   }
392 }
```

```

393 Matrix result = this->checkInShape(QP);
394 Napi::Array jsResult = Napi::Array::New(env, result.numRows());
395 for(size_t i = 0; i < result.numRows(); i++){
396     jsResult.Set(i, Napi::Boolean::New(env, result(i, 0) != 0));
397 }
398 return jsResult;
399 }
400
401 // WriteOffJS() function
402 // -----
403 void AlphaShape3D::WriteOffJS(const Napi::CallbackInfo& info) {
404 #ifdef DEBUG_ALPHA_SHAPE_3D
405     consoleLog(0, "Called WriteOffJS()");
406 #endif
407
408 Napi::Env env = info.Env();
409 if(info.Length() < 3 || !info[0].IsString() || !info[1].IsArray() || !info[2].IsArray()){
410     Napi::TypeError::New(env, "Expected arguments: filename (string), points (array), facets (array)").ThrowAsJavaScriptException();
411 }
412
413 std::string filename = info[0].As<Napi::String>().Utf8Value();
414 Napi::Array jsPoints = info[1].As<Napi::Array>();
415 Napi::Array jsFacets = info[2].As<Napi::Array>();
416
417 Matrix Points(jsPoints.Length(), 3);
418 Matrix bf(jsFacets.Length(), 3);
419
420 for(size_t i = 0; i < jsPoints.Length(); i++){
421     Napi::Array point = jsPoints.Get(i).As<Napi::Array>();
422     for(size_t j = 0; j < 3; j++){
423         Points(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
424     }
425 }
426
427 for(size_t i = 0; i < jsFacets.Length(); i++){
428     Napi::Array facet = jsFacets.Get(i).As<Napi::Array>();
429     for(size_t j = 0; j < 3; j++){
430         bf(i, j) = facet.Get(j).As<Napi::Number>().DoubleValue();
431     }
432 }
433
434 this->writeOff(filename, Points, bf);
435 }
436
437 // GetTriangulationJS() function
438 // -----
439 Napi::Value AlphaShape3D::GetTriangulationJS(const Napi::CallbackInfo& info) {
440 #ifdef DEBUG_ALPHA_SHAPE_3D
441     consoleLog(0, "Called GetTriangulationJS()");
442 #endif
443
444 Napi::Env env = info.Env();
445 Matrix triangulation = this->getTriangulation();

```

```

446 Napi::Array result = Napi::Array::New(env, triangulation.numRows()) ;
447 for(size_t i = 0; i < triangulation.numRows(); i++){
448     Napi::Array row = Napi::Array::New(env, 3);
449     for(size_t j = 0; j < 3; j++){
450         row.Set(j, Napi::Number::New(env, triangulation(i, j)));
451     }
452     result.Set(i, row);
453 }
454 return result;
455 }
456
457 // GetNearestNeighborJS() function
458 // -----
459 Napi::Value AlphaShape3D::GetNearestNeighborJS(const Napi::CallbackInfo& info)
460 {
461 #ifdef DEBUG_ALPHA_SHAPE_3D
462     consoleLog(0, "Called GetNearestNeighborJS()");
463 #endif
464
465     Napi::Env env = info.Env();
466     if(info.Length() < 1 || !info[0].IsArray()){
467         Napi::TypeError::New(env, "Array expected").ThrowAsJavaScriptException();
468         return env.Null();
469     }
470     Napi::Array jsArray = info[0].As<Napi::Array>();
471     Matrix QP(jsArray.Length(), 3);
472     for(size_t i = 0; i < jsArray.Length(); i++){
473         Napi::Array point = jsArray.Get(i).As<Napi::Array>();
474         for(size_t j = 0; j < 3; j++){
475             QP(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
476         }
477     }
478     std::pair<Matrix, Matrix> result = this->getNearestNeighbor(QP);
479
480     Napi::Object jsResult = Napi::Object::New(env);
481
482     Napi::Array indices = Napi::Array::New(env, result.first.numRows());
483     Napi::Array distances = Napi::Array::New(env, result.second.numRows());
484
485     for(size_t i = 0; i < result.first.numRows(); i++){
486         indices.Set(i, Napi::Number::New(env, result.first(i, 0)));
487         distances.Set(i, Napi::Number::New(env, result.second(i, 0)));
488     }
489
490     jsResult.Set("indices", indices);
491     jsResult.Set("distances", distances);
492     return jsResult;
493 }
494
495 // GetSimplifiedShapeJS() function
496 // -----
497 Napi::Value AlphaShape3D::GetSimplifiedShapeJS(const Napi::CallbackInfo& info)
498 {
499 #ifdef DEBUG_ALPHA_SHAPE_3D

```

```

499   consoleLog(0, "Called GetSimplifiedShapeJS ()");
500 #endif
501
502 Napi::Env env = info.Env();
503 std::pair<Matrix, Matrix> result;
504
505 if(info.Length() == 0){
506     result = this->getSimplifiedShape();
507 }
508 else if(info.Length() == 1){
509     if(info[0].IsNumber()){
510         double stop_ratio = info[0].As<Napi::Number>().DoubleValue();
511         result = this->getSimplifiedShape(stop_ratio);
512     }
513     else if(info[0].IsString()){
514         std::string filename = info[0].As<Napi::String>().Utf8Value();
515         result = this->getSimplifiedShape(filename);
516     }
517     else{
518         Napi::TypeError::New(env, "Invalid argument type").
519             ThrowAsJavaScriptException();
520         return env.Null();
521     }
522     else if(info.Length() == 2 && info[0].IsNumber() && info[1].IsString()){
523         double stop_ratio = info[0].As<Napi::Number>().DoubleValue();
524         std::string filename = info[1].As<Napi::String>().Utf8Value();
525         result = this->getSimplifiedShape(stop_ratio, filename);
526     }
527     else{
528         Napi::TypeError::New(env, "Invalid arguments").ThrowAsJavaScriptException
529             ();
530         return env.Null();
531     }
532
533 Napi::Object jsResult = Napi::Object::New(env);
534 Napi::Array points = Napi::Array::New(env, result.first.numRows());
535 Napi::Array facets = Napi::Array::New(env, result.second.numRows());
536
537 for(size_t i = 0; i < result.first.numRows(); i++){
538     Napi::Array point = Napi::Array::New(env, 3);
539     for(size_t j = 0; j < 3; j++){
540         point.Set(j, Napi::Number::New(env, result.first(i, j)));
541     }
542     points.Set(i, point);
543 }
544
545 for(size_t i = 0; i < result.second.numRows(); i++){
546     Napi::Array facet = Napi::Array::New(env, 3);
547     for(size_t j = 0; j < 3; j++){
548         facet.Set(j, Napi::Number::New(env, result.second(i, j)));
549     }
550     facets.Set(i, facet);
551 }
```



```

605     Napi::Array facet = Napi::Array::New(env, 3);
606     for(size_t j = 0; j < 3; j++){
607         facet.Set(j, Napi::Number::New(env, result.second(i, j)));
608     }
609     facets.Set(i, facet);
610 }
611
612 jsResult.Set("points", points);
613 jsResult.Set("facets", facets);
614 return jsResult;
615 }
616
617 // getAlpha() function
618 // -----
619 double AlphaShape3D::getAlpha(void) {
620 #ifdef DEBUG_ALPHA_SHAPE_3D
621     consoleLog(0, "Called getAlpha()");
622 #endif
623
624     return this->alphaShape->get_alpha();
625 }
626
627 // setAlpha() function
628 // -----
629 void AlphaShape3D::setAlpha(double alpha) {
630 #ifdef DEBUG_ALPHA_SHAPE_3D
631     consoleLog(0, "Called setAlpha()");
632 #endif
633
634     this->surface_mesh.clear();
635     this->alphaShape->set_alpha(alpha);
636
637 #ifdef DEBUG_ALPHA_SHAPE_3D
638     std::cout << "Number of solid components for alpha " << alpha
639     << " is " << this->numRegions() << std::endl;
640     std::list<As3::Cell_handle> cells;
641     std::list<As3::Facet> facets;
642     std::list<As3::Edge> edges;
643     std::list<As3::Vertex_handle> vertices;
644     this->alphaShape->get_alpha_shape_cells(std::back_inserter(cells),
645         As3::INTERIOR);
646     this->alphaShape->get_alpha_shape_facets(std::back_inserter(facets),
647         As3::REGULAR);
648     this->alphaShape->get_alpha_shape_facets(std::back_inserter(facets),
649         As3::SINGULAR);
650     this->alphaShape->get_alpha_shape_edges(std::back_inserter(edges),
651         As3::SINGULAR);
652     this->alphaShape->get_alpha_shape_vertices(std::back_inserter(vertices),
653         As3::REGULAR);
654     std::cout << "Number of interior tetrahedra is "
655     << cells.size() << std::endl;
656     std::cout << "Number of boundary facets is "
657     << facets.size() << std::endl;
658     std::cout << "Number of singular edges is "
659     << edges.size() << std::endl;

```

```

660     std::cout << "Number of singular vertices is "
661             << vertices.size() << std::endl;
662     std::cout << "Boundary surface construction." << std::endl;
663 #endif
664
665     std::vector<As3::Facet> bfacets;
666     this->alphaShape->get_alpha_shape_facets(std::back_inserter(bfacets),
667                                               As3::REGULAR);
668
669     std::size_t nbf = bfacets.size();
670     std::vector<CGAL_Polygon> polygons;
671     CGAL_Polygon p;
672
673     for (std::size_t i = 0; i < nbf; i++) {
674         if (this->alphaShape->classify(bfacets[i].first) != As3::EXTERIOR)
675             bfacets[i] = this->alphaShape->mirror_facet(bfacets[i]);
676
677         int32_t indices[3] = {
678             (bfacets[i].second + 1) % 4,
679             (bfacets[i].second + 2) % 4,
680             (bfacets[i].second + 3) % 4,
681         };
682
683         // Consistent orientation
684         if (bfacets[i].second % 2 == 0) std::swap(indices[0], indices[1]);
685
686         p.clear();
687         for (uint8_t j = 0; j < 3; j++) {
688             p.push_back(bfacets[i].first->vertex(indices[j])->info());
689         }
690         polygons.push_back(p);
691     }
692
693     PMP::polygon_soup_to_polygon_mesh(this->Points,
694                                         polygons, this->surface_mesh);
695 }
696
697 // numRegions() function
698 // -----
699 double AlphaShape3D::numRegions(void) {
700 #ifdef DEBUG_ALPHA_SHAPE_3D
701     consoleLog(0, "Called numRegions()");
702 #endif
703
704     return this->alphaShape->number_of_solid_components();
705 }
706
707 // getAlphaSpectrum() function
708 // -----
709 Matrix AlphaShape3D::getAlphaSpectrum() {
710 #ifdef DEBUG_ALPHA_SHAPE_3D
711     consoleLog(0, "Called getAlphaSpectrum()");
712 #endif
713
714     Matrix a(1, this->numAlphaValues);

```

```

715     for(uint32_t i = 0; i < this->numAlphaValues; i++){
716         a(0, i) = this->alphaShape->get_nth_alpha(i + 1);
717     }
718     return a;
719 }
720
721 // getCriticalAlpha() function
722 // -----
723 double AlphaShape3D::getCriticalAlpha(std::string type) {
724 #ifdef DEBUG_ALPHA_SHAPE_3D
725     consoleLog(0, "Called getCriticalAlpha()");
726 #endif
727
728     if(type == "all-points"){
729         return this->alphaShape->find_alpha_solid();
730     }
731     else if(type == "one-region"){
732         return *this->alphaShape->find_optimal_alpha(1);
733     }
734     else{
735         return nan("");
736     }
737 }
738
739 // getSurfaceArea() function
740 // -----
741 double AlphaShape3D::getSurfaceArea(void) {
742 #ifdef DEBUG_ALPHA_SHAPE_3D
743     consoleLog(0, "Called getSurfaceArea()");
744 #endif
745
746     return PMP::area(this->surface_mesh);
747 }
748
749 // getVolume() function
750 // -----
751 double AlphaShape3D::getVolume(void) {
752 #ifdef DEBUG_ALPHA_SHAPE_3D
753     consoleLog(0, "Called getVolume()");
754 #endif
755
756     return PMP::volume(this->surface_mesh);
757 }
758
759 // getBoundaryFacets() function
760 // -----
761 Matrix AlphaShape3D::getBoundaryFacets(void) {
762 #ifdef DEBUG_ALPHA_SHAPE_3D
763     consoleLog(0, "Called getBoundaryFacets()");
764 #endif
765
766     Matrix bf(this->surface_mesh.number_of_faces(), 3);
767     for(Mesh::Face_index face_index : this->surface_mesh.faces()){
768         CGAL::Vertex_around_face_circulator<Mesh>
769             vcirc(this->surface_mesh.halfedge(face_index), this->surface_mesh);

```

```

770     bf(face_index.idx(), 0) = *vcirc++;
771     bf(face_index.idx(), 1) = *vcirc++;
772     bf(face_index.idx(), 2) = *vcirc++;
773 }
774 return bf;
775 }
776
777 // getBoundaryFacets() function with filename
778 // -----
779 Matrix AlphaShape3D::getBoundaryFacets(std::string filename) {
780     Matrix bf = this->getBoundaryFacets();
781     this->writeOff(filename, this->inputPoints, bf);
782     return bf;
783 }
784
785 // writeBoundaryFacets() function
786 // -----
787 void AlphaShape3D::writeBoundaryFacets(std::string filename) {
788 #ifdef DEBUG_ALPHA_SHAPE_3D
789     consoleLog(0, "Called writeBoundaryFacets()");
790 #endif
791
792     this->writeOff(filename, this->inputPoints, this->getBoundaryFacets());
793 }
794
795 // checkInShape() function
796 // -----
797 Matrix AlphaShape3D::checkInShape(Matrix QP) {
798 #ifdef DEBUG_ALPHA_SHAPE_3D
799     consoleLog(0, "Called checkInShape()");
800 #endif
801
802     Matrix tf(QP.numRows(), 1);
803     for(uint32_t i = 0; i < QP.numRows(); i++){
804         tf(i, 0) = this->alphaShape->classify(Point(QP(i, 0), QP(i, 1), QP(i, 2)));
805     }
806     return tf;
807 }
808
809 // getTriangulation() function
810 // -----
811 Matrix AlphaShape3D::getTriangulation(void) {
812 #ifdef DEBUG_ALPHA_SHAPE_3D
813     consoleLog(0, "Called getTriangulation()");
814 #endif
815
816     return this->triangulationMatrix;
817 }
818
819 // getNearestNeighbor() function
820 // -----
821 std::pair<Matrix, Matrix> AlphaShape3D::getNearestNeighbor(Matrix QP) {
822 #ifdef DEBUG_ALPHA_SHAPE_3D
823     consoleLog(0, "Called getNearestNeighbor()");

```

```

824 #endif
825
826 Matrix I(QP.numRows(), 1);
827 Matrix D(QP.numRows(), 1);
828
829 Mesh surface_mesh_s(this->surface_mesh);
830 PMP::remove_isolated_vertices(surface_mesh_s);
831
832 std::vector<Point> points;
833 std::vector<uint32_t> vs;
834 for(Mesh::Vertex_index i : vertices(surface_mesh_s)){
835   if(!surface_mesh_s.is_removed(i)){
836     points.push_back(surface_mesh_s.point(i));
837     vs.push_back(i);
838   }
839 }
840
841 search_map map(points);
842 Tree tree(boost::counting_iterator<std::size_t>(0),
843            boost::counting_iterator<std::size_t>(
844              vertices(surface_mesh_s).size()),
845            Tree::Splitter(), Traits(map));
846 K_neighbor_search::Distance tr_dist(map);
847
848 for(uint32_t i = 0; i < QP.numRows(); i++){
849   K_neighbor_search search(tree, Point(QP(i, 0), QP(i, 1), QP(i, 2)),
850                           1, 0, true, tr_dist);
851   I(i, 0) = vs[search.begin()>first];
852   D(i, 0) =
853     tr_dist.inverse_of_transformed_distance(search.begin()>second);
854 }
855 return std::make_pair(I, D);
856 }
857
858 // getSimplifiedShape() function with stop_ratio
859 // -----
860 std::pair<Matrix, Matrix>
861 AlphaShape3D::getSimplifiedShape(double stop_ratio) {
862 #ifdef DEBUG_ALPHA_SHAPE_3D
863   consoleLog(0, "Called getSimplifiedShape()");
864 #endif
865
866   Mesh surface_mesh_s(this->surface_mesh);
867   SMS::Edge_count_ratio_stop_predicate<Mesh> stop(stop_ratio);
868   uint32_t r = SMS::edgeCollapse(surface_mesh_s, stop);
869   PMP::remove_isolated_vertices(surface_mesh_s);
870   surface_mesh_s.collect_garbage();
871
872 #ifdef DEBUG_ALPHA_SHAPE_3D
873   std::cout << "Number of edges removed is " << r << std::endl
874             << "Number of final edges is "
875             << surface_mesh_s.number_of_edges() << std::endl;
876 #endif
877
878 Matrix Points(surface_mesh_s.number_of_vertices(), 3);

```

```

879 Matrix bf(surface_mesh_s.number_of_faces(), 3);
880
881 for (Mesh::Vertex_index vertex_index : surface_mesh_s.vertices()) {
882     Point p = surface_mesh_s.point(vertex_index);
883     Points(vertex_index.idx(), 0) = p[0];
884     Points(vertex_index.idx(), 1) = p[1];
885     Points(vertex_index.idx(), 2) = p[2];
886 }
887
888 for (Mesh::Face_index face_index : surface_mesh_s.faces()) {
889     CGAL::Vertex_around_face_circulator<Mesh>
890         vcirc(surface_mesh_s.halfedge(face_index), this->surface_mesh);
891     bf(face_index.idx(), 0) = *vcirc++;
892     bf(face_index.idx(), 1) = *vcirc++;
893     bf(face_index.idx(), 2) = *vcirc++;
894 }
895 return std::make_pair(Points, bf);
896 }
897
898 // getSimplifiedShape() function without parameters
899 // -----
900 std::pair<Matrix, Matrix>
901     AlphaShape3D::getSimplifiedShape() {
902     double stop_ratio = 0.05;
903     return this->getSimplifiedShape(stop_ratio);
904 }
905
906 // getSimplifiedShape() function with filename
907 // -----
908 std::pair<Matrix, Matrix>
909     AlphaShape3D::getSimplifiedShape(std::string filename) {
910     double stop_ratio = 0.05;
911     std::pair<Matrix, Matrix> ret = this->getSimplifiedShape(stop_ratio);
912     this->writeOff(filename, ret.first, ret.second);
913     return ret;
914 }
915
916 // getSimplifiedShape() function with stop_ratio and filename
917 // -----
918 std::pair<Matrix, Matrix>
919     AlphaShape3D::getSimplifiedShape(double stop_ratio,
920         std::string filename) {
921     std::pair<Matrix, Matrix> ret = this->getSimplifiedShape(stop_ratio);
922     this->writeOff(filename, ret.first, ret.second);
923     return ret;
924 }
925
926 // removeUnusedPoints() function
927 // -----
928 std::pair<Matrix, Matrix>
929     AlphaShape3D::removeUnusedPoints(Matrix Pi, Matrix bfi) {
930 #ifdef DEBUG_ALPHA_SHAPE_3D
931     consoleLog(0, "Called removeUnusedPoints()");
932 #endif
933 }
```

```

934     Mesh surface_mesh_s;
935     std::vector<CGAL_Polygon> polygons;
936     CGAL_Polygon p;
937     std::vector<Point> points;
938
939 #ifdef DEBUG_ALPHA_SHAPE_3D
940     std::cout << "Boundary surface reconstruction. " << std::endl;
941 #endif
942
943     uint32_t n = Pi.numRows();
944     uint32_t nbf = bfi.numRows();
945
946     for (std::size_t i = 0; i < n; i++) {
947         points.push_back(Point(Pi(i, 0), Pi(i, 1), Pi(i, 2)));
948     }
949
950     for (std::size_t i = 0; i < nbf; i++) {
951         p.clear();
952         for (uint8_t j = 0; j < 3; j++) {
953             p.push_back(bfi(i, j));
954         }
955         polygons.push_back(p);
956     }
957
958     PMP::orient_polygon_soup(points, polygons);
959     PMP::repair_polygon_soup(points, polygons);
960     PMP::polygon_soup_to_polygon_mesh(points,
961                                         polygons, surface_mesh_s);
962     surface_mesh_s.collect_garbage();
963
964     Matrix Points(surface_mesh_s.number_of_vertices(), 3);
965     Matrix bf(surface_mesh_s.number_of_faces(), 3);
966
967     for (Mesh::Vertex_index vertex_index : surface_mesh_s.vertices()) {
968         Point p = surface_mesh_s.point(vertex_index);
969         Points(vertex_index.idx(), 0) = p[0];
970         Points(vertex_index.idx(), 1) = p[1];
971         Points(vertex_index.idx(), 2) = p[2];
972     }
973
974     for (Mesh::Face_index face_index : surface_mesh_s.faces()) {
975         CGAL::Vertex_around_face_circulator<Mesh>
976             vcirc(surface_mesh_s.halfedge(face_index), this->surface_mesh);
977         bf(face_index.idx(), 0) = *vcirc++;
978         bf(face_index.idx(), 1) = *vcirc++;
979         bf(face_index.idx(), 2) = *vcirc++;
980     }
981     return std::make_pair(Points, bf);
982 }
983
984 // writeOff() function
985 // -----
986 void AlphaShape3D::writeOff(std::string filename, Matrix Points, Matrix bf) {
987 #ifdef DEBUG_ALPHA_SHAPE_3D
988     consoleLog(0, "Called writeOff()");

```

```

989 #endif
990
991     uint32_t n = Points.numRows();
992     uint32_t nbf = bf.numRows();
993
994     std::stringstream pts;
995     std::stringstream ind;
996
997     for (std::size_t i = 0; i < n; i++) {
998         pts << Points(i, 0) << " " << Points(i, 1) << " " << Points(i, 2) << std::endl;
999     }
1000
1001    for (std::size_t i = 0; i < nbf; i++) {
1002        ind << "3 " << (uint64_t)bf(i, 0) << " " << (uint64_t)bf(i, 1)
1003        << " " << (uint64_t)bf(i, 2) << std::endl;
1004    }
1005
1006    std::ofstream of(filename);
1007    CGAL::set_ascii_mode(of);
1008    of << "OFF" << std::endl << n << " " << nbf << " 0" << std::endl;
1009    of << pts.str();
1010    of << ind.str();
1011    of.close();
1012 }
1013
1014 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
1015     return AlphaShape3D::Init(env, exports);
1016 }
1017
1018 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
1019
1020 } // namespace alpha_shape_3d_ns

```

Listing 5 - alpha-shape-3d.cpp

```

1 // AlphaShape3D - alpha-shape-3d.h
2 // Author: Milos Petrasinovic <mpetrasinovic@prdc.rs>
3 // PR-DC, Republic of Serbia
4 // info@prdc.rs
5 // -----
6
7 #ifndef ALPHA_SHAPE_3D_H
8 #define ALPHA_SHAPE_3D_H
9
10 // #define DEBUG_ALPHA_SHAPE_3D
11 // #define DEBUG_ALPHA_SHAPE_3D_LEVEL 0
12 // #define PROFILE_ALPHA_SHAPE_3D
13
14 #include <napi.h>
15 #include <chrono>
16 #include <thread>
17 #include <Windows.h>
18 #include <ctime>
19 #include <iomanip>
20 #include <sstream>

```

```

21 #include <string>
22 #include <fstream>
23 #include <iostream>
24 #include <filesystem>
25 #include <cassert>
26 #include <list>
27 #include <vector>
28
29 #include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
30
31 #include <CGAL/Delaunay_triangulation_3.h>
32 #include <CGAL/Triangulation_vertex_base_with_info_3.h>
33 #include <CGAL/Alpha_shape_3.h>
34 #include <CGAL/Alpha_shape_cell_base_3.h>
35 #include <CGAL/Alpha_shape_vertex_base_3.h>
36
37 #include <CGAL/Surface_mesh.h>
38 #include <CGAL/Surface_mesh_simplification/edgeCollapse.h>
39 #include <CGAL/Surface_mesh_simplification/Policies/Edge_collapse/
   Edge_count_ratio_stop_predicate.h>
40
41 #include <CGAL/Polygon_mesh_processing/repair_polygon_soup.h>
42 #include <CGAL/Polygon_mesh_processing/orient_polygon_soup.h>
43 #include <CGAL/Polygon_mesh_processing/polygon_soup_to_polygon_mesh.h>
44 #include <CGAL/Polygon_mesh_processing/measure.h>
45 #include <CGAL/Polygon_mesh_processing/repair.h>
46
47 #include <CGAL/Search_traits_3.h>
48 #include <CGAL/Search_traits_adapter.h>
49 #include <CGAL/Orthogonal_k_neighbor_search.h>
50 #include <CGAL/boost/iterator/counting_iterator.hpp>
51
52 namespace alpha_shape_3d_ns {
53
54 using namespace std;
55 using namespace std::chrono;
56
57 namespace SMS = CGAL::Surface_mesh_simplification;
58 namespace PMP = CGAL::Polygon_mesh_processing;
59
60 typedef CGAL::Exact_predicates_inexact_constructions_kernel Gt;
61
62 typedef CGAL::Triangulation_vertex_base_with_info_3<unsigned, Gt> Tvb;
63 typedef CGAL::Alpha_shape_vertex_base_3<Gt, Tvb> Vb;
64 typedef CGAL::Alpha_shape_cell_base_3<Gt> Fb;
65 typedef CGAL::Triangulation_data_structure_3<Vb, Fb> Tds;
66 typedef CGAL::Delaunay_triangulation_3<Gt, Tds, CGAL::Fast_location> Dt;
67 typedef Dt::Point Point;
68
69 class search_map {
70     const std::vector<Point>& points;
71 public:
72     typedef Point value_type;
73     typedef const value_type& reference;
74     typedef std::size_t key_type;

```

```

75     typedef boost::lvalue_property_map_tag category;
76     search_map(const std::vector<Point>& pts) : points(pts){}
77     reference operator[](key_type k) const {return points[k];}
78     friend reference get(const search_map& ppmap, key_type i)
79     {return ppmap[i];}
80   };
81
82   typedef CGAL::Alpha_shape_3<Dt> As3;
83   typedef CGAL::Surface_mesh<Point> Mesh;
84   typedef std::vector<std::size_t> CGAL_Polygon;
85
86   typedef CGAL::Search_traits_3<Gt> Trb;
87   typedef CGAL::Search_traits_adapter<std::size_t, search_map, Trb> Traits;
88   typedef CGAL::Orthogonal_k_neighbor_search<Traits> K_neighbor_search;
89   typedef K_neighbor_search::Tree Tree;
90
91   template <typename T>
92   class CustomMatrix {
93     private:
94       std::vector<T> data;
95       uint32_t rows;
96       uint32_t cols;
97
98     public:
99       CustomMatrix() : rows(0), cols(0) {}
100      CustomMatrix(uint32_t r, uint32_t c) : rows(r), cols(c), data(r * c) {}
101
102      T& operator()(uint32_t i, uint32_t j) {
103        return data[i * cols + j];
104      }
105
106      const T& operator()(uint32_t i, uint32_t j) const {
107        return data[i * cols + j];
108      }
109
110      uint32_t numRows() const { return rows; }
111      uint32_t numCols() const { return cols; }
112
113      void resize(uint32_t r, uint32_t c) {
114        rows = r;
115        cols = c;
116        data.resize(r * c);
117      }
118    };
119    typedef CustomMatrix<double> Matrix;
120
121   class AlphaShape3D : public Napi::ObjectWrap<AlphaShape3D> {
122     public:
123       static Napi::Object Init(Napi::Env env, Napi::Object exports);
124       AlphaShape3D(const Napi::CallbackInfo& info);
125       ~AlphaShape3D();
126
127       Matrix inputPoints;
128       std::vector<Point> Points;
129       std::vector<std::pair<Point, unsigned>> Vertices;

```

```

130   double getAlpha(void);
131   void setAlpha(double);
132   double numRegions(void);
133   Matrix getAlphaSpectrum(void);
134   double getCriticalAlpha(std::string);
135   double getSurfaceArea(void);
136   double getVolume(void);
137   Matrix getBoundaryFacets(void);
138   Matrix getBoundaryFacets(std::string);
139   void writeBoundaryFacets(std::string);
140   Matrix checkInShape(Matrix);
141   Matrix getTriangulation(void);
142   std::pair<Matrix, Matrix> getNearestNeighbor(Matrix);
143   std::pair<Matrix, Matrix> getSimplifiedShape(double);
144   std::pair<Matrix, Matrix> getSimplifiedShape();
145   std::pair<Matrix, Matrix> getSimplifiedShape(std::string);
146   std::pair<Matrix, Matrix> getSimplifiedShape(double, std::string);
147   std::pair<Matrix, Matrix> removeUnusedPoints(Matrix, Matrix);
148   void writeOff(std::string, Matrix, Matrix);
149
150 // New JavaScript wrapper methods
151   void NewShapeJS(const Napi::CallbackInfo& info);
152   Napi::Value GetAlphaJS(const Napi::CallbackInfo& info);
153   void SetAlphaJS(const Napi::CallbackInfo& info);
154   Napi::Value GetNumRegionsJS(const Napi::CallbackInfo& info);
155   Napi::Value GetAlphaSpectrumJS(const Napi::CallbackInfo& info);
156   Napi::Value GetCriticalAlphaJS(const Napi::CallbackInfo& info);
157   Napi::Value GetSurfaceAreaJS(const Napi::CallbackInfo& info);
158   Napi::Value GetVolumeJS(const Napi::CallbackInfo& info);
159   Napi::Value GetBoundaryFacetsJS(const Napi::CallbackInfo& info);
160   void WriteBoundaryFacetsJS(const Napi::CallbackInfo& info);
161   Napi::Value CheckInShapeJS(const Napi::CallbackInfo& info);
162   void WriteOffJS(const Napi::CallbackInfo& info);
163   Napi::Value GetTriangulationJS(const Napi::CallbackInfo& info);
164   Napi::Value GetNearestNeighborJS(const Napi::CallbackInfo& info);
165   Napi::Value GetSimplifiedShapeJS(const Napi::CallbackInfo& info);
166   Napi::Value RemoveUnusedPointsJS(const Napi::CallbackInfo& info);
167
168 private:
169   As3 *alphaShape;
170   Dt *delaunayTriangulation;
171   Matrix triangulationMatrix;
172   std::size_t numAlphaValues;
173   Mesh surface_mesh;
174 };
175
176 } // namespace alpha_shape_3d_ns
177
178 #endif // ALPHA_SHAPE_3D_H

```

Listing 6 - alpha-shape-3d.h

```

1 [
2 {
3   "target_name": "native_module",
4   "sources": [

```

```

5      "cpp/native-module.cpp"
6  ],
7  "include_dirs": [
8    "<!(node -p \`require('node-addon-api').include\`)",
9    "<(module_root_dir)/lib/eigen-3.4.0/"
10 ],
11 "cflags!": [
12   "-fno-exceptions"
13 ],
14 "cflags_cc!": [
15   "-fno-exceptions"
16 ],
17 "defines": [
18   "NAPI_DISABLE_CPP_EXCEPTIONS"
19 ],
20 "msvs_settings": {
21   "VCCLCompilerTool": {
22     "AdditionalOptions": [
23       "-std:c++17"
24     ]
25   }
26 },
27 },
28 {
29   "target_name": "alpha_shape_3d",
30   "sources": [
31     "cpp/alpha-shape-3d.cpp"
32   ],
33   "include_dirs": [
34     "<!(node -p \`require('node-addon-api').include\`)",
35     "<(module_root_dir)/lib/cgal-6.0.1/include/",
36     "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/include",
37     "<(module_root_dir)/lib/boost-1.86.0/"
38   ],
39   "libraries": [
40     "<(module_root_dir)/lib/auxiliary/gmp/lib/gmp.lib",
41     "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/lib/mpfr.lib"
42   ],
43   "cflags!": [
44     "-fno-exceptions"
45   ],
46   "cflags_cc!": [
47     "-fno-exceptions",
48     "-O3",
49     "-DNDEBUG"
50   ],
51   "defines": [
52     "NAPI_DISABLE_CPP_EXCEPTIONS"
53   ],
54   "copies": [
55     {
56       "destination": "<(module_root_dir)/build/Release",
57       "files": [
58         "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/gmp-10.dll",
59         "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/mpfr-6.dll"

```

```

60      ]
61    }
62  ],
63  "msvs_settings": {
64    "VCCLCompilerTool": {
65      "AdditionalOptions": [
66        "-std:c++17",
67        "/GR",
68        "/EHsc"
69      ]
70    }
71  }
72 ]
73 ]

```

Listing 7 - binding.gyp

```

1 // JSLAB - native-module.cpp
2 // Author: Milos Petrasinovic <mpetrasinovic@prdc.rs>
3 // PR-DC, Republic of Serbia
4 // info@prdc.rs
5 // -----
6
7 #include "native-module.h"
8
9 namespace native_module_ns {
10
11 #ifdef PROFILE_NATIVE_MODULE
12 // Function to start the timer and return the start time
13 time_point<steady_clock> tic() {
14   return steady_clock::now();
15 }
16
17 // Function to stop the timer and return the elapsed time
18 long toc(const time_point<steady_clock>& startTime) {
19   return duration_cast<milliseconds>(steady_clock::now() - startTime).count()
20 }
21 #endif
22
23 // Function to get current time
24 std::string getCurrentTime() {
25   // get current time
26   auto now = system_clock::now();
27
28   // get number of milliseconds for the current second
29   // (remainder after division into seconds)
30   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
31
32   // convert to std::time_t in order to convert to std::tm (broken time)
33   auto timer = system_clock::to_time_t(now);
34
35   // convert to broken time
36   std::tm bt = *std::localtime(&timer);
37
38   std::ostringstream oss;

```

```

39
40     oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
41     oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
42
43     return oss.str();
44 }
45
46 // Function to console log data
47 int consoleLog(uint8_t level, const char* format, ...) {
48 #ifdef DEBUG_NATIVE_MODULE_LEVEL
49     if(level <= DEBUG_NATIVE_MODULE_LEVEL) {
50         printf("\033[0;33m%s NativeModule]\033[0m ", getCurrentTime().c_str());
51         va_list vl;
52         va_start(vl, format);
53         auto ret = vprintf(format, vl);
54         va_end(vl);
55         printf("\n");
56         return ret;
57     }
58 #endif
59     return 0;
60 }
61
62 // NativeModule()
63 // Object constructor
64 // -----
65 NativeModule::NativeModule(const Napi::CallbackInfo& info)
66     : Napi::ObjectWrap<NativeModule>(info) {
67 #ifdef DEBUG_NATIVE_MODULE
68     consoleLog(0, "Called constructor");
69 #endif
70 }
71
72 // ~NativeModule()
73 // Object destructor
74 // -----
75 NativeModule::~NativeModule() {
76 }
77
78 // Init() function
79 // -----
80 NativeModule::Init(Napi::Env env, Napi::Object exports) {
81     Napi::Function func =
82         DefineClass(env,
83             "NativeModule", {
84                 InstanceMethod("roots", &NativeModule::roots),
85                 InstanceMethod("cumtrapz", &NativeModule::cumtrapz),
86                 InstanceMethod("trapz", &NativeModule::trapz),
87                 InstanceMethod("listSubprocesses", &NativeModule::
88                     listSubprocesses),
89             });
90
91     Napi::FunctionReference* constructor = new Napi::FunctionReference();
92     *constructor = Napi::Persistent(func);

```

```

93   env.SetInstanceData(constructor);
94
95   exports.Set("NativeModule", func);
96   return exports;
97 }
98
99 // roots() function
100 // -----
101 Napi::Value NativeModule::roots(const Napi::CallbackInfo& info) {
102   Napi::Env env = info.Env();
103   const double TOLERANCE = 1e-10;
104
105   // Ensure the input is an array
106   if(!info[0].IsArray()) {
107     Napi::TypeError::New(env, "Expected an array of coefficients").
108       ThrowAsJavaScriptException();
109     return Napi::Array::New(env);
110   }
111
112   // Extract the polynomial coefficients from the input
113   Napi::Array coefficientsArray = info[0].As<Napi::Array>();
114   int degree = coefficientsArray.Length() - 1;
115
116   // Create an Eigen vector for the coefficients
117   VectorXd coefficients(degree + 1);
118   for(int i = 0; i <= degree; ++i) {
119     coefficients(i) = coefficientsArray.Get(i).As<Napi::Number>().DoubleValue();
120   }
121
122   // Create the companion matrix
123   MatrixXd companionMatrix = MatrixXd::Zero(degree, degree);
124   for(int i = 1; i < degree; ++i) {
125     companionMatrix(i, i - 1) = 1.0;
126   }
127   for(int i = 0; i < degree; ++i) {
128     companionMatrix(i, degree - 1) = -coefficients(degree - i) / coefficients(0);
129   }
130
131   // Use Eigen's eigenvalue solver to find the roots
132   EigenSolver<MatrixXd> solver(companionMatrix);
133   VectorXcd roots = solver.eigenvalues();
134
135   // Convert the result to a JavaScript array
136   Napi::Array result = Napi::Array::New(env, degree);
137   for (int i = 0; i < degree; ++i) {
138     double realPart = roots(i).real();
139     double imagPart = roots(i).imag();
140
141     if(abs(imagPart) < TOLERANCE || isnan(imagPart)) {
142       // If the imaginary part is close to zero, return as a real number
143       result[i] = Napi::Number::New(env, realPart);
144     } else {
145       // Otherwise, return as a complex number object

```

```
145     Napi::Object complexRoot = Napi::Object::New(env);
146     complexRoot.Set("real", Napi::Number::New(env, realPart));
147     complexRoot.Set("imag", Napi::Number::New(env, imagPart));
148     result[i] = complexRoot;
149 }
150 }
151
152 return result;
153 }
154
155 // cumtrapz() function
156 // -----
157 Napi::Value NativeModule::cumtrapz(const Napi::CallbackInfo& info) {
158     Napi::Env env = info.Env();
159
160     // Ensure at least one argument is provided
161     if(info.Length() < 1) {
162         Napi::TypeError::New(env, "cumtrapz expects at least one argument").
163             ThrowAsJavaScriptException();
164         return env.Null();
165     }
166
167     // Ensure the first argument is an array
168     if(!info[0].IsArray()) {
169         Napi::TypeError::New(env, "First argument must be an array").
170             ThrowAsJavaScriptException();
171         return env.Null();
172     }
173
174     Napi::Array yInput = info[0].As<Napi::Array>();
175     Napi::Array xInput;
176     bool hasX = info.Length() > 1;
177
178     // If x is provided, ensure it's an array
179     if(hasX) {
180         if(!info[1].IsArray()) {
181             Napi::TypeError::New(env, "Second argument must be an array").
182                 ThrowAsJavaScriptException();
183             return env.Null();
184         }
185         xInput = info[1].As<Napi::Array>();
186     }
187
188     // Get the length of yInput
189     uint32_t n = yInput.Length();
190
191     // If x is provided, its length must match yInput
192     if(hasX && xInput.Length() != n) {
193         Napi::RangeError::New(env, "x and y arrays must have the same length").
194             ThrowAsJavaScriptException();
195         return env.Null();
196     }
197
198     // Handle empty array
199     if(n == 0) {
```

```

196     return Napi::Array::New(env, 0);
197 }
198
199 // Initialize Eigen vectors
200 Eigen::VectorXd y(n);
201 Eigen::VectorXd x(n);
202 Eigen::VectorXd result(n);
203
204 // Load y values from JavaScript array
205 for(uint32_t i = 0; i < n; ++i) {
206   Napi::Value val = yInput.Get(i);
207   if(!val.IsNumber()) {
208     Napi::TypeError::New(env, "y array must contain only numbers").
209       ThrowAsJavaScriptException();
210   }
211   y[i] = val.As<Napi::Number>().DoubleValue();
212 }
213
214 // If x is provided, load x values; otherwise, assume uniform spacing
215 if(hasX) {
216   for(uint32_t i = 0; i < n; ++i) {
217     Napi::Value val = xInput.Get(i);
218     if (!val.IsNumber()) {
219       Napi::TypeError::New(env, "x array must contain only numbers").
220         ThrowAsJavaScriptException();
221     }
222     x[i] = val.As<Napi::Number>().DoubleValue();
223   }
224 } else {
225   // Uniform spacing: x = [0, 1, 2, ..., n-1]
226   for(uint32_t i = 0; i < n; ++i) {
227     x[i] = static_cast<double>(i);
228   }
229 }
230
231 // Initialize result: first value is always 0
232 result[0] = 0.0;
233
234 // Cumulative trapezoidal integration
235 for(uint32_t i = 1; i < n; ++i) {
236   double dx = x[i] - x[i - 1]; // Difference in x
237   double dy = 0.5 * (y[i] + y[i - 1]); // Average height (trapezoid rule)
238   result[i] = result[i - 1] + dx * dy;
239 }
240
241 // Convert Eigen vector back to JavaScript array
242 Napi::Array jsResult = Napi::Array::New(env, n);
243 for(uint32_t i = 0; i < n; ++i) {
244   jsResult.Set(i, Napi::Number::New(env, result[i]));
245 }
246
247 return jsResult;
248 }
```

```

249
250 // trapz() function
251 // -----
252 Napi::Value NativeModule::trapz(const Napi::CallbackInfo& info) {
253   Napi::Env env = info.Env();
254
255   // Ensure at least one argument is provided
256   if(info.Length() < 1) {
257     Napi::TypeError::New(env, "trapz expects at least one argument").
258       ThrowAsJavaScriptException();
259     return env.Null();
260   }
261
262   // Ensure the first argument is an array
263   if(!info[0].IsArray()) {
264     Napi::TypeError::New(env, "First argument must be an array").
265       ThrowAsJavaScriptException();
266     return env.Null();
267   }
268
269   Napi::Array yInput = info[0].As<Napi::Array>();
270   Napi::Array xInput;
271   bool hasX = info.Length() > 1;
272
273   // If x is provided, ensure it's an array
274   if(hasX) {
275     if(!info[1].IsArray()) {
276       Napi::TypeError::New(env, "Second argument must be an array").
277         ThrowAsJavaScriptException();
278     }
279     xInput = info[1].As<Napi::Array>();
280   }
281
282   // Get the length of yInput
283   uint32_t n = yInput.Length();
284
285   // If x is provided, its length must match yInput
286   if(hasX && xInput.Length() != n) {
287     Napi::RangeError::New(env, "x and y arrays must have the same length").
288       ThrowAsJavaScriptException();
289     return env.Null();
290   }
291
292   // Handle cases with fewer than 2 points
293   if(n < 2) {
294     Napi::RangeError::New(env, "trapz requires at least two data points").
295       ThrowAsJavaScriptException();
296     return env.Null();
297   }
298
299   // Initialize Eigen vectors
300   Eigen::VectorXd y(n);
301   Eigen::VectorXd x(n);

```

```

299 // Load y values from JavaScript array
300 for(uint32_t i = 0; i < n; ++i) {
301   Napi::Value val = yInput.Get(i);
302   if (!val.IsNumber()) {
303     Napi::TypeError::New(env, "y array must contain only numbers").
304       ThrowAsJavaScriptException();
305   }
306   y[i] = val.As<Napi::Number>().DoubleValue();
307 }
308
309 // If x is provided, load x values; otherwise, assume uniform spacing
310 if(hasX) {
311   for(uint32_t i = 0; i < n; ++i) {
312     Napi::Value val = xInput.Get(i);
313     if (!val.IsNumber()) {
314       Napi::TypeError::New(env, "x array must contain only numbers").
315         ThrowAsJavaScriptException();
316     }
317     x[i] = val.As<Napi::Number>().DoubleValue();
318   }
319 } else {
320   // Uniform spacing: x = [0, 1, 2, ..., n-1]
321   for(uint32_t i = 0; i < n; ++i) {
322     x[i] = static_cast<double>(i);
323   }
324 }
325
326 // Perform trapezoidal integration
327 double total = 0.0;
328 for(uint32_t i = 1; i < n; ++i) {
329   double dx = x[i] - x[i - 1];
330   double dy = 0.5 * (y[i] + y[i - 1]);
331   total += dx * dy;
332 }
333
334 return Napi::Number::New(env, total);
335 }
336
337 // listSubprocesses() function
338 // -----
339 Napi::Value NativeModule::listSubprocesses(const Napi::CallbackInfo& info) {
340   Napi::Env env = info.Env();
341
342   // Ensure at least one argument is provided
343   if(info.Length() < 1) {
344     Napi::TypeError::New(env, "listSubprocesses expects at least one argument").
345       ThrowAsJavaScriptException();
346   }
347
348   // Ensure the first argument is an number
349   if (!info[0].IsNumber()) {
350     Napi::TypeError::New(env, "First argument must be an number").

```

```

      ThrowAsJavaScriptException() ;
351   return env.Null();
352 }
353
354 uint32_t parent_pid = info[0].As<Napi::Number>().Uint32Value();
355 uint32_t i = 0;
356 Napi::Array jsResult = Napi::Array::New(env);
357
358 // Create a snapshot of all processes in the system.
359 HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
360 if(hSnapshot == INVALID_HANDLE_VALUE) {
361   Napi::Error::New(env, "Unable to create process snapshot").
      ThrowAsJavaScriptException();
362   return jsResult;
363 }
364
365 PROCESSENTRY32 pe;
366 pe.dwSize = sizeof(PROCESSENTRY32);
367
368 // Retrieve the first process.
369 if(Process32First(hSnapshot, &pe)) {
370   do {
371     if(pe.th32ParentProcessID == parent_pid) {
372       jsResult.Set(i, Napi::Number::New(env, pe.th32ProcessID));
373       i = i+1;
374     }
375   } while(Process32Next(hSnapshot, &pe));
376 } else {
377   Napi::Error::New(env, "Unable to retrieve process information").
      ThrowAsJavaScriptException();
378 }
379
380 CloseHandle(hSnapshot);
381 return jsResult;
382 }
383
384 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
385   return NativeModule::Init(env, exports);
386 }
387
388 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
389 }
390 } // namespace native_module_ns

```

Listing 8 - native-module.cpp

```

1 // JSLAB - native-module.h
2 // Author: Milos Petrasinovic <mpetrasinovic@prdc.rs>
3 // PR-DC, Republic of Serbia
4 // info@prdc.rs
5 // -----
6
7 #ifndef NATIVE_MODULE_H
8 #define NATIVE_MODULE_H
9
10 // #define DEBUG_NATIVE_MODULE

```

```

11 // #define DEBUG_NATIVE_MODULE_LEVEL 0
12 // #define PROFILE_NATIVE_MODULE
13
14 #include <napi.h>
15 #include <chrono>
16 #include <thread>
17 #include <Windows.h>
18 #include <tlhelp32.h>
19 #include <ctime>
20 #include <iomanip>
21 #include <sstream>
22 #include <string>
23 #include <fstream>
24 #include <iostream>
25 #include <filesystem>
26 #include <vector>
27 #include <complex>
28 #include <Eigen/Dense>
29
30 namespace native_module_ns {
31
32 using namespace std;
33 using namespace std::chrono;
34 using namespace Eigen;
35
36 class NativeModule : public Napi::ObjectWrap<NativeModule> {
37 public:
38   static Napi::Object Init(Napi::Env env, Napi::Object exports);
39   NativeModule(const Napi::CallbackInfo& info);
40   ~NativeModule();
41
42   Napi::Value roots(const Napi::CallbackInfo& info);
43   Napi::Value cumtrapz(const Napi::CallbackInfo& info);
44   Napi::Value trapz(const Napi::CallbackInfo& info);
45   Napi::Value listSubprocesses(const Napi::CallbackInfo& info);
46 };
47
48 } // namespace native_module_ns
49
50 #endif // NATIVE_MODULE_H

```

Listing 9 - native-module.h

4 CSS

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
  abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
  strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
  label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas,
  details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby,
  section, summary, time, mark, audio, video { border: 0; font-size: 100%;
  font: inherit; vertical-align: baseline; margin: 0; padding: 0; }
  article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section {
  display: block; }
  body { line-height: 1; }
  ol, ul { list-style: none; }
  blockquote, q {

```

```

1 quotes:none}blockquote:before ,blockquote:after ,q:before ,q:after {
2   content:none}table{border-collapse:collapse; border-spacing:0}
3
4 html {
5   overflow: hidden;
6 }
7 :focus {
8   outline: 0;
9 }
10
11 div {
12   z-index: 2;
13   box-sizing: border-box;
14 }
15
16 section {
17   position: relative;
18   z-index: 2;
19 }
20
21 body {
22   font-family: 'Roboto', Arial;
23   width: 100%;
24   background: #fff ;
25 }
26
27 .clear {
28   clear: both;
29 }
30
31 .float-left {
32   float: left;
33 }
34
35 .float-right {
36   float: right;
37 }
```

Listing 10 - basic.css

```

1 .json-node-root a {
2   font-weight: 400! important ;
3 }
4
5 .json-node-type {
6   color: #7f7f7f ;
7 }
8
9 .json-node-toggler ,
10 .json-node-stub-toggler {
11   text-decoration: none;
12   color: inherit ;
13 }
14
15 .json-node-children ,
```

```
16 .json-node-root {  
17   padding-left: 1em;  
18 }  
19  
20 .json-node-header mark {  
21   background-color: rgba(199, 193, 0, 0.5);  
22   padding: 0;  
23 }  
24  
25 .json-node-header mark.highlight-active {  
26   background-color: rgba(199, 103, 46, 0.5);  
27 }  
28  
29 .json-node-label {  
30   color: #111;  
31 }  
32  
33 .json-node-number .json-node-value {  
34   color: #ff8000;  
35 }  
36  
37 .json-node-string .json-node-value {  
38   color: #808080;  
39 }  
40  
41 .json-node-boolean .json-node-value {  
42   color: #0000ff;  
43 }  
44  
45 .json-node-null .json-node-value {  
46   color: #ff8000;  
47 }  
48  
49 .json-node-number .json-node-type,  
50 .json-node-string .json-node-type,  
51 .json-node-boolean .json-node-type,  
52 .json-node-undefined .json-node-type,  
53 .json-node-null .json-node-type {  
54   display: none;  
55 }  
56  
57 .json-node-accessor {  
58   position: relative;  
59 }  
60  
61 .json-node-accessor::before {  
62   position: absolute;  
63   content: ' ';  
64   font-size: 0.7em;  
65   line-height: 1.6em;  
66   right: 0.5em;  
67   top: 0.1em;  
68   transition: transform 100ms ease-out;  
69   color: #333;  
70   opacity: 0.7;
```

```

71  }
72
73 .json-node-open .json-node-accessor::before {
74   transform: rotate(90deg);
75 }
76
77 .json-node-stub-toggler .json-node-label ,
78 .json-node-collapse {
79   color: #7f7f7f;
80 }
81 .json-node-collapse {
82   font-size: 0.8em;
83 }
84
85 @keyframes json-node-children-open {
86   from {
87     transform: scaleY(0);
88   }
89   to {
90     transform: scaleY(1);
91   }
92 }
93
94 .json-node-link {
95   display: none;
96   padding-left: 0.5em;
97   font-size: 0.8em;
98   color: #7f7f7f;
99   text-decoration: none;
100 }
```

Listing 11 - big-json-viewer-notepadpp-theme.css

```

1 #code {
2   height: calc(100vh - 74px);
3 }
4
5 .CodeMirror {
6   height: calc(100% - 6px);
7   margin: 5px;
8   border: 1px solid #f7df1e;
9   border-radius: 3px;
10  font-size: 16px;
11  margin-top: 1px;
12 }
13
14 .CodeMirror div {
15   z-index: auto;
16 }
17
18 .CodeMirror-gutter-wrapper, .CodeMirror-cursor {
19   z-index: 4!important;
20 }
21
22 .CodeMirror-vscrollbar::-webkit-scrollbar,
23 .CodeMirror-vscrollbar::-webkit-scrollbar-button,
```

```
24 .CodeMirror-vscrollbar::-webkit-scrollbar-track ,  
25 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece ,  
26 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb ,  
27 .CodeMirror-vscrollbar::-webkit-scrollbar-corner ,  
28 .CodeMirror-vscrollbar::-webkit-resizer {  
    background: transparent ;  
}  
31  
32 .CodeMirror-vscrollbar {  
33  
34 }  
35  
36 .CodeMirror-vscrollbar::-webkit-scrollbar {  
    width: 12px ;  
    height: 12px ;  
    -webkit-border-radius: 5px ;  
    border-radius: 5px ;  
}  
41  
42 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece {  
    -webkit-border-radius: 5px ;  
    border-radius: 5px ;  
}  
46  
47 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb {  
    background: #ddd ;  
    border-radius: 5px ;  
    background-clip: content-box ;  
    border: 2px solid transparent ;  
}  
53  
54 .CodeMirror-vscrollbar::-webkit-scrollbar-button {  
    width:0 ;  
    height:0 ;  
}  
58  
59  
60 .CodeMirror-ruler {  
    z-index: 1 !important ;  
}  
62  
63  
64 .CodeMirror-foldmarker {  
    color: #000 ;  
    text-shadow: none ;  
}  
67  
68  
69 .CodeMirror-lint-tooltip {  
    background-color: #fff ;  
    border: 1px solid #f7df1e ;  
    border-radius: 3px ;  
    color: #666 ;  
    font-family: Roboto ;  
    line-height: 16px ;  
}  
76  
77  
78 .CodeMirror-lint-tooltip {
```

```

79  background-color: #fff ;
80  border: 1px solid #f7df1e ;
81  border-radius: 3px ;
82  color: #666 ;
83  font-family: Roboto ;
84  line-height: 16px ;
85 }
86
87 .CodeMirror-hint {
88   line-height: 16px ;
89 }
90
91 .CodeMirror-focused .cm-matchhighlight {
92   background-color: #0f0 ;
93 }
94
95 .CodeMirror-search-match {
96   background: gold ;
97   border-top: 1px solid orange ;
98   border-bottom: 1px solid orange ;
99   -moz-box-sizing: border-box ;
100  box-sizing: border-box ;
101  opacity: .5 ;
102 }
103
104 .CodeMirror-selection-highlight-scrollbar {
105  background-color: #0f0 ;
106  width: 13px !important ;
107  z-index: 100 !important ;
108 }
```

Listing 12 - codemirror-custom.css

```

1 #code {
2   height: calc(100vh - 74px) ;
3 }
4
5 .CodeMirror {
6   height: calc(100% - 6px) ;
7   margin: 5px ;
8   border: 1px solid #f7df1e ;
9   border-radius: 3px ;
10  font-size: 16px ;
11  margin-top: 1px ;
12 }
13
14 .CodeMirror div {
15   z-index: auto ;
16 }
17
18 .CodeMirror-gutter-wrapper , .CodeMirror-cursor {
19   z-index: 4 !important ;
20 }
21
22 .CodeMirror-vscrollbar::-webkit-scrollbar ,
23 .CodeMirror-vscrollbar::-webkit-scrollbar-button ,
```

```
24 .CodeMirror-vscrollbar::-webkit-scrollbar-track ,  
25 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece ,  
26 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb ,  
27 .CodeMirror-vscrollbar::-webkit-scrollbar-corner ,  
28 .CodeMirror-vscrollbar::-webkit-resizer {  
    background: transparent ;  
}  
31  
32 .CodeMirror-vscrollbar {  
    z-index: 3 !important ;  
}  
35  
36 .CodeMirror-vscrollbar::-webkit-scrollbar {  
    width: 12px ;  
    height: 12px ;  
    -webkit-border-radius: 5px ;  
    border-radius: 5px ;  
}  
41  
42 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece {  
    -webkit-border-radius: 5px ;  
    border-radius: 5px ;  
}  
46  
47 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb {  
    background: #ddd ;  
    border-radius: 5px ;  
    background-clip: content-box ;  
    border: 2px solid transparent ;  
}  
53  
54  
55 .CodeMirror-vscrollbar::-webkit-scrollbar-button {  
    width: 0 ;  
    height: 0 ;  
}  
58  
59  
60 .CodeMirror-ruler {  
    z-index: 1 !important ;  
}  
62  
63  
64 .CodeMirror-foldmarker {  
    color: #000 ;  
    text-shadow: none ;  
}  
67  
68  
69 .CodeMirror-lint-tooltip {  
    background-color: #fff ;  
    border: 1px solid #f7df1e ;  
    border-radius: 3px ;  
    color: #666 ;  
    font-family: Roboto ;  
    line-height: 16px ;  
}  
76  
77  
78 .CodeMirror-lint-tooltip {
```

```
79 background-color: #fff ;
80 border: 1px solid #f7df1e ;
81 border-radius: 3px;
82 color: #666;
83 font-family: Roboto;
84 line-height: 16px;
85 }
86
87 .CodeMirror-hint {
88 line-height: 16px;
89 }
90
91 .CodeMirror-focused .cm-matchhighlight {
92 background-color: #0f0 ;
93 }
94
95 .CodeMirror-search-match {
96 background: gold ;
97 border-top: 1px solid orange;
98 border-bottom: 1px solid orange;
99 -moz-box-sizing: border-box;
100 box-sizing: border-box;
101 opacity: .5;
102 }
103
104 .CodeMirror-selection-highlight-scrollbar {
105 background-color: #0f0 ;
106 width: 13px!important;
107 z-index: 100!important;
108 }
109
110 .CodeMirror-selected , .CodeMirror-focused .CodeMirror-selected {
111 background-color: #c0c0c0 !important;
112 }
113
114 .CodeMirror-hints::-webkit-scrollbar ,
115 .CodeMirror-hints::-webkit-scrollbar-button ,
116 .CodeMirror-hints::-webkit-scrollbar-track ,
117 .CodeMirror-hints::-webkit-scrollbar-track-piece ,
118 .CodeMirror-hints::-webkit-scrollbar-thumb ,
119 .CodeMirror-hints::-webkit-scrollbar-corner ,
120 .CodeMirror-hints::-webkit-resizer {
121 background: transparent ;
122 }
123
124 .CodeMirror-hints::-webkit-scrollbar {
125 width: 12px;
126 height: 12px;
127 -webkit-border-radius: 5px ;
128 border-radius: 5px;
129 }
130
131 .CodeMirror-hints::-webkit-scrollbar-track-piece {
132 -webkit-border-radius: 5px ;
133 border-radius: 5px;
```

```
134    }
135
136  .CodeMirror-hints::-webkit-scrollbar-thumb {
137    background: #ddd;
138    border-radius: 5px;
139    background-clip: content-box;
140    border: 2px solid transparent;
141  }
142
143  .CodeMirror-hints::-webkit-scrollbar-button {
144    width:0;
145    height:0;
146  }
147
148  .CodeMirror-search-dialog {
149    font-family: Roboto;
150    font-size: 16px;
151    display: none;
152    position: absolute;
153    top: 0;
154    right: 15px;
155    background-color: rgba(255, 255, 255, 0.9);
156    padding: 10px;
157    border-radius: 0 0 5px 5px;
158    box-shadow: 0 0 10px #00000026;
159    z-index: 3 !important;
160  }
161
162  .CodeMirror-search-find, .CodeMirror-search-replace {
163    margin: 0;
164    border: 1px solid #ddd;
165    padding: 3px 5px;
166    display: block;
167    font-size: 16px;
168    border-radius: 3px;
169    margin-right: 10px;
170    float: left;
171    margin-bottom: 10px;
172    font-weight: 100;
173  }
174
175  .CodeMirror-search-find::placeholder, .CodeMirror-search-replace::placeholder {
176    font-weight: 100;
177    color: #999;
178  }
179
180  .CodeMirror-search-find:focus, .CodeMirror-search-replace:focus {
181    border: 1px solid #f7df1e;
182  }
183
184  .CodeMirror-search-find-prev-btn, .CodeMirror-search-find-next-btn,
185  .CodeMirror-search-replace-btn, .CodeMirror-search-replace-all-btn {
186    cursor: pointer;
187    background-size: 20px;
```



```
188 height: 20px;
189 width: 20px;
190 opacity: 0.5;
191 display: block;
192 float: left;
193 margin-top: 3px;
194 }
195
196 .CodeMirror-search-find-prev-btn:hover, .CodeMirror-search-find-next-btn:hover
197 , .CodeMirror-search-replace-btn:hover, .CodeMirror-search-replace-all-btn:hover
198 {
199   opacity: 1;
200 }
201
202 .CodeMirror-search-find-prev-btn {
203   background-image: url(../img/arrow.svg);
204   transform: rotate(90deg);
205   margin-right: 10px;
206 }
207
208 .CodeMirror-search-find-next-btn {
209   background-image: url(../img/arrow.svg);
210   transform: rotate(-90deg);
211 }
212
213 .CodeMirror-search-replace-btn {
214   background-image: url(../img/replace.svg);
215   margin-right: 10px;
216 }
217
218 .CodeMirror-search-replace-all-btn {
219   background-image: url(../img/replace-all.svg);
220 }
221
222 .CodeMirror-search-case-btn, .CodeMirror-search-regex-btn {
223   cursor: pointer;
224   background-size: 20px;
225   height: 20px;
226   width: 20px;
227   opacity: 0.5;
228   display: block;
229   float: left;
230 }
231
232 .CodeMirror-search-case-btn:hover, .CodeMirror-search-regex-btn:hover {
233   opacity: 0.7;
234 }
235
236 .CodeMirror-search-case-btn.active, .CodeMirror-search-regex-btn.active {
237   opacity: 1;
238 }
239
240 .CodeMirror-search-case-btn {
241   margin-left: 5px;
```

```

241   margin-right: 10px;
242   background-image: url(../img/match-case.svg);
243 }
244
245 .CodeMirror-search-regex-btn {
246   background-image: url(../img/regex.svg);
247 }
248
249 .CodeMirror-search-bottom-right {
250   float: right;
251   margin-top: 5px;
252   font-weight: 100;
253   color: #999;
254 }
255
256 .CodeMirror-search-close-btn {
257   cursor: pointer;
258   background-size: 15px;
259   height: 15px;
260   width: 15px;
261   opacity: 0.4;
262   display: block;
263   float: right;
264   background-image: url(../img/close.svg);
265   margin-left: 10px;
266 }
267
268 .CodeMirror-search-close-btn:hover {
269   opacity: 1;
270 }
```

Listing 13 - codemirror-editor-custom.css

```

1 .CodeMirror {
2   height: auto;
3   font-size: 16px;
4   font-family: 'Roboto', Arial;
5 }
6
7 .CodeMirror div {
8   z-index: auto;
9 }
10
11 .CodeMirror-gutter-wrapper, .CodeMirror-cursor {
12   z-index: 4!important;
13 }
14
15 .CodeMirror-lint-tooltip {
16   background-color: #fff;
17   border: 1px solid #f7df1e;
18   border-radius: 3px;
19   color: #666;
20   font-family: Roboto;
21   line-height: 16px;
22 }
23
```



```
24 .CodeMirror-lint-tooltip {  
25   background-color: #fff;  
26   border: 1px solid #f7df1e;  
27   border-radius: 3px;  
28   color: #666;  
29   font-family: 'Roboto', Arial;  
30   line-height: 16px;  
31 }  
32  
33 .CodeMirror-hint {  
34   font-family: 'Roboto', Arial;  
35   line-height: 16px;  
36 }  
37  
38 .CodeMirror-focused .cm-matchhighlight {  
39   background-color: #0f0;  
40 }  
41  
42 .CodeMirror-search-match {  
43   background: gold;  
44   border-top: 1px solid orange;  
45   border-bottom: 1px solid orange;  
46   -moz-box-sizing: border-box;  
47   box-sizing: border-box;  
48   opacity: .5;  
49 }  
50  
51 .CodeMirror-gutters {  
52   border-right: none;  
53   background-color: #fff;  
54 }  
55  
56 .CodeMirror-selected, .CodeMirror-focused .CodeMirror-selected {  
57   background-color: #c0c0c0 !important;  
58 }  
59  
60 .CodeMirror-hints::-webkit-scrollbar,  
61 .CodeMirror-hints::-webkit-scrollbar-button,  
62 .CodeMirror-hints::-webkit-scrollbar-track,  
63 .CodeMirror-hints::-webkit-scrollbar-track-piece,  
64 .CodeMirror-hints::-webkit-scrollbar-thumb,  
65 .CodeMirror-hints::-webkit-scrollbar-corner,  
66 .CodeMirror-hints::-webkit-resizer {  
67   background: transparent;  
68 }  
69  
70 .CodeMirror-hints::-webkit-scrollbar {  
71   width: 12px;  
72   height: 12px;  
73   -webkit-border-radius: 5px;  
74   border-radius: 5px;  
75 }  
76  
77 .CodeMirror-hints::-webkit-scrollbar-track-piece {  
78   -webkit-border-radius: 5px;
```

```
79 border-radius: 5px;  
80 }  
81  
82 .CodeMirror-hints::-webkit-scrollbar-thumb {  
83 background: #ddd;  
84 border-radius: 5px;  
85 background-clip: content-box;  
86 border: 2px solid transparent;  
87 }  
88  
89 .CodeMirror-hints::-webkit-scrollbar-button {  
90 width:0;  
91 height:0;  
92 }
```

Listing 14 - codemirror-main-custom.css

```
1 .cm-s-notepadapp span.cm-comment {  
2   color: #008000;  
3 }  
4 .cm-s-notepadapp span.cm-keyword {  
5   line-height: 1em;  
6   font-weight: bold;  
7   color: #0000ff ;  
8 }  
9 .cm-s-notepadapp span.cm-string {  
10  color: #808080;  
11 }  
12 .cm-s-notepadapp span.cm-string-2 {  
13  color: #060683;  
14 }  
15  
16 .cm-s-notepadapp span.cm-builtin {  
17  line-height: 1em;  
18  font-weight: bold ;  
19  color: #804000;  
20 }  
21 .cm-s-notepadapp span.cm-special {  
22  line-height: 1em;  
23  font-weight: bold ;  
24  color: #0aa ;  
25 }  
26 .cm-s-notepadapp span.cm-variable {  
27  color: black ;  
28 }  
29 .cm-s-notepadapp span.cm-number {  
30  color: #ff8000 ;  
31 }  
32  
33 .cm-s-notepadapp span.cm-atom {  
34  color: #0000ff ;  
35 }  
36 .cm-s-notepadapp span.cm-meta {  
37  color: #814203;  
38 }  
39
```

```

40 .cm-s-notepadapp span.cm-type {
41   color: #8206ff;
42 }
43
44 .cm-s-notepadapp span.cm-link {
45   color: #3a3;
46 }
47
48 .cm-s-notepadapp .CodeMirror-activeline-background {
49   background: #e8e8ff;
50 }
51 .cm-s-notepadapp .CodeMirror-matchingbracket {
52   outline: 1px solid grey;
53   color: black !important;
54 }
55
56 .cm-s-notepadapp span.cm-tag, .cm-s-notepadapp span.cm-def {
57   color: #0000ff;
58 }
59
60 .cm-s-notepadapp span.cm-attribute, .cm-s-notepadapp span.cm-qualifier {
61   color: #ff2324ff;
62 }
```

Listing 15 - codemirror-notepadpp-theme.css

```

1 #code {
2   height: calc(100vh - 83px);
3 }
4
5 .CodeMirror {
6   height: calc(100% - 6px);
7   margin: 5px;
8   border: 1px solid #f7df1e;
9   border-radius: 3px;
10  font-size: 16px;
11  margin-top: 1px;
12 }
13
14 .CodeMirror div {
15   z-index: auto;
16 }
17
18 .CodeMirror-gutter-wrapper, .CodeMirror-cursor {
19   z-index: 4!important;
20 }
21
22 .CodeMirror-vscrollbar::-webkit-scrollbar,
23 .CodeMirror-vscrollbar::-webkit-scrollbar-button,
24 .CodeMirror-vscrollbar::-webkit-scrollbar-track,
25 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece,
26 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb,
27 .CodeMirror-vscrollbar::-webkit-scrollbar-corner,
28 .CodeMirror-vscrollbar::-webkit-scrollbar-resizer {
29   background: transparent;
30 }
```



```
31
32 .CodeMirror-vscrollbar {
33   z-index: 3!important;
34 }
35
36 .CodeMirror-vscrollbar::-webkit-scrollbar {
37   width: 12px;
38   height: 12px;
39   -webkit-border-radius: 5px;
40   border-radius: 5px;
41 }
42
43 .CodeMirror-vscrollbar::-webkit-scrollbar-track-piece {
44   -webkit-border-radius: 5px;
45   border-radius: 5px;
46 }
47
48 .CodeMirror-vscrollbar::-webkit-scrollbar-thumb {
49   background: #ddd;
50   border-radius: 5px;
51   background-clip: content-box;
52   border: 2px solid transparent;
53 }
54
55 .CodeMirror-vscrollbar::-webkit-scrollbar-button {
56   width:0;
57   height:0;
58 }
59
60 .CodeMirror-ruler {
61   z-index: 1!important;
62 }
63
64 .CodeMirror-foldmarker {
65   color: #000;
66   text-shadow: none;
67 }
68
69 .CodeMirror-lint-tooltip {
70   background-color: #fff;
71   border: 1px solid #f7df1e;
72   border-radius: 3px;
73   color: #666;
74   font-family: Roboto;
75   line-height: 16px;
76 }
77
78 .CodeMirror-lint-tooltip {
79   background-color: #fff;
80   border: 1px solid #f7df1e;
81   border-radius: 3px;
82   color: #666;
83   font-family: Roboto;
84   line-height: 16px;
85 }
```

```
86
87 .CodeMirror-hint {
88   line-height: 16px;
89 }
90
91 .CodeMirror-focused .cm-matchhighlight {
92   background-color: #0f0 ;
93 }
94
95 .CodeMirror-search-match {
96   background: gold ;
97   border-top: 1px solid orange;
98   border-bottom: 1px solid orange;
99   -moz-box-sizing: border-box;
100  box-sizing: border-box;
101  opacity: .5;
102 }
103
104 .CodeMirror-selection-highlight-scrollbar {
105   background-color: #0f0 ;
106   width: 13px!important;
107   z-index: 100!important;
108 }
109
110 .CodeMirror-selected , .CodeMirror-focused .CodeMirror-selected {
111   background-color: #c0c0c0 !important;
112 }
113
114 .CodeMirror-hints::-webkit-scrollbar ,
115 .CodeMirror-hints::-webkit-scrollbar-button ,
116 .CodeMirror-hints::-webkit-scrollbar-track ,
117 .CodeMirror-hints::-webkit-scrollbar-track-piece ,
118 .CodeMirror-hints::-webkit-scrollbar-thumb ,
119 .CodeMirror-hints::-webkit-scrollbar-corner ,
120 .CodeMirror-hints::-webkit-resizer {
121   background: transparent ;
122 }
123
124 .CodeMirror-hints::-webkit-scrollbar {
125   width: 12px;
126   height: 12px;
127   -webkit-border-radius: 5px;
128   border-radius: 5px;
129 }
130
131 .CodeMirror-hints::-webkit-scrollbar-track-piece {
132   -webkit-border-radius: 5px;
133   border-radius: 5px;
134 }
135
136 .CodeMirror-hints::-webkit-scrollbar-thumb {
137   background: #ddd;
138   border-radius: 5px;
139   background-clip: content-box;
140   border: 2px solid transparent;
```



```
141    }
142
143 .CodeMirror-hints::-webkit-scrollbar-button {
144   width:0;
145   height:0;
146 }
147
148 .CodeMirror-search-dialog {
149   font-family: Roboto;
150   font-size: 16px;
151   display: none;
152   position: absolute;
153   top: 0;
154   right: 15px;
155   background-color: rgba(255, 255, 255, 0.9);
156   padding: 10px;
157   border-radius: 0 0 5px 5px;
158   box-shadow: 0 0 10px #00000026;
159   z-index: 3!important;
160 }
161
162 .CodeMirror-search-find, .CodeMirror-search-replace {
163   margin: 0;
164   border: 1px solid #ddd;
165   padding: 3px 5px;
166   display: block;
167   font-size: 16px;
168   border-radius: 3px;
169   margin-right: 10px;
170   float: left;
171   margin-bottom: 10px;
172   font-weight: 100;
173 }
174
175 .CodeMirror-search-find::placeholder, .CodeMirror-search-replace::placeholder {
176   font-weight: 100;
177   color: #999;
178 }
179
180 .CodeMirror-search-find:focus, .CodeMirror-search-replace:focus {
181   border: 1px solid #f7df1e;
182 }
183
184 .CodeMirror-search-find-prev-btn, .CodeMirror-search-find-next-btn,
185 .CodeMirror-search-replace-btn, .CodeMirror-search-replace-all-btn {
186   cursor: pointer;
187   background-size: 20px;
188   height: 20px;
189   width: 20px;
190   opacity: 0.5;
191   display: block;
192   float: left;
193   margin-top: 3px;
194 }
```



```
195
196 .CodeMirror-search-find-prev-btn:hover, .CodeMirror-search-find-next-btn:hover
197 , .CodeMirror-search-replace-btn:hover, .CodeMirror-search-replace-all-btn:hover
198 {
199   opacity: 1;
200 }
201
202 .CodeMirror-search-find-prev-btn {
203   background-image: url(../img/arrow.svg);
204   transform: rotate(90deg);
205   margin-right: 10px;
206 }
207
208 .CodeMirror-search-find-next-btn {
209   background-image: url(../img/arrow.svg);
210   transform: rotate(-90deg);
211 }
212
213 .CodeMirror-search-replace-btn {
214   background-image: url(../img/replace.svg);
215   margin-right: 10px;
216 }
217
218 .CodeMirror-search-replace-all-btn {
219   background-image: url(../img/replace-all.svg);
220 }
221
222 .CodeMirror-search-case-btn, .CodeMirror-search-regex-btn {
223   cursor: pointer;
224   background-size: 20px;
225   height: 20px;
226   width: 20px;
227   opacity: 0.5;
228   display: block;
229   float: left;
230 }
231
232 .CodeMirror-search-case-btn:hover, .CodeMirror-search-regex-btn:hover {
233   opacity: 0.7;
234 }
235
236 .CodeMirror-search-case-btn.active, .CodeMirror-search-regex-btn.active {
237   opacity: 1;
238 }
239
240 .CodeMirror-search-case-btn {
241   margin-left: 5px;
242   margin-right: 10px;
243   background-image: url(../img/match-case.svg);
244 }
245
246 .CodeMirror-search-regex-btn {
247   background-image: url(../img/regex.svg);
248 }
```

```

248
249 .CodeMirror-search-bottom-right {
250   float: right;
251   margin-top: 5px;
252   font-weight: 100;
253   color: #999;
254 }
255
256 .CodeMirror-search-close-btn {
257   cursor: pointer;
258   background-size: 15px;
259   height: 15px;
260   width: 15px;
261   opacity: 0.4;
262   display: block;
263   float: right;
264   background-image: url(../img/close.svg);
265   margin-left: 10px;
266 }
267
268 .CodeMirror-search-close-btn:hover {
269   opacity: 1;
270 }
```

Listing 16 - codemirror-presentation-editor-custom.css

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
  abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
  strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
  label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas,
  details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby,
  section, summary, time, mark, audio, video {border:0; font-size:100%;}
  font: inherit; vertical-align: baseline; margin:0; padding:0}article, aside,
  details, figcaption, figure, footer, header, hgroup, menu, nav, section{
  display:block}body{line-height:1}ol, ul{list-style:none}blockquote, q{
  quotes:none}blockquote:before, blockquote:after, q:before, q:after{
  content:none}table{border-collapse:collapse; border-spacing:0}
2
3 html {
4   overflow: hidden;
5 }
6
7 :focus {
8   outline: 0;
9 }
10
11 div {
12   z-index: 2;
13   box-sizing: border-box;
14 }
15
16 section {
17   position: relative;
18   z-index: 2;
19 }
```

```
21 body {
22   font-family: 'Roboto', Arial;
23   width: 100%;
24   background: #fff;
25 }
26
27 lang {
28   display: none;
29 }
30
31 .clear {
32   clear: both;
33 }
34
35 .float-left {
36   float: left;
37 }
38
39 .float-right {
40   float: right;
41 }
42
43 #app-logo {
44   height: 18px;
45   padding-top: 6px;
46   padding-left: 10px;
47   padding-right: 10px;
48   float: left;
49 }
50
51 #app-title {
52   float: right;
53   font-size: 16px;
54   padding-top: 8px;
55   font-weight: 100;
56   border-right: 1px solid #fffff2b;
57   padding-right: 10px;
58   height: 30px;
59 }
60
61 #window-controls {
62   -webkit-app-region: no-drag;
63   float: right;
64   padding-left: 6px
65 }
66
67 #window-controls li {
68   float: left;
69   padding: 1px 10px;
70   display: inline-block;
71   font-size: 18px;
72   font-weight: 100;
73   line-height: 24px;
74   -webkit-transition: all .2s linear;
75   transition: all .2s linear;
```

```
76     cursor: pointer;  
77     -webkit-app-region: no-drag;  
78 }  
79  
80 #window-controls li img {  
81     width: 16px;  
82     height: 16px;  
83     user-select: none;  
84     -webkit-user-drag: none;  
85     user-drag: none;  
86 }  
87  
88 #window-controls li:hover,  
89 #window-controls li:active {  
90     background: #fffff99;  
91 }  
92  
93 #window-controls #win-close:hover,  
94 #window-controls #win-close:active {  
95     background: #cf00f;  
96 }  
97  
98 #window-controls img {  
99     width: 100%;  
100    height: 100%;  
101    filter: invert(1);  
102 }  
103  
104 .tabs {  
105     font-family: 'Roboto', Arial;  
106     border-radius: 0;  
107     padding-right: 65px;  
108     background: #eee;  
109     font-size: 16px;  
110     padding-top: 5px;  
111     height: 43px;  
112 }  
113  
114 .tab-content, .tab-close {  
115     cursor: pointer!important;  
116 }  
117  
118 .tab-saved {  
119     flex-grow: 0;  
120     flex-shrink: 0;  
121     position: relative;  
122     width: 5px;  
123     height: 5px;  
124     border-radius: 50%;  
125     background: transparent;  
126     top: 5px;  
127     right: 5px;  
128 }  
129  
130 .changed .tab-saved {
```



```
131     background: #2e85c7;
132 }
133
134 #new-tab {
135     position: absolute;
136     top: 5px;
137     right: 35px;
138     height: 30px;
139     background: transparent;
140     padding: 5px;
141     border-radius: 15px;
142     transition: all .2s linear;
143     cursor: pointer;
144 }
145
146 #new-tab:hover {
147     background: #ccc;
148 }
149
150 #editor-more-icon {
151     position: absolute;
152     top: 5px;
153     right: 5px;
154     height: 30px;
155     background: transparent;
156     padding: 5px;
157     border-radius: 15px;
158     transition: all .2s linear;
159     cursor: pointer;
160     opacity: 0.7;
161 }
162
163 #editor-more-icon:hover {
164     background: #ccc;
165 }
166
167 #editor-menu-container {
168     background: #12568a;
169     background: linear-gradient(to left, #2e85c7, #12568a);
170     color: #fff;
171     user-select: none;
172     -webkit-user-drag: none;
173     user-drag: none;
174     -webkit-app-region: drag;
175     border-top: 2px solid #f7df1e;
176 }
177
178 #editor-menu li {
179     float: left;
180     padding: 5px 10px;
181     display: inline-block;
182     font-size: 16px;
183     font-weight: 100;
184     line-height: 20px;
185     -webkit-transition: all .2s linear;
```

```
186 transition: all .2s linear;
187 cursor: pointer;
188 -webkit-app-region: no-drag;
189 }
190
191 #editor-menu li:hover {
192 background: #fffff4 0 ;
193 }
194
195 #editor-menu li img {
196 float: left ;
197 width: 20px ;
198 height: 20px ;
199 margin-right: 5px ;
200 user-select: none ;
201 -webkit-user-drag: none ;
202 user-drag: none ;
203 }
204
205 #close-dialog-cont {
206 position: absolute ;
207 top: 75px ;
208 left: 6px ;
209 right: 6px ;
210 bottom: 7px ;
211 background-color: rgba(255, 255, 255, 0.5) ;
212 }
213
214 #close-dialog {
215 margin: 0 auto ;
216 width: 400px ;
217 position: relative ;
218 background: #fff ;
219 box-shadow: 0 0 10px #00000026;
220 border-radius: 10px ;
221 padding: 20px ;
222 padding-bottom: 20px ;
223 color: #333 ;
224 margin-bottom: 20px ;
225 margin-top: 20px ;
226 line-height: 22px ;
227 }
228
229 #close-dialog-header {
230 position: relative ;
231 border-bottom: 1px solid #666 ;
232 margin-bottom: 25px ;
233 padding-bottom: 10px ;
234 user-select: none ;
235 -webkit-user-drag: none ;
236 user-drag: none ;
237 }
238
239 #close-dialog-header span {
240 color: #666 ;
```



```
241 display: block;
242 font-weight: bold;
243 font-size: 24px;
244 border-radius: 3px;
245 }
246
247 #close-file {
248   font-weight: bold;
249 }
250
251 #close-dialog-buttons {
252   float: right;
253   padding-top: 10px;
254 }
255
256 #close-dialog-buttons button {
257   color: #fff;
258   background: #2e85c7;
259   cursor: pointer;
260   border: 2px solid #2e85c7;
261   transition: background 0.2s linear, color 0.2s linear;
262   -webkit-transition: background 0.2s linear, color 0.2s linear;
263   border-radius: 3px;
264   padding: 2px 5px;
265   font-size: 16px;
266   margin: 3px 1px;
267   font-weight: normal;
268   text-decoration: none;
269 }
270
271 #close-dialog-buttons button:hover {
272   background: transparent;
273   color: #2e85c7;
274 }
275
276 #editor-more-popup {
277   position: absolute;
278   display: block;
279   border-radius: 5px;
280   width: 245px;
281   top: 81px;
282   right: 10px;
283   background: #ffffff;
284   border: 1px solid #f7df1e;
285   border-bottom: 1px solid #f7df1e;
286 }
287
288 #editor-more-popup .popup-triangle {
289   position: absolute;
290   width: 0;
291   height: 0;
292   border-style: solid;
293   border-width: 10px 10px 0 10px;
294   border-color: #f7df1e transparent transparent transparent;
295   top: -11px;
```

```
296     right: 5px;  
297     transform: rotate(180deg);  
298 }  
299  
300 #editor-more-popup li {  
301     margin: 5px;  
302     border-radius: 5px;  
303     border: 1px solid #dddddd;  
304     color: #999;  
305     padding: 5px 10px;  
306     font-weight: 300;  
307     font-size: 18px;  
308     cursor: pointer;  
309     transition: background 0.2s linear;  
310     -webkit-transition: background 0.2s linear;  
311 }  
312  
313 #editor-more-popup li:hover {  
314     background: #f3f3f3;  
315 }  
316  
317 #editor-more-popup li img {  
318     float: left;  
319     width: 20px;  
320     height: 20px;  
321     margin-right: 10px;  
322     user-select: none;  
323     -webkit-user-drag: none;  
324     user-drag: none;  
325     margin-top: -1px;  
326     filter: invert(1);  
327     opacity: 0.3;  
328 }  
329  
330 .arduino {  
331     display: none;  
332 }  
333  
334 .file-ino .arduino {  
335     display: block;  
336 }  
337  
338 .file-ino #run-menu {  
339     opacity: 0.6;  
340     pointer-events: none;  
341 }  
342  
343 @media only screen and (max-width: 830px) {  
344     #editor-menu li img {  
345         display: none;  
346     }  
347 }  
348  
349 @media only screen and (max-width: 670px) {  
350     #editor-menu li {
```

```

351     padding: 5px;
352 }
353
354 #editor-menu li img {
355     display: block;
356     margin-right: 0px;
357 }
358
359 #editor-menu li str {
360     display: none;
361 }
362 }
```

Listing 17 - editor.css

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas,
details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby,
section, summary, time, mark, audio, video { border: 0; font-size: 100%;
font: inherit; vertical-align: baseline; margin: 0; padding: 0 } article, aside,
details, figcaption, figure, footer, header, hgroup, menu, nav, section {
display: block } body { line-height: 1 } ol, ul { list-style: none } blockquote, q {
quotes: none } blockquote:before, blockquote:after, q:before, q:after {
content: none } table { border-collapse: collapse; border-spacing: 0 }

2
3 html {
4     overflow: hidden;
5 }
6
7 :focus {
8     outline: 0;
9 }
10
11 div {
12     z-index: 2;
13     box-sizing: border-box;
14 }
15
16 section {
17     position: relative;
18     z-index: 2;
19 }
20
21 body {
22     font-family: 'Roboto', Arial;
23     width: 100%;
24     background: #fff;
25 }
26
27 .clear {
28     clear: both;
29 }
30
31 .float-left {
```



```
32     float: left ;
33 }
34
35 .float-right {
36   float: right ;
37 }
38
39 #figure-menu-button {
40   position: absolute;
41   left: 0;
42   right: 0;
43   top: -5px;
44   z-index: 9999;
45   transition: all .2s linear;
46   width: 20px;
47   height: 20px;
48   border-radius: 0 0 50% 50%;
49   background-color: #12568a;
50   margin: 5px;
51   cursor: pointer;
52 }
53
54 #figure-menu-button.active , #figure-menu-button.hovered {
55   margin-top: 25px;
56 }
57
58 #figure-menu-button img {
59   transition: transform .2s linear;
60   transform: rotate(180deg);
61   width: 60%;
62   display: block;
63   margin: 0 auto;
64   margin-top: 4px;
65 }
66
67 #figure-menu-button.active img , #figure-menu-button.hovered img {
68   transform: rotate(0deg);
69 }
70
71 #figure-menu-container {
72   background: #12568a;
73   background: linear-gradient(to left , #2e85c7 , #12568a);
74   color: #fff;
75   user-select: none;
76   -webkit-user-drag: none;
77   user-drag: none;
78   max-height: 5px;
79   overflow: hidden;
80   transition: all .2s linear;
81   position: absolute;
82   left: 0;
83   right: 0;
84   top: 0;
85   z-index: 9998;
86 }
```

```

87
88 #figure-menu-container .active , #figure-menu-container .hovered {
89     max-height: 30px;
90     border-bottom: 2px solid #ccc;
91 }
92
93 #figure-menu ul {
94     padding-left: 25px;
95 }
96
97 #figure-menu li {
98     float: left;
99     padding: 5px;
100    display: inline-block;
101    font-size: 16px;
102    font-weight: 100;
103    line-height: 20px;
104    -webkit-transition: all .2s linear;
105    transition: all .2s linear;
106    cursor: pointer;
107 }
108
109 #figure-menu li:hover {
110     background: #fffff4 0;
111 }
112
113 #figure-menu li img {
114     float: left;
115     width: 20px;
116     height: 20px;
117     margin-right: 7px;
118     user-select: none;
119     -webkit-user-drag: none;
120     user-drag: none;
121 }
122
123 .plot-cont {
124     position: absolute;
125     top: 5px;
126     left: 0;
127     right: 0;
128     bottom: 0;
129     z-index: 1;
130 }
131
132 .modebar-container , .plotly-notifier {
133     display: none;
134 }
135
136 #figure-menu li .specific-2d , #figure-menu li .specific-3d {
137     display: none;
138 }
139
140 #figure-menu .figure-2d li .specific-2d {
141     display: initial;

```

```

142 }
143
144 #figure-menu figure-3d li .specific-3d {
145     display: initial;
146 }
147
148 @media only screen and (max-width: 720px) {
149     #figure-menu li img {
150         display: none;
151     }
152 }
153
154 @media only screen and (max-width: 600px) {
155     #figure-menu li img {
156         display: block;
157         padding-bottom: 10px;
158     }
159
160     #figure-menu li {
161         padding: 5px;
162         width: 20px;
163         height: 20px;
164         overflow: hidden;
165     }
166 }
```

Listing 18 - figure.css

```

1 . hljs {
2     color: #000;
3 }
4
5 . hljs-built_in ,
6 . hljs-name ,
7 . hljs-selector-tag ,
8 . hljs-tag {
9     color: #00f ;
10 }
11 . hljs-addition ,
12 . hljs-attribute ,
13 . hljs-section ,
14 . hljs-template-tag ,
15 . hljs-template-variable ,
16 . hljs-type {
17     color: #a31515;
18 }
19
20 . hljs-literal {
21     color: #0000ff ;
22 }
23
24 . hljs-deletion ,
25 . hljs-meta ,
26 . hljs-selector-attr ,
27 . hljs-selector-pseudo {
28     color: #2b91af;
```

```

29 }
30 .hljs-doctag {
31   color: grey;
32 }
33 .hljs-attr {
34   color: red;
35 }
36 .hljs-bullet,
37 .hljs-link,
38 .hljs-symbol {
39   color: #00b0e8;
40 }
41 .hljs-emphasis {
42   font-style: italic;
43 }
44 .hljs-strong {
45   font-weight: 700;
46 }
47
48 .hljs-keyword {
49   color: #00f;
50   font-weight: bold;
51 }
52
53 .hljs-variable, .hljs-title {
54   color: black;
55 }
56
57 .hljs-number {
58   color: #ff8000;
59 }
60
61 .hljs-comment {
62   color: #008000;
63 }
64
65 .hljs-string,
66 .hljs-quote {
67   color: #808080;
68 }

```

Listing 19 - highlight-notepadpp-theme.css

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
  abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
  strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
  label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas
  , details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby
  , section, summary, time, mark, audio, video{border:0;font-size:100%;
  font:inherit;vertical-align:baseline;margin:0;padding:0}article, aside,
  details, figcaption, figure, footer, header, hgroup, menu, nav, section{
  display:block}body{line-height:1}ol, ul{list-style:none}blockquote, q{
  quotes:none}blockquote:before, blockquote:after, q:before, q:after{
  content:none}table{border-collapse:collapse; border-spacing:0}
2
3 html {

```



```
4      overflow: hidden;
5  }
6
7  b {
8      font-weight: bold;
9  }
10
11 i {
12     font-style: italic;
13 }
14
15 :focus {
16     outline: 0;
17 }
18
19 div {
20     z-index: 2;
21     box-sizing: border-box;
22 }
23
24 section {
25     position: relative;
26     z-index: 2;
27 }
28
29 body {
30     font-family: 'Roboto', Arial;
31     width: 100%;
32     background: #fff;
33 }
34
35 lang {
36     display: none;
37 }
38
39 .clear {
40     clear: both;
41 }
42
43 .float-left {
44     float: left;
45 }
46
47 .float-right {
48     float: right;
49 }
50
51 .monospaced {
52     font-family: 'RobotoMono';
53 }
54
55 #app-logo {
56     height: 24px;
57     padding-top: 8px;
58     padding-left: 10px;
```



```
59     float: left ;
60 }
61
62 #app-title {
63     float: right;
64     font-size: 20px;
65     padding-top: 9px;
66     font-weight: 100;
67     border-right: 1px solid #fffff2b ;
68     padding-right: 10px;
69     height: 40px;
70 }
71
72 #window-controls {
73     -webkit-app-region: no-drag;
74     float: right;
75     padding-left: 6px
76 }
77 #window-controls li {
78     float: left;
79     padding: 4px 10px;
80     display: inline-block;
81     font-size: 18px;
82     font-weight: 100;
83     line-height: 24px;
84     -webkit-transition: all .2s linear;
85     transition: all .2s linear;
86     cursor: pointer;
87     -webkit-app-region: no-drag;
88 }
89
90 #window-controls li img {
91     width: 16px;
92     height: 16px;
93     user-select: none;
94     -webkit-user-drag: none;
95     user-drag: none;
96 }
97
98 #window-controls li:hover,
99 #window-controls li:active {
100     background: #fffff99 ;
101 }
102
103 #window-controls #win-close:hover ,
104 #window-controls #win-close:active {
105     background: #cf000f;
106 }
107
108 #window-controls img {
109     width: 100%;
110     height: 100%;
111     filter: invert(1);
112 }
113
```

```
114 #main-menu-container {  
115   background: #12568a;  
116   background: linear-gradient(to left, #2e85c7, #12568a);  
117   color: #fff;  
118   user-select: none;  
119   -webkit-user-drag: none;  
120   user-drag: none;  
121   -webkit-app-region: drag;  
122   border-top: 2px solid #f7df1e;  
123 }  
124  
125 #company-logo {  
126   filter: invert(1);  
127   height: 30px;  
128   float: left;  
129   padding: 5px;  
130   padding-left: 10px;  
131   user-select: none;  
132   -webkit-user-drag: none;  
133   user-drag: none;  
134   margin-left: 10px;  
135   border-left: 1px solid #00000054;  
136   border-right: 1px solid #00000054;  
137   padding-right: 10px;  
138 }  
139  
140 #main-menu {  
141   padding: 0 10px;  
142   -webkit-app-region: no-drag;  
143 }  
144  
145 #main-menu li {  
146   float: left;  
147   padding: 8px 10px;  
148   display: inline-block;  
149   font-size: 18px;  
150   font-weight: 100;  
151   line-height: 24px;  
152   -webkit-transition: all .2s linear;  
153   transition: all .2s linear;  
154   cursor: pointer;  
155   -webkit-app-region: no-drag;  
156 }  
157  
158 #main-menu li:hover {  
159   background: #fffff4 0;  
160 }  
161  
162 #main-menu li img {  
163   float: left;  
164   width: 24px;  
165   height: 24px;  
166   user-select: none;  
167   -webkit-user-drag: none;  
168   user-drag: none;
```

```
169     margin-right: 5px;
170 }
171
172 #paths-container li {
173     padding-bottom: 5px;
174     padding-top: 5px;
175     line-height: 22px;
176     padding-left: 10px;
177     padding-right: 10px;
178     margin: 0 10px;
179     position: relative;
180     white-space: pre-wrap;
181     border-bottom: 1px solid #f7df1e;
182     min-height: 22px;
183     cursor: pointer;
184     transition: all linear 0.3s;
185 }
186
187 #paths-container li.inactive {
188     color: #cf000f;
189 }
190
191 #paths-container li:after {
192     transition: all linear 0.3s;
193 }
194
195 #paths-container li:not(.no-paths):hover:after {
196     content: '';
197     position: absolute;
198     top: 3px;
199     left: 0px;
200     right: 0px;
201     bottom: 3px;
202     border: 1px solid #999;
203     border-radius: 3px;
204     background: #fffff69;
205     z-index: -1;
206 }
207
208 #paths-container li:last-child {
209     border-bottom: 1px solid transparent;
210 }
211
212 #paths-container img.remove-path {
213     height: 15px;
214     width: 15px;
215     position: absolute;
216     right: 5px;
217     top: 8px;
218     opacity: 0.3;
219     display:none;
220     transition: all linear 0.3s;
221 }
222
223 #paths-container li:hover img.remove-path {
```



```
224     display: block;
225 }
226
227 #paths-container li:hover img.remove-path:hover {
228     display: block;
229     opacity: 0.6;
230 }
231
232 #paths-container li.no-paths {
233     text-align: center;
234     font-size: 24px;
235 }
236
237 #folder-navigation-container {
238     padding: 5px 10px;
239     border-bottom: 1px solid #dadce0;
240     color: #5f6368;
241     user-select: none;
242     -webkit-user-drag: none;
243     user-drag: none;
244 }
245
246 #folder-navigation-container img, #folder-navigation-container i {
247     height: 18px;
248     width: 18px;
249     display: block;
250     margin: 0 auto;
251     -webkit-user-drag: none;
252 }
253
254 #folder-navigation-container #save-path i {
255     background-image: url(../img/star.svg);
256     background-size: 18px 18px;
257     background-position: center;
258     transition: all .2s linear;
259 }
260
261 #folder-navigation-container #save-path.saved i {
262     background-image: url(../img/star-filled.svg);
263 }
264
265 #folder-navigation-container i.i-next-folder {
266     background-image: url(../img/next.svg);
267     background-size: 14px 14px;
268     background-position: center;
269     width: 18px;
270     height: 14px;
271     display: inline-block;
272     margin-bottom: -2px;
273 }
274
275 #folder-navigation-container .next-folder {
276     transform: rotate(180deg);
277 }
278
```

```
279 #folder-navigation-container .up-folder {  
280   transform: rotate(90deg);  
281 }  
282  
283 #folder-navigation-container .previous-folder.disabled,  
284 #folder-navigation-container .next-folder.disabled,  
285 #folder-navigation-container .up-folder.disabled {  
286   opacity: 0.3;  
287   pointer-events: none;  
288 }  
289  
290 #folder-navigation-container .address-line-container {  
291   float: left;  
292   width: calc(100% - 165px);  
293   position: relative;  
294 }  
295  
296 #folder-navigation-container .address-line {  
297   width: calc(100% - 76px);  
298   height: 20px;  
299   background: #f1f3f4;  
300   border-radius: 14px;  
301   padding: 2px 35px;  
302   margin-left: 3px;  
303   border: 2px solid #f1f3f4;  
304   -webkit-transition: all .2s linear;  
305   transition: all .2s linear;  
306   color: #5f6368;  
307   font-family: 'Roboto', Arial;  
308   font-size: 14px;  
309 }  
310  
311 #folder-navigation-container .current-address-cont {  
312   position: absolute;  
313   left: 40px;  
314   right: 35px;  
315   top: 2px;  
316   background: #f1f3f4;  
317   padding: 4px 0;  
318   pointer-events: none;  
319   white-space: nowrap;  
320   overflow: hidden;  
321   bottom: 2px;  
322 }  
323  
324 #folder-navigation-container .current-address-wrap {  
325   position: relative;  
326   max-width: 100%;  
327   display: inline-block;  
328 }  
329  
330 #folder-navigation-container .current-address {  
331   float: right;  
332 }  
333
```



```
334 #folder-navigation-container .current-address {  
335   content: '';  
336   clear: both;  
337 }  
338  
339 #folder-navigation-container .address-line:focus {  
340   background: #fff;  
341   border: 2px solid #f7df1e;  
342 }  
343  
344 #folder-navigation-container .address-line:focus ~ .current-address-cont,  
345 #folder-navigation-container .address-line:focus ~ .save-icon {  
346   display: none;  
347 }  
348  
349 #folder-navigation-container .folder-icon {  
350   position: absolute;  
351   left: 14px;  
352   top: 4px;  
353   pointer-events: none;  
354 }  
355  
356 #folder-navigation-container span.folder {  
357   cursor: pointer;  
358   pointer-events: all;  
359 }  
360  
361 #folder-navigation-container .folder-address {  
362   font-size: 14px;  
363 }  
364  
365 #folder-navigation-container #save-path {  
366   position: absolute;  
367   top: 2.5px;  
368   padding: 2px;  
369   right: 5px;  
370   background: #fffff00;  
371   margin: 0px 2px;  
372   cursor: pointer;  
373   width: 28px;  
374   -webkit-transition: all .2s linear;  
375   transition: all .2s linear;  
376   display: block;  
377   height: 22px;  
378   -webkit-user-drag: none;  
379 }  
380  
381 #folder-navigation-container #save-path:hover {  
382   background: #fff;  
383   border-radius: 12px;  
384 }  
385  
386 #folder-navigation-container .button {  
387   padding: 5px;  
388   margin: 0px 2px;
```

```
389 -webkit-transition: all .2s linear;
390 transition: all .2s linear;
391 cursor: pointer;
392 }
393
394 #folder-navigation-container .button:hover {
395 background: #f1f3f4;
396 border-radius: 14px;
397 }
398
399 #panels-container {
400 position: absolute;
401 top: 81px;
402 bottom: 25px;
403 left: 0px;
404 right: 0px;
405 }
406
407 #panels-container .horizontal-resizer {
408 width: 100%;
409 background-image: url(../img/hdrag.svg);
410 background-repeat: no-repeat;
411 background-position: center center;
412 background-size: auto 6px;
413 cursor: n-resize;
414 user-select: none;
415 -webkit-user-drag: none;
416 user-drag: none;
417 position: absolute;
418 }
419
420 #panels-container .horizontal-resizer:after {
421 content: '';
422 border-bottom: 1px solid transparent;
423 position: absolute;
424 top: 4px;
425 left: 0px;
426 right: 0px;
427 pointer-events: none;
428 }
429
430 #panels-container .vertical-resizer {
431 height: 100%;
432 background-image: url(../img/vdrag.svg);
433 background-repeat: no-repeat;
434 background-position: center center;
435 background-size: 6px auto;
436 cursor: w-resize;
437 user-select: none;
438 -webkit-user-drag: none;
439 user-drag: none;
440 position: absolute;
441 }
442
443 #panels-container .vertical-resizer:after {
```

```
444 content: '';
445 border-right: 1px solid transparent;
446 position: absolute;
447 top: 0px;
448 bottom: 0px;
449 left: 4px;
450 pointer-events: none;
451 }
452
453 #panels-container .vertical-resizer:hover:after, #panels-container .
454   horizontal-resizer:hover:after {
455     border-right: 1px solid #f7df1e;
456   }
457
458 #panels-container .horizontal-resizer:hover:after {
459   border-bottom: 1px solid #f7df1e;
460 }
461
462 #panels-container .panel-container {
463   height: 100%;
464 }
465
466 #panels-container .cell-padding {
467   padding: 5px;
468   height: 100%;
469   width: 100%;
470   position: relative;
471 }
472
473 #left-panel {
474   height: 100%;
475   width: 20%;
476   position: absolute;
477   left: 0;
478   top: 0;
479 }
480
481 #right-panel {
482   height: 100%;
483   width: 80%;
484   position: absolute;
485   left: 20%;
486   top: 0;
487 }
488
489 #left-top-panel, #left-middle-panel, #left-bottom-panel {
490   width: 100%;
491   height: 33.333%;
492 }
493
494 #panels-container .panel-title {
495   margin: 5px 10px;
496   margin-bottom: 0px;
497   color: #666;
498   padding-bottom: 3px;
```

```
498 border-bottom: 1px solid #f7df1e;
499 font-weight: 300;
500 font-size: 14px;
501 user-select: none;
502 -webkit-user-drag: none;
503 user-drag: none;
504 white-space: nowrap;
505 overflow: hidden;
506 text-overflow: ellipsis;
507 }
508
509 #left-middle-panel .panel-title, #left-bottom-panel .panel-title {
510   padding-right: 20px;
511 }
512
513 #right-panel .panel-title {
514   padding-right: 155px;
515 }
516
517 #panels-container .panel-container {
518   border: 1px solid #f1f3f4;
519   -webkit-transition: border .2s linear;
520   transition: border .2s linear;
521   position: relative;
522   border-radius: 3px;
523 }
524
525 #panels-container .panel-container:hover {
526   border: 1px solid #f7df1e;
527 }
528
529 #panels-container .panel {
530   overflow: auto;
531   position: absolute;
532   top: 23px;
533   left: 0px;
534   right: 0px;
535   bottom: 0px;
536 }
537
538 #help-container, #info-container, #settings-container, #paths-container, #
539   script-path-container {
540   display: none;
541   top: 79px;
542   left: 0;
543   right: 0;
544   bottom: 25px;
545   position: absolute;
546   -webkit-backdrop-filter: blur(5px);
547   backdrop-filter: blur(5px);
548   background-color: rgba(255, 255, 255, 0.7);
549 }
550 #info-container .app-logo {
551   width: 150px;
```



```
552     text-align: center;
553     margin: 0 auto;
554     display: block;
555 }
556
557 #info-container .app-name {
558     font-size: 34px;
559     color: #000;
560     text-align: center;
561     padding-top: 10px;
562     line-height: 46px;
563 }
564
565 #info-container .app-version {
566     font-size: 16px;
567     text-align: center;
568     font-weight: 100;
569     line-height: 18px;
570 }
571
572 #info-container .app-company {
573     font-size: 24px;
574     text-align: center;
575     font-weight: 100;
576     line-height: 30px;
577 }
578
579 #info-container .company-logo {
580     width: 100px;
581     margin: 0 auto;
582     display: block;
583     padding: 10px;
584 }
585
586 #info-container p {
587     line-height: 22px;
588     text-align: center;
589     font-weight: 100;
590 }
591
592 #info-container ul {
593     padding-top: 20px;
594     padding-left: 40px;
595     list-style: inside square;
596 }
597
598 #info-container li {
599     padding: 10px 0;
600 }
601
602 #info-container li a {
603     text-decoration: none;
604     color: #444;
605     font-size: 22px;
606 }
```



```
607
608 #info-container li a:hover {
609   opacity: 0.4;
610 }
611
612 #file-browser-cont {
613   margin: 0 10px;
614   margin-left: 0px;
615 }
616
617 #file-browser-cont ul {
618   margin-left: 15px;
619 }
620
621 #file-browser li {
622   position: relative;
623   background-repeat: no-repeat;
624   background-position: 18px 5px;
625   background-size: 18px 18px;
626 }
627
628 #file-browser li span {
629   font-size: 14px;
630   padding-bottom: 5px;
631   padding-top: 5px;
632   line-height: 18px;
633   padding-left: 40px;
634   padding-right: 0px;
635   user-select: none;
636   -webkit-user-drag: none;
637   user-drag: none;
638   white-space: nowrap;
639   overflow: hidden;
640   text-overflow: ellipsis;
641   min-height: 18px;
642   cursor: pointer;
643   color: #666;
644   position: relative;
645   display: block;
646 }
647
648 #file-browser li.folder {
649   background-image: url(../img/browser-folder.svg);
650 }
651
652 #file-browser li.link {
653   background-image: url(../img/browser-link.svg);
654 }
655
656 #file-browser li.file {
657   background-image: url(../img/browser-file.svg);
658 }
659
660 #file-browser li.file.js {
661   background-image: url(../img/browser-file-js.svg);
```

```
662 }
663
664 #file-browser li .file .jsl {
665   background-image: url(../img/browser-file-jsl.svg);
666 }
667
668 #file-browser li .file .json {
669   background-image: url(../img/browser-file-json.svg);
670 }
671
672 #file-browser li i .expend {
673   cursor: pointer;
674   background: url(../img/expand.svg);
675   background-size: 12px;
676   height: 12px;
677   width: 12px;
678   display: block;
679   position: absolute;
680   left: 0px;
681   top: 7px;
682   transition: all .2s linear;
683   filter: grayscale(1);
684 }
685
686 #file-browser li i .expend .expended {
687   transform: rotate(90deg);
688   filter: none;
689 }
690
691 #command-history > div {
692   font-size: 14px;
693   padding-bottom: 5px;
694   padding-top: 5px;
695   line-height: 18px;
696   padding-left: 10px;
697   padding-right: 10px;
698   margin: 0 10px;
699   position: relative;
700   white-space: pre-wrap;
701   border-bottom: 1px solid #eee;
702   min-height: 18px;
703   cursor: pointer;
704   color: #666;
705   word-break: break-word;
706 }
707
708 #workspace {
709   overflow-y: scroll !important;
710   margin-left: 10px;
711 }
712
713 #workspace .table .col {
714   font-size: 14px;
715   padding-bottom: 5px;
716   padding-top: 5px;
```



```
717 line-height: 18px;  
718 padding-left: 10px;  
719 padding-right: 10px;  
720 margin: 0 10px;  
721 position: relative;  
722 white-space: nowrap;  
723 overflow: hidden;  
724 text-overflow: ellipsis;  
725 border-bottom: 1px solid #eee;  
726 min-height: 18px;  
727 cursor: pointer;  
728 color: #666;  
729 margin: 0;  
730 float: left;  
731 border-bottom: none;  
732 left: 0!important;  
733 }  
734  
735 #workspace-table-head .col {  
736 font-size: 14px;  
737 padding-bottom: 5px;  
738 padding-top: 5px;  
739 line-height: 18px;  
740 padding-left: 10px;  
741 padding-right: 10px;  
742 min-height: 18px;  
743 cursor: pointer;  
744 color: #666;  
745 float: left;  
746 border-right: 1px solid #ddd;  
747 white-space: nowrap;  
748 overflow: hidden;  
749 text-overflow: ellipsis;  
750 float: left;  
751 text-align: center;  
752 margin-bottom: -1px;  
753 margin-top: 3px;  
754 }  
755  
756 #workspace .table .row {  
757 border-bottom: 1px solid #eee;  
758 }  
759  
760 #workspace .row:after, #workspace-table-head .row:after {  
761 content: '';  
762 clear: both;  
763 display: block;  
764 }  
765  
766 #workspace-table-head {  
767 border-bottom: 1px solid #ddd;  
768 margin: 0 10px;  
769 position: absolute;  
770 top: 23px;  
771 left: 0;
```



```
772     right: 0;
773     background: #fff;
774     z-index: 3;
775     user-select: none;
776     -webkit-user-drag: none;
777     user-drag: none;
778     margin-right: 12px;
779     pointer-events: none;
780   }
781
782 #workspace .table {
783   margin-top: 3px;
784   text-align: center;
785   padding-top: 29px;
786   user-select: none;
787   -webkit-user-drag: none;
788   user-drag: none;
789 }
790
791 #workspace .vertical-resizer {
792   cursor: ew-resize;
793   position: absolute;
794   top: 0px;
795   bottom: 3px;
796   width: 5px;
797   left: calc(50% - 5px);
798   pointer-events: auto;
799   background: none;
800 }
801
802 #workspace .vertical-resizer:after {
803   content: '';
804   border-right: 1px solid transparent;
805   position: absolute;
806   top: 0px;
807   bottom: 0px;
808   left: 4px;
809   pointer-events: none;
810 }
811
812 #workspace .vertical-resizer:hover:after {
813   border-right: 1px solid #f7df1e;
814 }
815
816 #workspace .col-1, #workspace-table-head .col-1 {
817   width: 50%;
818   text-align: left;
819 }
820
821 #workspace .col-2, #workspace .col-3,
822 #workspace-table-head .col-2, #workspace-table-head .col-3{
823   width: 25%;
824 }
825
826 #workspace .table .row {
```

```
827     display: block;
828     position: relative;
829 }
830
831 #command-history > div.comment{
832     color: #26a65b;
833 }
834
835 #command-history > div:after , #workspace .table .row:before ,
836 #file-browser li span:hover:after {
837     transition: all linear 0.3s;
838 }
839
840 #command-history > div:last-child , #workspace .row:last-child {
841     border-bottom: 1px solid transparent;
842 }
843
844 #workspace-table-head .col-3 , #workspace .table .col {
845     border-right: 1px solid transparent;
846 }
847
848 #command-history > div:not(.comment):hover:after ,
849 #workspace .table .row:hover:before ,
850 #file-browser li span:hover:after
851 {
852     content: '';
853     position: absolute;
854     top: 3px;
855     left: 0px;
856     right: 0px;
857     bottom: 3px;
858     border: 1px solid #999;
859     border-radius: 3px;
860     z-index: -1;
861     display: block;
862     clear: both;
863 }
864
865 #file-browser li span:hover:after {
866     left: 15px;
867 }
868
869 #command-history-options , #workspace-options , #file-browser-options {
870     position: absolute;
871     top: -1px;
872     right: 8px;
873 }
874
875 #command-history-options.options .options-right ,
876 #workspace-options.options .options-right ,
877 #file-browser-options.options .options-right {
878     float: right;
879 }
880
881 #command-history-options.options i.clear ,
```



```
882 #workspace-options.options i.clear {
883   background: url(../img/clear.svg) no-repeat center;
884 }
885
886 #file-browser-options.options i.refresh {
887   background: url(../img/refresh.svg) no-repeat center;
888 }
889
890 #command-history-options.options i,
891 #workspace-options.options i,
892 #file-browser-options.options i {
893   width: 16px;
894   height: 18px;
895   display: block;
896   background-size: 16px !important;
897   float: left;
898   clear: none;
899   padding: 3px 5px;
900   opacity: 0.3;
901   user-select: none;
902   -webkit-user-drag: none;
903 }
904
905 #command-history-options.options i:hover,
906 #workspace-options.options i:hover,
907 #file-browser-options.options i:hover {
908   opacity: 1;
909   cursor: pointer;
910 }
911
912 #command-history-options.options i.active,
913 #workspace-options.options i.active,
914 #file-browser-options.options i.active{
915   opacity: 0.8;
916 }
917
918 #script-path-dialog-msg {
919   color: #333;
920   line-height: 22px;
921 }
922
923 #script-path {
924   font-weight: bold;
925 }
926
927 #script-path-dialog-buttons{
928   float: right;
929   padding-top: 10px;
930 }
931
932 #script-path-dialog-buttons button {
933   color: #fff ;
934   background: #2e85c7;
935   cursor: pointer;
936   border: 2px solid #2e85c7;
```

```
937 transition: background 0.2s linear, color 0.2s linear;  
938 -webkit-transition: background 0.2s linear, color 0.2s linear;  
939 border-radius: 3px;  
940 padding: 2px 5px;  
941 font-size: 16px;  
942 margin: 3px 1px;  
943 font-weight: normal;  
944 text-decoration: none;  
945 width: auto;  
946 }  
947  
948 #script-path-dialog-buttons button:hover {  
949   background: transparent;  
950   color: #2e85c7;  
951 }  
952  
953 .page-cont {  
954   margin: 0 auto;  
955   width: 460px;  
956   position: relative;  
957   background: #fffff6b;  
958   box-shadow: 0 0 10px #00000026;  
959   border-radius: 10px;  
960   padding: 20px;  
961   padding-bottom: 20px;  
962   color: #777;  
963   margin-bottom: 20px;  
964   margin-top: 20px;  
965   display: flex;  
966   flex-direction: column;  
967   max-height: calc(100% - 40px);  
968 }  
969  
970 .page-cont.wide {  
971   width: 800px;  
972 }  
973  
974 .page-panel {  
975   overflow-y: auto;  
976   padding: 10px;  
977   padding-top: 25px;  
978 }  
979  
980 .page-panel h1 {  
981   font-size: 24px;  
982   color: #000;  
983   font-weight: 100;  
984 }  
985  
986 .page-panel table {  
987   width: 100%;  
988   color: #333;  
989   font-size: 18px;  
990   text-align: left;  
991   margin-top: 20px;
```

```
992 }
993
994 .page-panel table, .page-panel th, .page-panel td {
995   border: 1px solid #ccc;
996   border-collapse: collapse;
997 }
998
999 .page-panel th {
1000   font-weight: bold;
1001   padding: 10px;
1002 }
1003
1004 .page-panel td {
1005   padding: 10px;
1006 }
1007
1008 .panel::-webkit-scrollbar,
1009 .panel::-webkit-scrollbar-button,
1010 .panel::-webkit-scrollbar-track,
1011 .panel::-webkit-scrollbar-track-piece,
1012 .panel::-webkit-scrollbar-thumb,
1013 .panel::-webkit-scrollbar-corner,
1014 .panel::-webkit-resizer {
1015   background: transparent;
1016 }
1017
1018 .panel::-webkit-scrollbar {
1019   width: 12px;
1020   height: 12px;
1021   -webkit-border-radius: 5px;
1022   border-radius: 5px;
1023 }
1024
1025 .panel::-webkit-scrollbar-track-piece {
1026   -webkit-border-radius: 5px;
1027   border-radius: 5px;
1028 }
1029
1030 .panel::-webkit-scrollbar-thumb {
1031   background: #ddd;
1032   border-radius: 5px;
1033   background-clip: content-box;
1034   border: 2px solid transparent;
1035 }
1036
1037 .panel::-webkit-scrollbar-button {
1038   width:0;
1039   height:0;
1040 }
1041
1042 .options-cont input[type="text"], .options-cont input[type="number"], .
1043   options-cont input[type="submit"], .options-cont input[type="password"], .
1044   options-cont textarea, .options-cont select, .options-cont button,
1045   main-dialog input[type="text"], .main-dialog input[type="number"], .
1046   main-dialog input[type="submit"], .main-dialog input[type="password"], .
```

```
1044 main-dialog textarea, .main-dialog select, .main-dialog button {  
1045   font-family: 'Roboto';  
1046   border: 2px solid #ccc;  
1047   color: #333;  
1048   font-size: 18px;  
1049   padding-left: 10px;  
1050   padding-right: 10px;  
1051   padding-top: 8px;  
1052   padding-bottom: 8px;  
1053   width: calc(100% - 24px);  
1054   margin-bottom: 10px;  
1055   transition: all 0.2s linear;  
1056   -webkit-transition: all 0.2s linear;  
1057   font-weight: bold;  
1058   border-radius: 3px;  
1059   -webkit-appearance: none;  
1060   background: #fff;  
1061   font-family: 'Roboto';  
1062   -webkit-box-sizing:content-box;  
1063   box-sizing:content-box;  
1064   cursor: pointer;  
1065   line-height: 22px;  
1066 }  
1067 .options-cont textarea, .main-dialog textarea {  
1068   resize: vertical;  
1069 }  
1070 .options-cont input[type="submit"], .main-dialog input[type="submit"] {  
1071   color: #fff;  
1072   background: #2e85c7;  
1073   background-size: 30px 30px;  
1074   border: 0;  
1075   margin-left: 2px;  
1076   margin-bottom: 10px;  
1077   cursor: pointer;  
1078   text-transform: uppercase;  
1079   border: 2px solid #2e85c7;  
1080 }  
1081 }  
1082 .options-cont input[type="text"]:focus, .options-cont input[type="number"]:  
1083   focus, .options-cont textarea:focus, .options-cont select:focus,  
1084   .main-dialog input[type="text"]:focus, .main-dialog input[type="number"]:  
1085   focus, .main-dialog textarea:focus, .main-dialog select:focus {  
1086   border: 2px solid #2e85c7;  
1087 }  
1088 .options-cont input[type="text"]:read-only:focus, .options-cont input[type="  
1089   number"]:  
1090   read-only:focus {  
1091   border: 2px solid #ccc;  
1092 }  
1093 .options-cont input[type="text"]:read-only:hover, .options-cont input[type="
```

```
number"] :read-only:hover ,  
1094 .main-dialog input[type="text"] :read-only:hover , .main-dialog input[type="  
1095 number"] :read-only:hover {  
1096   cursor: auto;  
1097 }  
1098 .options-cont input[type="submit"] :hover , .main-dialog input[type="submit"]  
1099   :hover {  
1100     color: #2e85c7;  
1101     background: #fff ;  
1102 }  
1103 .options-cont input[type="submit"] :disabled , .main-dialog input[type="submit"]  
1104   :disabled {  
1105     background-image: linear-gradient(135deg, rgba(255, 255, 255, .15) 25%,  
1106       transparent 25%,  
1107       transparent 50%, rgba(255, 255, 255, .15) 50%, rgba(255, 255, 255, .15)  
1108       75%,  
1109       transparent 75%, transparent);  
1110     background-size: 30px 30px;  
1111     animation: animate-stripes 1s linear infinite;  
1112     -webkit-animation: animate-stripes 1s linear infinite;  
1113 }  
1114 @keyframes animate-stripes {  
1115   0% {background-position: 0 0;} 100% {background-position: 60px 0;}  
1116 }  
1117 @-moz-keyframes animate-stripes {  
1118   0% {background-position: 0 0;} 100% {background-position: 60px 0;}  
1119 }  
1120 @-webkit-keyframes animate-stripes {  
1121   0% {background-position: 0 0;} 100% {background-position: 60px 0;}  
1122 }  
1123 .options-cont input:disabled , .options-cont textarea:disabled , .options-cont  
1124   select:disabled ,  
1125 .main-dialog input:disabled , .main-dialog textarea:disabled , .  
1126   main-dialog select:disabled {  
1127     opacity: 0.5;  
1128     pointer-events: none;  
1129 }  
1130 .options-cont ::placeholder , .options-cont ::-webkit-input-placeholder ,  
1131 .main-dialog ::placeholder , .main-dialog ::-webkit-input-placeholder {  
1132   color: #ccc;  
1133   font-weight: normal;  
1134   text-transform: none;  
1135 }  
1136 .options-cont label , .main-dialog label {  
1137   color: #12568a;  
1138   margin-bottom: 2px;  
1139   display: block;
```

```
1141     margin-left: 3px;
1142 }
1143
1144 .options-cont label span, .main-dialog label span {
1145   color: #555;
1146 }
1147
1148 .options-cont label.checkcont, .main-dialog label.checkcont {
1149   margin: 15px 0;
1150   font-size: 18px;
1151   padding-top: 3px;
1152   padding-bottom: 3px;
1153 }
1154
1155 .options-cont label.checkcont, .main-dialog label.checkcont {
1156   padding-top: 6px;
1157   color: #333;
1158 }
1159
1160 .checkcont {
1161   display: block;
1162   position: relative;
1163   padding-left: 35px;
1164   margin-bottom: 12px;
1165   cursor: pointer;
1166   font-size: 22px;
1167   -webkit-user-select: none;
1168   -moz-user-select: none;
1169   -ms-user-select: none;
1170   user-select: none;
1171   -webkit-user-drag: none;
1172   user-drag: none;
1173 }
1174
1175 .checkcont input {
1176   position: absolute;
1177   opacity: 0;
1178   cursor: pointer;
1179   height: 0;
1180   width: 0;
1181 }
1182
1183 .checkmark {
1184   position: absolute;
1185   top: 0;
1186   left: 0;
1187   height: 25px;
1188   width: 25px;
1189   background-color: #fff;
1190   border: 2px solid #ccc;
1191   border-radius: 3px;
1192 }
1193
1194 .checkcont:hover input ~ .checkmark {
1195   background-color: #ccc;
```

```

1196 }
1197
1198 .checkcont input:checked ~ .checkmark {
1199     background-color: #2e85c7;
1200 }
1201
1202 .checkmark:after {
1203     content: "";
1204     position: absolute;
1205     display: none;
1206 }
1207
1208 .checkcont input:checked ~ .checkmark:after {
1209     display: block;
1210 }
1211
1212 .checkcont .checkmark:after {
1213     left: 9px;
1214     top: 5px;
1215     width: 5px;
1216     height: 10px;
1217     border: solid white;
1218     border-width: 0 3px 3px 0;
1219     -webkit-transform: rotate(45deg);
1220     -ms-transform: rotate(45deg);
1221     transform: rotate(45deg);
1222 }
1223
1224 .float-input {
1225     display: flex;
1226     flex-flow: column-reverse;
1227     position: relative;
1228     margin-top: 5px;
1229 }
1230
1231 .float-select {
1232     display: flex;
1233     flex-flow: column-reverse;
1234     position: relative;
1235     margin-top: 5px;
1236 }
1237
1238 .no-float-input {
1239     position: relative;
1240     margin-top: 5px;
1241 }
1242
1243 label.float-label {
1244     pointer-events: none;
1245     position: absolute;
1246     top: -13px;
1247     margin-left: 12px!important;
1248     padding: 1px 8px;
1249     border-radius: 3px;
1250     color: #777!important;

```

```
1251     font-weight: bold;
1252     background: #eee;
1253 }
1254
1255 .float-select label.float-label {
1256   color: #777;
1257   font-weight: bold;
1258   background: #eee;
1259 }
1260
1261 label.float-label, .float-input input, .float-input textarea {
1262   transition: all 0.2s;
1263   touch-action: manipulation;
1264 }
1265 .float-input input:placeholder-shown + label.float-label, .float-input
1266   textarea:placeholder-shown + label.float-label {
1267   cursor: text;
1268   white-space: nowrap;
1269   overflow: hidden;
1270   text-overflow: ellipsis;
1271   transform-origin: left bottom;
1272   transform: translate(0, 1.7rem) scale(1.2);
1273   background: none;
1274 }
1275 .float-input input::-webkit-input-placeholder, .float-input input::placeholder
1276   , .float-input textarea::-webkit-input-placeholder, .float-input
1277   textarea::placeholder {
1278   opacity: 0;
1279   transition: inherit;
1280 }
1281
1282 .float-input input:focus::-webkit-input-placeholder, .float-input
1283   textarea:focus::-webkit-input-placeholder {
1284   opacity: 1;
1285 }
1286
1287 .float-input input:not(:placeholder-shown) + label.float-label,
1288 .float-input input:focus + label.float-label,
1289 .float-input textarea:not(:placeholder-shown) + label.float-label,
1290   .float-input textarea:focus + label.float-label,
1291   .float-input input:-webkit-autofill + label.float-label
1292   .float-input textarea:-webkit-autofill + label.float-label {
1293   transform: translate(0, 0) scale(1);
1294   cursor: pointer;
1295   background: #eee;
1296 }
1297
1298 .float-input input:not(:placeholder-shown) + label.float-label,
1299 .float-input input:focus + label.float-label,
1300   .float-input textarea:not(:placeholder-shown) + label.float-label,
1301   .float-input textarea:focus + label.float-label,
1302   .float-input input:-webkit-autofill + label.float-label,
1303   .float-input textarea:-webkit-autofill + label.float-label{
1304   color: #777!important;
```

```
1302     font-weight: bold ;
1303     background: #eee ;
1304 }
1305
1306 .options-cont button:not ([disabled]):hover {
1307   color: #2e85c7 ;
1308   background: #fff ;
1309 }
1310
1311 .options-cont button {
1312   color: #fff ;
1313   background: #2e85c7 ;
1314   background-size: 30px 30px ;
1315   border: 0 ;
1316   cursor: pointer ;
1317   text-transform: uppercase ;
1318   border: 2px solid #2e85c7 ;
1319   margin-top: 15px ;
1320 }
1321
1322 .options-cont button:disabled {
1323   background-image: linear-gradient(135deg, rgba(255, 255, 255, .15) 25%,
1324   transparent 25%,
1325   transparent 50%, rgba(255, 255, 255, .15) 50%, rgba(255, 255, 255, .15)
1326   75%,
1327   transparent 75%, transparent) ;
1328   background-size: 30px 30px ;
1329 }
1330
1331 #status-container {
1332   position: absolute;
1333   bottom: 0;
1334   left: 0;
1335   right: 0;
1336   background: #f1f3f4 ;
1337   padding: 5px 20px;
1338   font-size: 14px;
1339   line-height: 14px;
1340   color: #5f6368 ;
1341 }
1342
1343 #status {
1344   height: 14px;
1345   overflow: hidden;
1346   float: left ;
1347 }
1348
1349 #status-icons {
1350   float: right ;
1351 }
1352
1353 #status-icons img {
1354   height: 14px;
1355   width: 14px;
1356   display: block;
```



```
1355 float: left;
1356 -webkit-user-select: none;
1357 user-select: none;
1358 -webkit-user-drag: none;
1359 user-drag: none;
1360 cursor: pointer;
1361 }
1362
1363 #status-icons img:hover {
1364 opacity: 0.5;
1365 }
1366
1367 #status-icons #help-icon {
1368 opacity: 0.4;
1369 }
1370
1371 #status-icons #help-icon:hover {
1372 opacity: 0.2;
1373 }
1374
1375 #sandbox-stats-icon {
1376 height: 14px;
1377 overflow: hidden;
1378 float: left;
1379 width: 14px;
1380 background: #999;
1381 border-radius: 50%;
1382 margin-right: 10px;
1383 margin-top: -1px;
1384 cursor: pointer;
1385 }
1386
1387 #sandbox-stats-icon.ready {
1388 background: #26a65b;
1389 }
1390
1391 #sandbox-stats-icon.async-busy {
1392 background: #ffb40c;
1393 }
1394
1395 #sandbox-stats-icon.busy {
1396 background: #cf000f;
1397 }
1398
1399 #sandbox-stats-icon:active {
1400 opacity: 0.7;
1401 }
1402
1403 #sandbox-stats-popup {
1404 position: absolute;
1405 display: block;
1406 border-radius: 5px;
1407 width: 245px;
1408 bottom: 35px;
1409 left: 10px;
```



```
1410    background: #fffff ;
1411    border: 1px solid #f7df1e ;
1412    border-bottom: 1px solid #f7df1e ;
1413 }
1414
1415 #sandbox-stats-popup .popup-triangle {
1416   position: absolute;
1417   width: 0;
1418   height: 0;
1419   border-style: solid;
1420   border-width: 10px 10px 0 10px;
1421   border-color: #f7df1e transparent transparent transparent;
1422   margin-left: 6px;
1423   bottom: -11px;
1424 }
1425
1426 #sandbox-stats-header {
1427   margin: 5px 10px;
1428   margin-bottom: 0px;
1429   color: #666;
1430   padding-bottom: 3px;
1431   border-bottom: 1px solid #f7df1e;
1432   font-weight: 300;
1433   font-size: 14px;
1434   user-select: none;
1435   -webkit-user-drag: none;
1436   user-drag: none;
1437   white-space: nowrap;
1438   overflow: hidden;
1439   text-overflow: ellipsis;
1440 }
1441
1442 .sandbox-stats-cont {
1443   position: relative;
1444   padding: 8px 10px;
1445   padding-right: 100px;
1446   color: #666;
1447 }
1448
1449 .sandbox-stats-val {
1450   position: absolute;
1451   right: 10px;
1452   top: 5px;
1453   width: 45px;
1454   text-align: center;
1455   background: #999;
1456   color: #fff;
1457   padding: 3px;
1458   border-radius: 3px;
1459 }
1460
1461 #command-window-messages span.eval-code {
1462   font-weight: bold;
1463   text-decoration: underline;
1464   cursor: pointer;
```

```

1465 }
1466
1467 #command-window-messages span.open-editor {
1468   color: #12568a;
1469   text-decoration: underline;
1470   cursor: pointer;
1471 }
1472
1473 @media only screen and (max-width: 900px) {
1474   .page-cont.wide {
1475     width: 460px;
1476   }
1477
1478   #main-menu li img {
1479     display: none;
1480   }
1481 }
1482
1483 @media only screen and (max-width: 780px) {
1484   #main-menu li img {
1485     display: block;
1486     margin-right: 0;
1487   }
1488
1489   #main-menu li str {
1490     display: none;
1491   }
1492 }
```

Listing 20 - main.css

```

1 body {
2   height: 100vh;
3 }
4
5 .graph.prdc-svg-viewer {
6   height: 100%;
7   width: 100%;
8 }
9
10 .graph.prdc-svg-viewer iframe {
11   height: 100%;
12   width: 100% !important;
13   max-width: 100% !important;
14 }
15
16 .graph.prdc-svg-viewer svg {
17   height: 100%;
18   width: 100%
19 }
20
21 .graph.node rect,
22 .graph.node circle,
23 .graph.node ellipse,
24 .graph.node polygon,
25 .graph.node path,
```

```

26 .graph .actor ,
27 .graph .actor-line {
28   fill: #73add938 !important;
29   stroke: #2e85c7 !important;
30 }
```

Listing 21 - mermaid-graph.css

```

1 .cm-s-notepadpp span.cm-comment { color: #008000; }
2 .cm-s-notepadpp span.cm-keyword { line-height: 1em; font-weight: bold; color:
  #0000ff; }
3 .cm-s-notepadpp span.cm-string { color: #808080; }
4 .cm-s-notepadpp span.cm-builtin { line-height: 1em; font-weight: bold; color:
  #804000; }
5 .cm-s-notepadpp span.cm-special { line-height: 1em; font-weight: bold; color:
  #0aa; }
6 .cm-s-notepadpp span.cm-variable { color: black; }
7 .cm-s-notepadpp span.cm-number, .cm-s-notepadpp span.cm-atom { color: #ff8000;
  }
8 .cm-s-notepadpp span.cm-meta { color: #555; }
9 .cm-s-notepadpp span.cm-link { color: #3a3; }
10
11 .cm-s-notepadpp .CodeMirror-activeline-background { background: #e8e8ff; }
12 .cm-s-notepadpp .CodeMirror-matchingbracket { outline: 1px solid grey;
  color: black !important; }
```

Listing 22 - notepadapp-theme.css

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
  abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
  strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
  label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas
  , details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby
  , section, summary, time, mark, audio, video { border: 0; font-size: 100%;
  font: inherit; vertical-align: baseline; margin: 0; padding: 0; } article, aside,
  details, figcaption, figure, footer, header, hgroup, menu, nav, section {
  display: block; } body { line-height: 1; } ol, ul { list-style: none; } blockquote, q {
  quotes: none; } blockquote::before, blockquote::after, q::before, q::after {
  content: none; } table { border-collapse: collapse; border-spacing: 0; }
2
3 html {
4   overflow: hidden;
5 }
6
7 :focus {
8   outline: 0;
9 }
10
11 div {
12   z-index: 2;
13   box-sizing: border-box;
14 }
15
16 section {
17   position: relative;
18   z-index: 2;
```

```
19 }
20
21 body {
22   font-family: 'Roboto', Arial;
23   background: #fff;
24   height: 100vh;
25   width: 100vw;
26   overflow: hidden;
27   border-top: 1px solid #f7df1e;
28 }
29
30 lang {
31   display: none;
32 }
33
34 .dragging webview {
35   pointer-events: none;
36 }
37
38 .clear {
39   clear: both;
40 }
41
42 .float-left {
43   float: left;
44 }
45
46 .float-right {
47   float: right;
48 }
49
50 #left-panel-cont {
51   display: flex;
52   flex-direction: column;
53   padding: 10px;
54   justify-content: center;
55   align-items: center;
56   height: 100%;
57   width: 100%;
58 }
59
60 #right {
61   border: 1px solid #eee;
62   position: relative;
63   border-radius: 3px;
64   margin: 5px;
65   overflow: hidden;
66 }
67
68 #webview-wrap {
69   position: relative;
70   width: 100%;
71   box-shadow: 0 0 10px 0 #0000000d;
72   overflow: hidden;
73 }
```



```
74
75 #preview {
76   position: absolute;
77   top: 0;
78   left: 0;
79   width: 1920px;
80   height: 1080px;
81   border: 0;
82   background: #eee;
83   transform-origin: 0 0;
84 }
85
86 .tabs {
87   font-family: 'Roboto', Arial;
88   border-radius: 0;
89   padding-right: 65px;
90   background: #eee;
91   font-size: 16px;
92   padding-top: 5px;
93   height: 43px;
94   margin-top: 5px;
95 }
96
97 .tab-saved {
98   flex-grow: 0;
99   flex-shrink: 0;
100  position: relative;
101  width: 5px;
102  height: 5px;
103  border-radius: 50%;
104  background: transparent;
105  top: 5px;
106  right: 5px;
107 }
108
109 .changed .tab-saved {
110   background: #2e85c7;
111 }
112
113 .tabs .tab .tab-close {
114   display: none;
115 }
116
117 #tab-save {
118   position: absolute;
119   top: 5px;
120   right: 35px;
121   height: 30px;
122   background: transparent;
123   padding: 5px;
124   border-radius: 15px;
125   transition: all .2s linear;
126   cursor: pointer;
127 }
128
```

```
129 #tab-save:hover {  
130     background: #ccc;  
131 }  
132  
133 #editor-more-icon {  
134     position: absolute;  
135     top: 5px;  
136     right: 5px;  
137     height: 30px;  
138     background: transparent;  
139     padding: 5px;  
140     border-radius: 15px;  
141     transition: all .2s linear;  
142     cursor: pointer;  
143     opacity: 0.7;  
144 }  
145  
146 #editor-more-icon:hover {  
147     background: #ccc;  
148 }  
149  
150 #editor-menu-container {  
151     background: #12568a;  
152     background: linear-gradient(to left, #2e85c7, #12568a);  
153     color: #fff;  
154     user-select: none;  
155     -webkit-user-drag: none;  
156     user-drag: none;  
157     -webkit-app-region: drag;  
158     border-top: 2px solid #f7df1e;  
159 }  
160  
161 #editor-menu li {  
162     float: left;  
163     padding: 5px 10px;  
164     display: inline-block;  
165     font-size: 16px;  
166     font-weight: 100;  
167     line-height: 20px;  
168     -webkit-transition: all .2s linear;  
169     transition: all .2s linear;  
170     cursor: pointer;  
171     -webkit-app-region: no-drag;  
172 }  
173  
174 #editor-menu li:hover {  
175     background: #fffff4 0;  
176 }  
177  
178 #editor-menu li img {  
179     float: left;  
180     width: 20px;  
181     height: 20px;  
182     margin-right: 5px;  
183     user-select: none;
```



```
184     -webkit-user-drag: none;  
185     user-drag: none;  
186 }  
187  
188 #editor-more-popup {  
189     position: absolute;  
190     display: block;  
191     border-radius: 5px;  
192     width: 245px;  
193     top: 77px;  
194     right: 10px;  
195     background: #ffffff;  
196     border: 1px solid #f7df1e;  
197     border-bottom: 1px solid #f7df1e;  
198 }  
199  
200 #editor-more-popup .popup-triangle {  
201     position: absolute;  
202     width: 0;  
203     height: 0;  
204     border-style: solid;  
205     border-width: 10px 10px 0 10px;  
206     border-color: #f7df1e transparent transparent transparent;  
207     top: -11px;  
208     right: 5px;  
209     transform: rotate(180deg);  
210 }  
211  
212 #editor-more-popup li {  
213     margin: 5px;  
214     border-radius: 5px;  
215     border: 1px solid #dddddd;  
216     color: #999;  
217     padding: 5px 10px;  
218     font-weight: 300;  
219     font-size: 18px;  
220     cursor: pointer;  
221     transition: background 0.2s linear;  
222     -webkit-transition: background 0.2s linear;  
223 }  
224  
225 #editor-more-popup li:hover {  
226     background: #f3f3f3;  
227 }  
228  
229 #editor-more-popup li img {  
230     float: left;  
231     width: 20px;  
232     height: 20px;  
233     margin-right: 10px;  
234     user-select: none;  
235     -webkit-user-drag: none;  
236     user-drag: none;  
237     margin-top: -1px;  
238     filter: invert(1);
```



```
239     opacity: 0.3;
240 }
241
242 #presentation-title {
243     padding: 5px;
244     padding-bottom: 10px;
245     font-size: 18px;
246     color: #999
247 }
248
249 #slide-controls {
250     padding: 5px;
251     padding-top: 10px;
252     font-size: 18px;
253     color: #999;
254 }
255
256 #slide-controls .button {
257     display: inline-block;
258     cursor: pointer;
259 }
260
261 #slide-controls input {
262     border: 1px solid #999;
263     border-radius: 5px;
264     width: 40px;
265     line-height: 20px;
266     font-size: 18px;
267     color: #999;
268     text-align: center;
269     -webkit-appearance: none;
270     margin: 0;
271 }
272
273 #slide-controls input::-webkit-outer-spin-button ,
274 #slide-controls input::-webkit-inner-spin-button {
275     -webkit-appearance: none;
276     margin: 0;
277 }
278
279 .panel::-webkit-scrollbar ,
280 .panel::-webkit-scrollbar-button ,
281 .panel::-webkit-scrollbar-track ,
282 .panel::-webkit-scrollbar-track-piece ,
283 .panel::-webkit-scrollbar-thumb ,
284 .panel::-webkit-scrollbar-corner ,
285 .panel::-webkit-resizer {
286     background: transparent;
287 }
288
289 .panel::-webkit-scrollbar {
290     width: 12px;
291     height: 12px;
292     -webkit-border-radius: 5px;
293     border-radius: 5px;
```



```
294    }
295
296    .panel::-webkit-scrollbar-track-piece {
297        -webkit-border-radius: 5px;
298        border-radius: 5px;
299    }
300
301    .panel::-webkit-scrollbar-thumb {
302        background: #ddd;
303        border-radius: 5px;
304        background-clip: content-box;
305        border: 2px solid transparent;
306    }
307
308    .panel::-webkit-scrollbar-button {
309        width:0;
310        height:0 ;
311    }
312
313    .horizontal-resizer {
314        width: 100%;
315        background-image: url(../img/hdrag.svg);
316        background-repeat: no-repeat;
317        background-position: center center;
318        background-size: auto 6px;
319        cursor: n-resize;
320        user-select: none;
321        -webkit-user-drag: none;
322        user-drag: none;
323        position: absolute;
324    }
325
326    .horizontal-resizer:after {
327        content: '';
328        border-bottom: 1px solid transparent;
329        position: absolute;
330        top: 4px;
331        left: 0px;
332        right: 0px;
333        pointer-events: none;
334    }
335
336    .vertical-resizer {
337        height: 100%;
338        background-image: url(../img/vdrag.svg);
339        background-repeat: no-repeat;
340        background-position: center center;
341        background-size: 6px auto;
342        cursor: w-resize;
343        user-select: none;
344        -webkit-user-drag: none;
345        user-drag: none;
346        position: absolute;
347    }
348
```



```
349 .vertical-resizer:after {
350   content: '';
351   border-right: 1px solid transparent;
352   position: absolute;
353   top: 0px;
354   bottom: 0px;
355   left: 4px;
356   pointer-events: none;
357 }
358
359 .vertical-resizer:hover:after, .horizontal-resizer:hover:after {
360   border-right: 1px solid #f7df1e;
361 }
362
363 .horizontal-resizer:hover:after {
364   border-bottom: 1px solid #f7df1e;
365 }
366
367 .panel-container {
368   height: 100%;
369 }
370
371 .cell-padding {
372   padding: 5px;
373   height: 100%;
374   width: 100%;
375   position: relative;
376 }
377
378 #left-panel {
379   height: 100%;
380   width: 60%;
381   position: absolute;
382   left: 0;
383   top: 0;
384 }
385
386 #right-panel {
387   height: 100%;
388   width: 40%;
389   position: absolute;
390   left: 60%;
391   top: 0;
392 }
393
394 .panel-title {
395   margin: 5px 10px;
396   margin-bottom: 0px;
397   color: #666;
398   padding-bottom: 3px;
399   border-bottom: 1px solid #f7df1e;
400   font-weight: 300;
401   font-size: 14px;
402   user-select: none;
403   -webkit-user-drag: none;
```

```

404   user-drag: none;
405   white-space: nowrap;
406   overflow: hidden;
407   text-overflow: ellipsis;
408 }
409
410 #right-panel .panel-title {
411   padding-right: 155px;
412 }
413
414 .panel-container {
415   border: 1px solid #f1f3f4;
416   -webkit-transition: border .2s linear;
417   transition: border .2s linear;
418   position: relative;
419   border-radius: 3px;
420 }
421
422 .panel-container:hover {
423   border: 1px solid #f7df1e;
424 }
425
426 .panel {
427   overflow: auto;
428   position: absolute;
429   top: 23px;
430   left: 0px;
431   right: 0px;
432   bottom: 0px;
433 }
```

Listing 23 - presentation-editor.css

```

1 html , body , div , span , applet , object , iframe , h1 , h2 , h3 , h4 , h5 , h6 , p , blockquote , pre , a ,
  abbr , acronym , address , big , cite , code , del , dfn , em , img , ins , kbd , q , s , samp , small ,
  strike , strong , sub , sup , tt , var , b , u , i , center , dl , dt , dd , ol , ul , li , fieldset , form ,
  label , legend , table , caption , tbody , tfoot , thead , tr , th , td , article , aside , canvas ,
  details , embed , figure , figcaption , footer , header , hgroup , menu , nav , output , ruby ,
  section , summary , time , mark , audio , video { border:0 ; font-size:100% ;
  font:inherit ; vertical-align:baseline ; margin:0 ; padding:0 } article , aside ,
  details , figcaption , figurehtml , body , div , span , applet , object , iframe , h1 , h2 , h3 ,
  h4 , h5 , h6 , p , blockquote , pre , a , abbr , acronym , address , big , cite , code , del , dfn , em ,
  img , ins , kbd , q , s , samp , small , strike , strong , sub , sup , tt , var , b , u , i , center , dl , dt
  , dd , ol , ul , li , fieldset , form , label , legend , table , caption , tbody , tfoot , thead , tr ,
  th , td , article , aside , canvas , details , embed , figure , figcaption , footer , header ,
  hgroup , menu , nav , output , ruby , section , summary , time , mark , audio , video { border:0 ;
  font-size:100% ; font:inherit ; vertical-align:baseline ; margin:0 ; padding:0 }
  article , aside , details , figcaption , figure , footer , header , hgroup , menu , nav ,
  section { display:block } body { line-height:1 } ol , ul { list-style:none } blockquote ,
  q { quotes:none } blockquote:before , blockquote:after , q:before , q:after {
  content:none } table { border-collapse:collapse ; border-spacing:0 }
```

2

3 html {

4 overflow: hidden;

5 }

6

```
7  :focus {
8    outline: 0;
9  }
10
11 div {
12   z-index: 2;
13   box-sizing: border-box;
14 }
15
16 .clear {
17   clear: both;
18 }
19
20 .float-left {
21   float: left;
22 }
23
24 .float-right {
25   float: right;
26 }
27
28 body {
29   font-family: Arial;
30   width: 100vw;
31   height: 100vh;
32   background: #333;
33   font-size: 24px;
34 }
35
36 table {
37   border-collapse: collapse;
38   text-align: center;
39   margin: 0 auto;
40 }
41
42 th,
43 td {
44   border: 1px solid #ccc;
45   padding: 0.5rem 0.75rem;
46 }
47
48 th {
49   background: #f5f5f5;
50   font-weight: bold;
51   padding: 0.75rem 0.75rem;
52 }
53
54 tbody tr:nth-child(even) {
55   background: #fafafa;
56 }
57
58 sup {
59   vertical-align: super;
60   font-size: smaller;
61 }
```



```
62
63 sub {
64   vertical-align: sub;
65   font-size: smaller;
66 }
67
68 b, strong {
69   font-weight: bold;
70 }
71
72 i, em {
73   font-style: italic;
74 }
75
76 u {
77   text-decoration: underline;
78 }
79
80 b i,
81 strong em,
82 b em,
83 strong i {
84   font-weight: bold;
85   font-style: italic;
86 }
87
88 .bold {
89   font-weight: bold;
90 }
91
92 .italic {
93   font-style: italic;
94 }
95
96 .underline {
97   text-decoration: underline;
98 }
99
100 .one_half {
101   width: 48%;
102 }
103
104 .one_third {
105   width: 30.66%;
106 }
107
108 .two_third {
109   width: 65.33%;
110 }
111
112 .one_half_center {
113   width: 48%;
114   margin: 0 auto;
115   position: relative;
116 }
```

```

117
118 .one-fourth {
119     width: 22%;
120     margin-right: 4%;
121     float: left;
122 }
123
124 .one-five {
125     width: 20%;
126     float: left;
127     position: relative;
128 }
129
130 .one_half, .one_third, .two_third, .one_fourth, .three_fourth {
131     float: left;
132     margin-right: 4%;
133     position: relative;
134 }
135
136 .last {
137     margin-right: 0!important;
138     clear: right;
139 }
140
141 img.center {
142     display: block;
143     margin: 0 auto;
144 }
145
146 #slides-cont {
147     background: #fff;
148     transform-origin: left top;
149     position: absolute;
150     top: 50%;
151     left: 50%;
152 }
153
154 slide {
155     display: none;
156     width: 100%;
157     height: 100%;
158     overflow: hidden;
159     background: #fff;
160     position: absolute;
161     top: 0;
162     left: 0;
163 }
164
165 slide.active {
166     display: block;
167 }
168
169 #slide-controls {
170     position: fixed;
171     top: 10px;

```

```
172 left: 50%;  
173 transform: translateX(-50%);  
174 font-size: 18px;  
175 color: #999;  
176 padding: 5px 10px;  
177 border-radius: 5px;  
178 z-index: 9999;  
179 background: rgba(0, 0, 0, .65);  
180 box-shadow: 0 0 10px 0 #0000000d;  
181 }  
182  
183 #slide-controls[hidden]{  
184 display:none;  
185 }  
186  
187 #slide-controls .button {  
188 display: inline-block;  
189 cursor: pointer;  
190 }  
191  
192 #slide-controls input {  
193 border: 1px solid #999;  
194 border-radius: 5px;  
195 width: 40px;  
196 line-height: 20px;  
197 font-size: 18px;  
198 color: #999;  
199 text-align: center;  
200 -webkit-appearance: none;  
201 margin: 0;  
202 background: transparent;  
203 }  
204  
205 #slide-controls input::-webkit-outer-spin-button,  
206 #slide-controls input::-webkit-inner-spin-button {  
207 -webkit-appearance: none;  
208 margin: 0;  
209 }  
210  
211 plot-json {  
212 width: max-content;  
213 display: block;  
214 margin: 0 auto;  
215 }  
216  
217 plot-json .js-plotly-plot, plot-json .plot-container, plot-json .svg-container  
218 {  
219 width: max-content;  
220 }  
221 @media print {  
222 html {  
223 overflow: auto;  
224 }  
225 }
```

```
226 body {
227   margin: 0;
228 }
229
230 #slides-cont {
231   transform: none !important;
232   position: relative;
233   top: 0;
234   left: 0;
235 }
236
237 slide {
238   display: block !important;
239   -webkit-print-color-adjust: exact;
240   position: relative;
241 }
242
243 slide:not(:last-child) {
244   page-break-after: always;
245   break-after: page;
246 }
247 }
248
249 .slide-in,
250 .slide-out {
251   animation-duration: 0.50s;
252   animation-fill-mode: both;
253   animation-timing-function: ease;
254 }
255
256 @keyframes fadeIn {
257   from {
258     opacity: 0;
259   }
260
261   to {
262     opacity: 1;
263   }
264 }
265
266 @keyframes fadeOut {
267   from {
268     opacity: 1;
269   }
270
271   to {
272     opacity: 0;
273   }
274 }
275
276 .fade-in {
277   animation-name: fadeIn;
278 }
279
280 .fade-out {
```



```
281     animation-name: fadeOut;  
282 }  
283  
284 @keyframes slideLeftIn {  
285   from {  
286     transform: translateX(100%);  
287   }  
288  
289   to {  
290     transform: none;  
291   }  
292 }  
293  
294 @keyframes slideLeftOut {  
295   from {  
296     transform: none;  
297   }  
298  
299   to {  
300     transform: translateX(-100%);  
301   }  
302 }  
303  
304 @keyframes slideRightIn {  
305   from {  
306     transform: translateX(-100%);  
307   }  
308  
309   to {  
310     transform: none;  
311   }  
312 }  
313  
314 @keyframes slideRightOut {  
315   from {  
316     transform: none;  
317   }  
318  
319   to {  
320     transform: translateX(100%);  
321   }  
322 }  
323  
324 @keyframes slideUpIn {  
325   from {  
326     transform: translateY(100%);  
327   }  
328  
329   to {  
330     transform: none;  
331   }  
332 }  
333  
334 @keyframes slideUpOut {  
335   from {
```

```
336     transform: none;  
337 }  
338  
339 to {  
340     transform: translateY(-100%);  
341 }  
342 }  
343  
344 @keyframes slideDownIn {  
345     from {  
346         transform: translateY(-100%);  
347     }  
348  
349     to {  
350         transform: none;  
351     }  
352 }  
353  
354 @keyframes slideDownOut {  
355     from {  
356         transform: none;  
357     }  
358  
359     to {  
360         transform: translateY(100%);  
361     }  
362 }  
363  
364 .slide-left-in {  
365     animation-name: slideLeftIn;  
366 }  
367  
368 .slide-left-out {  
369     animation-name: slideLeftOut;  
370 }  
371  
372 .slide-right-in {  
373     animation-name: slideRightIn;  
374 }  
375  
376 .slide-right-out {  
377     animation-name: slideRightOut;  
378 }  
379  
380 .slide-up-in {  
381     animation-name: slideUpIn;  
382 }  
383  
384 .slide-up-out {  
385     animation-name: slideUpOut;  
386 }  
387  
388 .slide-down-in {  
389     animation-name: slideDownIn;  
390 }
```



```
391
392 .slide-down-out {
393   animation-name: slideDownOut;
394 }
395
396 @keyframes zoomIn {
397   from {
398     transform: scale(.85);
399     opacity: 0;
400   }
401
402   to {
403     transform: none;
404     opacity: 1;
405   }
406 }
407
408 @keyframes zoomOut {
409   from {
410     transform: none;
411     opacity: 1;
412   }
413
414   to {
415     transform: scale(1.1);
416     opacity: 0;
417   }
418 }
419
420 .zoom-in {
421   animation-name: zoomIn;
422 }
423
424 .zoom-out {
425   animation-name: zoomOut;
426 }
427
428 @keyframes flipYIn {
429   from {
430     transform: rotateY(-90deg);
431     opacity: 0;
432   }
433
434   to {
435     transform: none;
436     opacity: 1;
437   }
438 }
439
440 @keyframes flipYOut {
441   from {
442     transform: none;
443     opacity: 1;
444   }
445 }
```

```
446     to {
447       transform: rotateY(90deg);
448       opacity: 0;
449     }
450   }
451
452   . flip-y-in {
453     animation-name: flipYIn;
454     transform-origin: center left;
455     backface-visibility: hidden;
456   }
457
458   . flip-y-out {
459     animation-name: flipYOut;
460     transform-origin: center right;
461     backface-visibility: hidden;
462   }
463
464   @keyframes flipXIn {
465     from {
466       transform: rotateX(90deg);
467       opacity: 0;
468     }
469
470     to {
471       transform: none;
472       opacity: 1;
473     }
474   }
475
476   @keyframes flipXOut {
477     from {
478       transform: none;
479       opacity: 1;
480     }
481
482     to {
483       transform: rotateX(-90deg);
484       opacity: 0;
485     }
486   }
487
488   . flip-x-in {
489     animation-name: flipXIn;
490     transform-origin: bottom center;
491     backface-visibility: hidden;
492   }
493
494   . flip-x-out {
495     animation-name: flipXOut;
496     transform-origin: top center;
497     backface-visibility: hidden;
498   }
499
500   . cube-common {
```

```
501     perspective: 1200px;  
502 }  
503  
504 @keyframes cubeLeftIn {  
505   from {  
506     transform: rotateY(90deg);  
507   }  
508  
509   to {  
510     transform: none;  
511   }  
512 }  
513  
514 @keyframes cubeLeftOut {  
515   from {  
516     transform: none;  
517   }  
518  
519   to {  
520     transform: rotateY(-90deg);  
521   }  
522 }  
523  
524 @keyframes cubeRightIn {  
525   from {  
526     transform: rotateY(-90deg);  
527   }  
528  
529   to {  
530     transform: none;  
531   }  
532 }  
533  
534 @keyframes cubeRightOut {  
535   from {  
536     transform: none;  
537   }  
538  
539   to {  
540     transform: rotateY(90deg);  
541   }  
542 }  
543  
544 @keyframes cubeUpIn {  
545   from {  
546     transform: rotateX(-90deg);  
547   }  
548  
549   to {  
550     transform: none;  
551   }  
552 }  
553  
554 @keyframes cubeUpOut {  
555   from {
```

```
556     transform: none;  
557 }  
558  
559 to {  
560     transform: rotateX(90deg);  
561 }  
562 }  
563  
564 @keyframes cubeDownIn {  
565     from {  
566         transform: rotateX(90deg);  
567     }  
568  
569     to {  
570         transform: none;  
571     }  
572 }  
573  
574 @keyframes cubeDownOut {  
575     from {  
576         transform: none;  
577     }  
578  
579     to {  
580         transform: rotateX(-90deg);  
581     }  
582 }  
583  
584 .cube-left-in {  
585     animation-name: cubeLeftIn;  
586     transform-origin: center right;  
587 }  
588  
589 .cube-left-out {  
590     animation-name: cubeLeftOut;  
591     transform-origin: center left;  
592 }  
593  
594 .cube-right-in {  
595     animation-name: cubeRightIn;  
596     transform-origin: center left;  
597 }  
598  
599 .cube-right-out {  
600     animation-name: cubeRightOut;  
601     transform-origin: center right;  
602 }  
603  
604 .cube-up-in {  
605     animation-name: cubeUpIn;  
606     transform-origin: bottom center;  
607 }  
608  
609 .cube-up-out {  
610     animation-name: cubeUpOut;
```



```
611     transform-origin: top center;  
612 }  
613  
614 .cube-down-in {  
615     animation-name: cubeDownIn;  
616     transform-origin: top center;  
617 }  
618  
619 .cube-down-out {  
620     animation-name: cubeDownOut;  
621     transform-origin: bottom center;  
622 }  
623  
624 .cube-left-in,  
625 .cube-left-out,  
626 .cube-right-in,  
627 .cube-right-out,  
628 .cube-up-in,  
629 .cube-up-out,  
630 .cube-down-in,  
631 .cube-down-out {  
632     perspective: 1200px;  
633 }  
634  
635 @keyframes coverIn {  
636     from {  
637         transform: translateX(100%);  
638     }  
639  
640     to {  
641         transform: none;  
642     }  
643 }  
644  
645 @keyframes coverOut {  
646     from {  
647         opacity: 1;  
648     }  
649  
650     to {  
651         opacity: 1;  
652     }  
653 }  
654  
655 @keyframes uncoverIn {  
656     from {  
657         opacity: 1;  
658     }  
659  
660     to {  
661         opacity: 1;  
662     }  
663 }  
664  
665 @keyframes uncoverOut {
```

```
666  from {
667    transform: none;
668  }
669
670  to {
671    transform: translateX(-100%);
672  }
673 }
674
675 .cover-in {
676   animation-name: coverIn;
677   z-index: 3;
678 }
679
680 .cover-out {
681   animation-name: coverOut;
682   z-index: 1;
683 }
684
685 .uncover-in {
686   animation-name: uncoverIn;
687   z-index: 1;
688 }
689
690 .uncover-out {
691   animation-name: uncoverOut;
692   z-index: 3;
693 }
694
695 .cover-left-in {
696   animation-name: coverIn;
697   z-index: 3;
698 }
699 .cover-left-out {
700   animation-name: coverOut;
701   z-index: 1;
702 }
703
704 .uncover-left-in {
705   animation-name: uncoverIn;
706   z-index: 1;
707 }
708 .uncover-left-out {
709   animation-name: uncoverOut;
710   z-index: 3;
711 }
712
713 @keyframes coverRightIn {
714   from {
715     transform: translateX(-100%)
716   }
717
718   to {
719     transform: none
720   }
```



```
721    }
722
723  @keyframes coverUpIn {
724    from {
725      transform: translateY(100%)
726    }
727
728    to {
729      transform: none
730    }
731  }
732
733  @keyframes coverDownIn {
734    from {
735      transform: translateY(-100%)
736    }
737
738    to {
739      transform: none
740    }
741  }
742
743  @keyframes uncoverRightOut {
744    from {
745      transform: none
746    }
747
748    to {
749      transform: translateX(100%)
750    }
751  }
752
753  @keyframes uncoverUpOut {
754    from {
755      transform: none
756    }
757
758    to {
759      transform: translateY(-100%)
760    }
761  }
762
763  @keyframes uncoverDownOut {
764    from {
765      transform: none
766    }
767
768    to {
769      transform: translateY(100%)
770    }
771  }
772
773  .cover-right-in {
774    animation-name: coverRightIn;
775    z-index: 3;
```

```
776 }
777
778 .cover-right-out {
779   animation-name: coverOut;
780   z-index: 1;
781 }
782
783 .cover-up-in {
784   animation-name: coverUpIn;
785   z-index: 3;
786 }
787
788 .cover-up-out {
789   animation-name: coverOut;
790   z-index: 1;
791 }
792
793 .cover-down-in {
794   animation-name: coverDownIn;
795   z-index: 3;
796 }
797
798 .cover-down-out {
799   animation-name: coverOut;
800   z-index: 1;
801 }
802
803 .uncover-right-in {
804   animation-name: uncoverIn;
805   z-index: 1;
806 }
807
808 .uncover-right-out {
809   animation-name: uncoverRightOut;
810   z-index: 3;
811 }
812
813 .uncover-up-in {
814   animation-name: uncoverIn;
815   z-index: 1;
816 }
817
818 .uncover-up-out {
819   animation-name: uncoverUpOut;
820   z-index: 3;
821 }
822
823 .uncover-down-in {
824   animation-name: uncoverIn;
825   z-index: 1;
826 }
827
828 .uncover-down-out {
829   animation-name: uncoverDownOut;
830   z-index: 3;
```

```
831 }
832
833 @keyframes pushLeftIn {
834   from {
835     transform: translateX(100%);
836   }
837
838   to {
839     transform: none;
840   }
841 }
842
843 @keyframes pushLeftOut {
844   from {
845     transform: none;
846   }
847
848   to {
849     transform: translateX(-100%);
850   }
851 }
852
853 @keyframes pushRightIn {
854   from {
855     transform: translateX(-100%);
856   }
857
858   to {
859     transform: none;
860   }
861 }
862
863 @keyframes pushRightOut {
864   from {
865     transform: none;
866   }
867
868   to {
869     transform: translateX(100%);
870   }
871 }
872
873 @keyframes pushUpIn {
874   from {
875     transform: translateY(100%);
876   }
877
878   to {
879     transform: none;
880   }
881 }
882
883 @keyframes pushUpOut {
884   from {
885     transform: none;
```

```
886     }
887
888     to {
889       transform: translateY(-100%);
890     }
891   }
892
893 @keyframes pushDownIn {
894   from {
895     transform: translateY(-100%);
896   }
897
898   to {
899     transform: none;
900   }
901 }
902
903 @keyframes pushDownOut {
904   from {
905     transform: none;
906   }
907
908   to {
909     transform: translateY(100%);
910   }
911 }
912
913 .push-left-in {
914   animation-name: pushLeftIn;
915 }
916
917 .push-left-out {
918   animation-name: pushLeftOut;
919 }
920
921 .push-right-in {
922   animation-name: pushRightIn;
923 }
924
925 .push-right-out {
926   animation-name: pushRightOut;
927 }
928
929 .push-up-in {
930   animation-name: pushUpIn;
931 }
932
933 .push-up-out {
934   animation-name: pushUpOut;
935 }
936
937 .push-down-in {
938   animation-name: pushDownIn;
939 }
940
```

```
941 .push-down-out {  
942   animation-name: pushDownOut;  
943 }
```

Listing 24 - presentation.css

```
1 .prdc-svg-viewer {  
2   width: 100%;  
3   height: auto;  
4   border-top: 1px solid #ddd;  
5   overflow: hidden;  
6   position: relative;  
7 }  
8  
9 .data-card .prdc-svg-viewer {  
10   background: #fff;  
11 }  
12  
13 .prdc-svg-viewer iframe {  
14   width: 100%;  
15   height: 100%;  
16 }  
17  
18 .prdc-svg-viewer.autoheight iframe {  
19   height: auto;  
20 }  
21  
22 .prdc-svg-viewer .prdc-svg-viewer-controls {  
23   position: absolute;  
24   top: 5px;  
25   right: 20px;  
26   user-select: none;  
27 }  
28  
29 body:not(.touchscreen) .prdc-svg-viewer .prdc-svg-viewer-controls > div:hover {  
30   border: 1px solid #2e85c7;  
31 }  
32  
33 .prdc-svg-viewer .prdc-svg-viewer-controls > div:active {  
34   opacity: 0.5;  
35 }  
36  
37 .prdc-svg-viewer .prdc-svg-viewer-controls > div {  
38   display: inline-block;  
39   height: 36px;  
40   padding: 5px 10px;  
41   margin: 5px 2px;  
42   border-radius: 3px;  
43   color: #ccc;  
44   vertical-align: top;  
45   cursor: pointer;  
46   border: 1px solid #ddd;  
47   transition: border .2s linear;  
48   width: 36px;  
49   background-color: rgba(255,255,255,0.9);
```

```
50    background-position: center ;
51    background-repeat: no-repeat ;
52    background-size: contain ;
53 }
54
55 .prdc-svg-viewer .prdc-svg-viewer-controls .prdc-svg-viewer-control-zoom-plus
56 {
57   background-image: url('../img/zoom-plus.svg') ;
58 }
59
60 .prdc-svg-viewer .prdc-svg-viewer-controls .prdc-svg-viewer-control-zoom-minus
61 {
62   background-image: url('../img/zoom-minus.svg') ;
63 }
64
65 .prdc-svg-viewer .prdc-svg-viewer-controls .prdc-svg-viewer-control-autoscale
66 {
67   background-image: url('../img/autoscale.svg') ;
68 }
69
70 .prdc-svg-viewer .prdc-svg-viewer-controls .prdc-svg-viewer-control-reset {
71   background-image: url('../img/reset2.svg') ;
72 }
73
74 .prdc-svg-viewer .prdc-svg-viewer-controls .prdc-svg-viewer-control-layers {
75   background-image: url('../img/layers.svg') ;
76 }
77
78 .prdc-svg-viewer-no-layers .prdc-svg-viewer-controls .
79   prdc-svg-viewer-control-layers {
80     display: none;
81   }
82
83 .prdc-svg-viewer-layers-popup {
84   position: absolute;
85   display: none;
86   border-radius: 5px;
87   width: 400px;
88   top: 56px;
89   right: 20px;
90   background: #ffffffff5 ;
91   border: 1px solid #ddd;
92   color: #fff ;
93   pointer-events: all ;
94   cursor: pointer;
95   min-height: 100px;
96 }
97
98 .prdc-svg-viewer-layers-popup .popup-triangle {
99   position: absolute;
100  width: 0;
101  height: 0;
102  border-style: solid;
103  border-width: 0 10px 10px 10px;
104  border-color: transparent transparent #ccc transparent ;
```



```
101 margin-right: 5px;  
102 top: -11px;  
103 right: 3px;  
104 }  
105  
106 .prdc-svg-viewer-layers-popup-cont {  
107   overflow-y: auto;  
108   max-height: 400px;  
109 }
```

Listing 25 - svg-viewer.css

```
1 .tabs {  
2   box-sizing: border-box;  
3   position: relative;  
4   font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,  
5     Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "  
6     Segoe UI Symbol";  
7   font-size: 12px;  
8   height: 46px;  
9   padding: 8px 3px 4px 3px;  
10  background: #dee1e6;  
11  border-radius: 5px 5px 0 0;  
12  overflow: hidden;  
13 }  
14 .tabs * {  
15   box-sizing: inherit;  
16   font: inherit;  
17 }  
18 .tabs .tabs-content {  
19   position: relative;  
20   width: 100%;  
21   height: 100%;  
22 }  
23 .tabs .tab {  
24   position: absolute;  
25   left: 0;  
26   height: 36px;  
27   width: 240px;  
28   border: 0;  
29   margin: 0;  
30   z-index: 1;  
31   pointer-events: none;  
32 }  
33 .tabs .tab,  
34 .tabs .tab * {  
35   user-select: none;  
36   cursor: pointer;  
37 }  
38 .tabs .tab .tab-dividers {  
39   position: absolute;  
40   top: 7px;  
41   bottom: 7px;  
42   left: var(--tab-content-margin);  
43   right: var(--tab-content-margin);  
44 }
```

```
43 .tabs .tab .tab-dividers ,  
44 .tabs .tab .tab-dividers::before ,  
45 .tabs .tab .tab-dividers::after {  
46   pointer-events: none;  
47 }  
48 .tabs .tab .tab-dividers::before ,  
49 .tabs .tab .tab-dividers::after {  
50   content: "";  
51   display: block;  
52   position: absolute;  
53   top: 0;  
54   bottom: 0;  
55   width: 1px;  
56   background: #a9adb0;  
57   opacity: 1;  
58   transition: opacity 0.2s ease;  
59 }  
60 .tabs .tab .tab-dividers::before {  
61   left: 0;  
62 }  
63 .tabs .tab .tab-dividers::after {  
64   right: 0;  
65 }  
66 .tabs .tab:first-child .tab-dividers::before ,  
67 .tabs .tab:last-child .tab-dividers::after {  
68   opacity: 0;  
69 }  
70 .tabs .tab .tab-background {  
71   position: absolute;  
72   top: 0;  
73   left: 0;  
74   width: 100%;  
75   height: 100%;  
76   overflow: hidden;  
77   pointer-events: none;  
78 }  
79 .tabs .tab .tab-background > svg {  
80   width: 100%;  
81   height: 100%;  
82 }  
83 .tabs .tab .tab-background > svg .tab-geometry {  
84   fill: #f4f5f6;  
85 }  
86 .tabs .tab[active] {  
87   z-index: 5;  
88 }  
89 .tabs .tab[active] .tab-background > svg .tab-geometry {  
90   fill: #fff;  
91 }  
92 .tabs .tab:not([active]) .tab-background {  
93   transition: opacity 0.2s ease;  
94   opacity: 0;  
95 }  
96 @media (hover: hover) {  
97   .tabs .tab:not([active]):hover {
```



```
98     z-index: 2;
99 }
100 .tabs .tab:not([active]):hover .tab-background {
101   opacity: 1;
102 }
103 }
104 .tabs .tab.tab-was-just-added {
105   top: 10px;
106   animation: tab-was-just-added 120ms forwards ease-in-out;
107 }
108 .tabs .tab .tab-content {
109   position: absolute;
110   display: flex;
111   top: 0;
112   bottom: 0;
113   left: var(--tab-content-margin);
114   right: var(--tab-content-margin);
115   padding: 9px 8px;
116   border-top-left-radius: 8px;
117   border-top-right-radius: 8px;
118   overflow: hidden;
119   pointer-events: all;
120 }
121 .tabs .tab[is-mini] .tab-content {
122   padding-left: 2px;
123   padding-right: 2px;
124 }
125 .tabs .tab .tab-favicon {
126   position: relative;
127   flex-shrink: 0;
128   flex-grow: 0;
129   height: 16px;
130   width: 16px;
131   background-size: 16px;
132   margin-left: 4px;
133 }
134 .tabs .tab[is-small] .tab-favicon {
135   margin-left: 0;
136 }
137 .tabs .tab[is-mini]:not([active]) .tab-favicon {
138   margin-left: auto;
139   margin-right: auto;
140 }
141 .tabs .tab[is-mini][active] .tab-favicon {
142   display: none;
143 }
144 .tabs .tab .tab-title {
145   flex: 1;
146   vertical-align: top;
147   overflow: hidden;
148   white-space: nowrap;
149   margin-left: 4px;
150   color: #5f6368;
151   -webkit-mask-image: linear-gradient(90deg, #000 0%, #000 calc(100% - 24px), transparent);
```



```
152     mask-image: linear-gradient(90deg, #000 0%, #000 calc(100% - 24px) ,  
153         transparent);  
154 }  
155 .tabs .tab[is-small] .tab-title {  
156     margin-left: 0;  
157 }  
158 .tabs .tab .tab-favicon + .tab-title ,  
159 .tabs .tab[is-small] .tab-favicon + .tab-title {  
160     margin-left: 8px;  
161 }  
162 .tabs .tab[is-smaller] .tab-favicon + .tab-title ,  
163 .tabs .tab[is-mini] .tab-title {  
164     display: none;  
165 }  
166 .tabs .tab[active] .tab-title {  
167     color: #45474a;  
168 }  
169 .tabs .tab .tab-drag-handle {  
170     position: absolute;  
171     top: 0;  
172     bottom: 0;  
173     right: 0;  
174     left: 0;  
175     border-top-left-radius: 8px;  
176     border-top-right-radius: 8px;  
177 }  
178 .tabs .tab .tab-close {  
179     flex-grow: 0;  
180     flex-shrink: 0;  
181     position: relative;  
182     width: 16px;  
183     height: 16px;  
184     border-radius: 50%;  
185     background-image: url("data:image/svg+xml;utf8,<svg xmlns='http://www.w3.org  
186         /2000/svg' viewBox='0 0 8 8'><path stroke='rgba(0, 0, 0, .65)'  
187             stroke-linecap='square' stroke-width='1.5' d='M0 0 L8 8 M8 0 L0 8'></  
188             path></svg>");  
189     background-position: center center;  
190     background-repeat: no-repeat;  
191     background-size: 8px 8px;  
192 }  
193 @media (hover: hover) {  
194     .tabs .tab .tab-close:hover {  
195         background-color: #e8eaed ;  
196     }  
197     .tabs .tab .tab-close:hover:active {  
198         background-color: #dadce0 ;  
199     }  
200 }  
201 }  
202 @media (hover: hover) {
```

```
203   .tabs .tab:not([active]) .tab-close:not(:hover):not(:active) {  
204     opacity: 0.8;  
205   }  
206 }  
207 .tabs .tab[is-smaller] .tab-close {  
208   margin-left: auto;  
209 }  
210 .tabs .tab[is-mini]:not([active]) .tab-close {  
211   display: none;  
212 }  
213 .tabs .tab[is-mini][active] .tab-close {  
214   margin-left: auto;  
215   margin-right: auto;  
216 }  
217 @-moz-keyframes tab-was-just-added {  
218   to {  
219     top: 0;  
220   }  
221 }  
222 @-webkit-keyframes tab-was-just-added {  
223   to {  
224     top: 0;  
225   }  
226 }  
227 @-o-keyframes tab-was-just-added {  
228   to {  
229     top: 0;  
230   }  
231 }  
232 @keyframes tab-was-just-added {  
233   to {  
234     top: 0;  
235   }  
236 }  
237 .tabs.tabs-is-sorting .tab:not(.tab-is-dragging),  
238 .tabs:not(.tabs-is-sorting) .tab.tab-was-just-dragged {  
239   transition: transform 120ms ease-in-out;  
240 }  
241 .tabs .tabs-bottom-bar {  
242   position: absolute;  
243   bottom: 0;  
244   height: 4px;  
245   left: 0;  
246   width: 100%;  
247   background: #fff;  
248   z-index: 10;  
249 }  
250 .tabs-optional-shadow-below-bottom-bar {  
251   position: relative;  
252   height: 1px;  
253   width: 100%;  
254   background-image: url("data:image/svg+xml;utf8,<svg xmlns='http://www.w3.org  
255 /2000/svg' width='1' height='1' viewBox='0 0 1 1'><rect x='0' y='0'  
width='1' height='1' fill='rgba(0, 0, 0, .17)'></rect></svg>");  
background-size: 1px 1px;
```

```

256   background-repeat: repeat-x;
257   background-position: 0% 0%;
258 }
259 @media only screen and (-webkit-min-device-pixel-ratio: 2), only screen and (
260   min--moz-device-pixel-ratio: 2), only screen and (
261   o-min-device-pixel-ratio: 2/1), only screen and (min-device-pixel-ratio:
262   2), only screen and (min-resolution: 192dpi), only screen and (
263   min-resolution: 2dppx) {
264   .tabs-optional-shadow-below-bottom-bar {
265     background-image: url("data:image/svg+xml;utf8,<svg xmlns='http://www.w3.
266     org/2000/svg' width='2' height='2' viewBox='0 0 2 2'><rect x='0' y='0'
267       width='2' height='1' fill='rgba(0, 0, 0, .27)'></rect></svg>");
268   }
269 }
```

Listing 26 - tabs.css

```

1 #command-window-options {
2   position: absolute;
3   top: -1px;
4   right: 8px;
5 }
6
7 #command-window-options.options .options-right {
8   float: right;
9 }
10
11 #command-window-options.options i.settings {
12   background: url(..../img/settings.svg) no-repeat center;
13 }
14
15 #command-window-options.options i.timestamp {
16   background: url(..../img/timestamp.svg) no-repeat center;
17 }
18
19 #command-window-options.options i.autoscroll {
20   background: url(..../img/autoscroll.svg) no-repeat center;
21 }
22
23 #command-window-options.options i.clear {
24   background: url(..../img/clear.svg) no-repeat center;
25 }
26
27 #command-window-options.options i.log {
28   background: url(..../img/save-log.svg) no-repeat center;
29 }
30
31 #command-window-options.options i.to-bottom {
32   background: url(..../img/to-bottom.svg) no-repeat center;
33 }
34
35 #command-window-options.options i {
36   width: 16px;
37   height: 18px;
38   display: block;
39   background-size: 16px !important;
```

```
40 float: left;
41 clear: none;
42 padding: 3px 5px;
43 opacity: 0.3;
44 user-select: none;
45 -webkit-user-drag: none;
46 }
47
48 #command-window-options .options i:hover {
49   opacity: 1;
50   cursor: pointer;
51 }
52
53 #command-window-options .options i.active {
54   opacity: 0.8;
55 }
56
57 #command-window-input-container {
58   padding: 10px;
59   padding-left: 25px;
60   position: relative;
61   background-image: url(../img/input.svg);
62   background-position: left 12px top 12px;
63   background-size: 15px;
64   background-repeat: no-repeat;
65   padding-right: 45px;
66 }
67
68 #command-window-input {
69   border: 0;
70   color: #333;
71   font-family: 'Roboto', Arial;
72   font-size: 16px;
73   font-weight: 300;
74   padding: 0;
75   width: 100%;
76   background: transparent;
77   resize: none;
78   overflow: hidden;
79   min-height: 34px;
80 }
81
82 #command-window-messages {
83   padding-top: 10px;
84 }
85
86 #command-window-messages .welcome-message .app-logo {
87   float: left;
88   height: 200px;
89   padding: 20px;
90   padding-left: 0px;
91   display: block;
92   user-select: none;
93   -webkit-user-drag: none;
94   user-drag: none;
```

```
95    }
96
97 #command-window-messages .welcome-message .company-logo {
98   float: left;
99   height: 136px;
100  padding: 52px 10px;
101  padding-right: 10px;
102  opacity: 0.1;
103  display: block;
104  user-select: none;
105  -webkit-user-drag: none;
106  user-drag: none;
107 }
108
109 #command-window-messages .welcome-message p {
110   display: block;
111   line-height: 22px;
112   padding: 10px 0px;
113   color: #333;
114   font-weight: 300;
115   opacity: 0.8;
116   max-width: 800px;
117 }
118
119 #command-window-messages .welcome-message span:not(.timestamp) {
120   font-weight: 500;
121 }
122
123 #command-window-messages .welcome-message, #command-window-messages .
124   welcome-message * {
125     white-space: normal!important;
126   }
127
128 #command-window-messages a {
129   font-weight: 500;
130   color: #000;
131 }
132
133 #command-window-messages.messages > div {
134   padding-bottom: 5px;
135   padding-top: 5px;
136   line-height: 22px;
137   padding-left: 130px;
138   padding-right: 10px;
139   position: relative;
140   white-space: pre-wrap;
141   min-height: 32px;
142 }
143
144 #command-window-messages.messages.no-timestamp > div {
145   padding-left: 25px;
146 }
147
148 #command-window-messages.messages div.system-in {
149   background: url(../img/system-in.svg) no-repeat 4px 8px;
```



```
149    background-size: 14px;
150  }
151
152 #command-window-messages.messages div.data-in {
153   background: url(../img/console-in.svg) no-repeat 4px 8px;
154   background-size: 14px;
155 }
156
157 #command-window-messages.messages div.data-out {
158   background: url(../img/console-out.svg) no-repeat 4px 8px;
159   background-size: 14px;
160 }
161
162 #command-window-messages.messages div:hover {
163   background-color: #f9f9f9;
164 }
165
166 #command-window-messages.messages.no-timestamp div span.timestamp {
167   display: none;
168 }
169
170 #command-window-messages.messages div:hover span.timestamp {
171   color: #2e85c7
172 }
173
174 #command-window-messages.messages div span.timestamp {
175   color: #999;
176   display: block;
177   position: absolute;
178   left: 25px;
179 }
180
181 #command-window-messages.messages div span.log {
182   color: #666;
183 }
184
185 #command-window-messages.messages div span.msg {
186   color: #12568a;
187 }
188
189 #command-window-messages.messages div span.data {
190   color: #17d838;
191 }
192
193 #command-window-messages.messages div span.error {
194   color: #cf000f;
195 }
196
197 #command-window-messages.messages div span.warn {
198   color: #dd8f00;
199 }
200 #command-window-messages.messages:hover div:hover {
201   filter: none;
202 }
203
```



```
204 #command-window-input-submit-cont {  
205   position: absolute;  
206   right: 10px;  
207   z-index: 3;  
208   background: #f7df1e;  
209   border-radius: 100%;  
210   width: 30px;  
211   height: 30px;  
212   cursor: pointer;  
213 }  
214  
215 #command-window-input-submit-cont:hover {  
216   opacity: 0.6;  
217 }  
218  
219 #command-window-input-submit-cont img {  
220   width: 18px;  
221   padding: 6px;  
222 }  
223  
224 /* #command-window-messages.messages:hover div {  
225   filter: grayscale(100%);  
226 } */  
227  
228 .save-log, .change-settings {  
229   margin-bottom: 0px;  
230 }  
231  
232 .history-cont {  
233   display: none;  
234   left: 0px;  
235   position: absolute;  
236   bottom: 0px;  
237   top: 23px;  
238   right: 0px;  
239   -webkit-backdrop-filter: blur(5px);  
240   backdrop-filter: blur(5px);  
241   background-color: rgba(255, 255, 255, 0.7);  
242   padding: 20px;  
243 }  
244  
245 .history-panel {  
246   top: 55px!important;  
247   left: 20px!important;  
248   right: 20px!important;  
249   bottom: 0px!important;  
250   padding-top: 10px;  
251   padding-bottom: 10px;  
252 }  
253  
254 .history-panel li {  
255   padding-bottom: 5px;  
256   padding-top: 5px;  
257   line-height: 22px;  
258   padding-left: 10px;
```

```
259 padding-right: 10px;  
260 margin: 0 10px;  
261 position: relative;  
262 white-space: pre-wrap;  
263 border-bottom: 1px solid #f7df1e;  
264 min-height: 22px;  
265 cursor: pointer;  
266 transition: all linear 0.3s;  
267 }  
268  
269 .history-panel li:after {  
270   transition: all linear 0.3s;  
271 }  
272  
273 .history-panel li:after {  
274   content: '';  
275   position: absolute;  
276   top: 3px;  
277   left: 0px;  
278   right: 0px;  
279   bottom: 3px;  
280   border: 1px solid transparent;  
281   border-radius: 3px;  
282   background: #fffff69;  
283   z-index: -1;  
284 }  
285  
286 .history-panel li.active:after {  
287   border: 1px solid #999;  
288 }  
289  
290 .history-panel li:not(.active):hover:after {  
291   border: 1px solid #999;  
292 }  
293  
294 .history-panel li:last-child {  
295   border-bottom: 1px solid transparent;  
296 }  
297  
298 .history-panel .history-empty {  
299   padding-bottom: 5px;  
300   padding-top: 5px;  
301   line-height: 22px;  
302   padding-left: 10px;  
303   padding-right: 10px;  
304   margin: 0 10px;  
305   position: relative;  
306   text-align: center;  
307 }  
308  
309 .options-panel {  
310   display: none;  
311   top: 0;  
312   left: 0;  
313   right: 0;
```



```
314     bottom: 0;
315     position: absolute;
316     -webkit-backdrop-filter: blur(5px);
317     backdrop-filter: blur(5px);
318     background-color: rgba(255, 255, 255, 0.7);
319     overflow-y: auto;
320   }
321
322   .options-cont {
323     margin: 0 auto;
324     width: 460px;
325     position: relative;
326     background: #fffff6b;
327     box-shadow: 0 0 10px #00000026;
328     border-radius: 10px;
329     padding: 20px;
330     padding-bottom: 20px;
331     color: #777;
332     margin-bottom: 20px;
333     margin-top: 20px;
334   }
335
336   .options-header, .page-header, .history-header {
337     position: relative;
338     border-bottom: 1px solid #666;
339     margin-bottom: 25px;
340     padding-bottom: 10px;
341     user-select: none;
342     -webkit-user-drag: none;
343     user-drag: none;
344   }
345
346   .page-header {
347     margin-bottom: 0px;
348   }
349
350   .options-header span, .page-header span, .history-header span {
351     color: #666;
352     display: block;
353     font-weight: bold;
354     font-size: 24px;
355     border-radius: 3px;
356   }
357
358   .options-close, .page-close, .history-close {
359     opacity: 0.7;
360     position: absolute;
361     display: block;
362     top: 0px;
363     right: 0px;
364     cursor: pointer;
365   }
366
367   .options-close:hover, .page-close:hover, .history-close:hover {
368     opacity: 0.3;
```

```

369 }
370
371 @media only screen and (max-width: 900px) {
372     .page-cont {
373         width: 460px;
374     }
375 }
```

Listing 27 - terminal.css

```

1 html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
2 abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
3 strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
4 label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas
5 , details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby
6 , section, summary, time, mark, audio, video{ border:0; font-size:100%;}
7 font: inherit; vertical-align: baseline; margin:0; padding:0}article, aside,
8 details, figcaption, figure, footer, header, hgroup, menu, nav, section{
9 display:block}body{line-height:1}ol, ul{list-style:none}blockquote, q{
10 quotes:none}blockquote:before, blockquote:after, q:before, q:after{
11 content:none}table{border-collapse:collapse; border-spacing:0}
12
13 html {
14     overflow: hidden;
15 }
16
17 :focus {
18     outline: 0;
19 }
20
21 div {
22     z-index: 2;
23     box-sizing: border-box;
24 }
25
26
27 section {
28     position: relative;
29     z-index: 2;
30 }
31
32 body {
33     font-family: 'Roboto', Arial;
34     width: 100%;
35     background: #fff;
36 }
```

37 }

Listing 28 - three.css

```
1  html {
2      overflow: auto;
3      height: 100%;
4  }
5
6  body {
7      border-top: 1px solid #ccc;
8      padding-top: 5px;
9  }
10
11 sup {
12     vertical-align: super;
13     font-size: smaller;
14 }
15
16 sub {
17     vertical-align: sub;
18     font-size: smaller;
19 }
20
21 button.ui {
22     text-align: center;
23     text-transform: uppercase;
24     padding: 10px;
25     cursor: pointer;
26     background: #fff;
27     transition: background 0.2s linear, color 0.2s linear;
28     -webkit-transition: background 0.2s linear, color 0.2s linear;
29 }
30
31 button.ui:hover {
32     background: #eee;
33 }
34
35 button.ui.blue {
36     background: #2e85c7;
37     border-color: #2e85c7;
38     color: #fff;
39 }
40
41 button.ui.blue:hover {
42     background: #fff;
43     color: #2e85c7;
44 }
45
46 button.ui.red {
47     background: #cf000f;
48     border-color: #cf000f;
49     color: #fff;
50 }
51
52 button.ui.red:hover {
```



```
53    background: #fff ;
54    color: #cf000f ;
55  }
56
57 button.ui.green {
58   background: #26a65b ;
59   border-color: #26a65b ;
60   color: #fff ;
61 }
62
63 button.ui.green:hover {
64   background: #fff ;
65   color: #26a65b ;
66 }
67
68 label.ui {
69   display: block ;
70   font-size: 14px ;
71   margin-bottom: 3px ;
72   margin-left: 5px ;
73 }
74
75 input.ui.warning {
76   border-color: #fff800 !important ;
77 }
78
79 input.ui.error {
80   border-color: #f00 !important ;
81 }
82
83 select.ui, input.ui, button.ui, div.result.ui {
84   display: block ;
85   margin: 5px ;
86   margin-bottom: 5px ;
87   width: calc(100% - 10px) ;
88   padding: 8px ;
89   border: 1px solid #ccc ;
90   border-radius: 5px ;
91   font-size: 16px ;
92   box-sizing: border-box ;
93 }
94
95 li.ui {
96   text-align: center ;
97   padding: 10px ;
98   margin: 5px ;
99   border: 1px solid #ccc ;
100  border-radius: 5px ;
101  cursor: pointer ;
102  transition: background 0.2s linear, color 0.2s linear ;
103  -webkit-transition: background 0.2s linear, color 0.2s linear ;
104 }
105
106 li.ui:hover {
107   background: #eee ;
```

```
108 }
109 .ui.grid-cont {
110   margin: 5px;
111 }
112 }
113 .ui.one_half, .ui.one_third, .ui.two_third, .ui.one_fourth, .ui.one_fifth {
114   float: left;
115   margin-right: 4%;
116   position: relative;
117   margin-left: 0;
118 }
119 }
120 .ui.one_half {
121   width: 48%;
122 }
123 }
124 .ui.one_third {
125   width: 30.66%;
126 }
127 }
128 .ui.one_fourth {
129   width: 22%;
130 }
131 }
132 .ui.one_fifth {
133   width: 16.8%;
134 }
135 }
136 .ui.last {
137   margin-right: 0 !important;
138   clear: right;
139 }
140 }
141 .ui.options-panel {
142   display: none;
143   top: 0;
144   left: 0;
145   right: 0;
146   bottom: 0;
147   position: absolute;
148   -webkit-backdrop-filter: blur(5px);
149   backdrop-filter: blur(5px);
150   background-color: rgba(255, 255, 255, 0.7);
151   overflow-y: auto;
152 }
153 }
154 .ui.options-cont {
155   margin: 0 auto;
156   width: 460px;
157   position: relative;
158   background: #fffff6b;
159   box-shadow: 0 0 10px #00000026;
160   border-radius: 10px;
161   padding: 20px;
```



```
163 padding-bottom: 20px;  
164 color: #777;  
165 margin-bottom: 20px;  
166 margin-top: 20px;  
167 }  
168  
169 .ui.options-header {  
170 position: relative;  
171 border-bottom: 1px solid #666;  
172 margin-bottom: 25px;  
173 padding-bottom: 10px;  
174 user-select: none;  
175 -webkit-user-drag: none;  
176 user-drag: none;  
177 }  
178  
179 .ui.options-header span {  
180 color: #666;  
181 display: block;  
182 font-weight: bold;  
183 font-size: 24px;  
184 border-radius: 3px;  
185 }  
186  
187 .ui.options-close {  
188 opacity: 0.7;  
189 position: absolute;  
190 display: block;  
191 top: 0px;  
192 right: 0px;  
193 cursor: pointer;  
194 }  
195  
196 .ui.options-close:hover {  
197 opacity: 0.3;  
198 }  
199  
200 .ui.float-input {  
201 display: flex;  
202 flex-flow: column-reverse;  
203 position: relative;  
204 margin-top: 6px;  
205 }  
206  
207 .ui.float-select {  
208 display: flex;  
209 flex-flow: column-reverse;  
210 position: relative;  
211 margin-top: 5px;  
212 }  
213  
214 .ui.no-float-input {  
215 position: relative;  
216 margin-top: 5px;  
217 }
```

```
218 .ui label.float-label {  
219   pointer-events: none;  
220   position: absolute;  
221   top: -11px;  
222   margin-left: 12px!important;  
223   padding: 1px 8px;  
224   border-radius: 3px;  
225   background: #fff;  
226 }  
227  
228 .ui.options-cont .ui label.float-label {  
229   color: #777!important;  
230   font-weight: bold;  
231   background: #eee;  
232 }  
233  
234 .ui.options-cont .ui.float-select label.float-label {  
235   color: #777;  
236   font-weight: bold;  
237   background: #eee;  
238 }  
239  
240 .ui label.float-label,  
241 .ui.float-input input,  
242 .ui.float-input textarea {  
243   transition: all 0.2s;  
244   touch-action: manipulation;  
245 }  
246  
247 .ui.float-input input:placeholder-shown + label.float-label,  
248 .ui.float-input textarea:placeholder-shown + label.float-label {  
249   cursor: text;  
250   white-space: nowrap;  
251   overflow: hidden;  
252   text-overflow: ellipsis;  
253   transform-origin: left bottom;  
254   transform: translate(0, 1.7rem) scale(1.2);  
255   background: none;  
256 }  
257  
258 .ui.float-input input::-webkit-input-placeholder,  
259 .ui.float-input input::placeholder,  
260 .ui.float-input textarea::-webkit-input-placeholder,  
261 .ui.float-input textarea::placeholder {  
262   opacity: 0;  
263   transition: inherit;  
264 }  
265  
266 .ui.float-input input:focus::-webkit-input-placeholder,  
267 .ui.float-input textarea:focus::-webkit-input-placeholder {  
268   opacity: 1;  
269 }  
270  
271 .ui.float-input input:not(:placeholder-shown) + label.float-label,
```

```
273 .ui.float-input input:focus + label.float-label,
274 .ui.float-input textarea:not(:placeholder-shown) + label.float-label,
275 .ui.float-input textarea:focus + label.float-label,
276 .ui.float-input input:-webkit-autofill + label.float-label
277 .ui.float-input textarea:-webkit-autofill + label.float-label {
278   transform: translate(0, 0) scale(1);
279   cursor: pointer;
280 }
281
282 .ui.options-cont .ui.float-input input:not(:placeholder-shown) + label.
283   float-label,
284 .ui.options-cont .ui.float-input input:focus + label.float-label,
285 .ui.options-cont .ui.float-input textarea:not(:placeholder-shown) + label.
286   float-label,
287 .ui.options-cont .ui.float-input textarea:focus + label.float-label,
288 .ui.options-cont .ui.float-input input:-webkit-autofill + label.float-label
289 .ui.options-cont .ui.float-input textarea:-webkit-autofill + label.float-label
290   {
291   background: #eee;
292 }
293
294 .ui.float-input input:not(:placeholder-shown) + label.float-label,
295 .ui.float-input input:focus + label.float-label,
296 .ui.float-input textarea:not(:placeholder-shown) + label.float-label,
297 .ui.float-input textarea:focus + label.float-label,
298 .ui.float-input input:-webkit-autofill + label.float-label,
299 .ui.float-input textarea:-webkit-autofill + label.float-label{
300   display: inline-block;
301 }
302
303 .ui.options-cont .ui.float-input input:not(:placeholder-shown) + label.
304   float-label,
305 .ui.options-cont .ui.float-input input:focus + label.float-label,
306 .ui.options-cont .ui.float-input textarea:not(:placeholder-shown) + label.
307   float-label,
308 .ui.options-cont .ui.float-input textarea:focus + label.float-label,
309 .ui.options-cont .ui.float-input input:-webkit-autofill + label.float-label,
310 .ui.options-cont .ui.float-input textarea:-webkit-autofill + label.float-label
311   {
312   color: #777!important;
313   font-weight: bold;
314   background: #eee;
315 }
316
317 .ui.options-cont ::placeholder,
318 .ui.options-cont ::-webkit-input-placeholder {
319   color: #ccc;
320   font-weight: normal;
321   text-transform: none;
322 }
323
324 .ui.options-cont input:disabled,
325 .ui.options-cont textarea:disabled,
326 .ui.options-cont select:disabled {
327   opacity: 0.5;
```

```
322     pointer-events: none;  
323 }  
324  
325 .ui.options-cont label {  
326     color: #12568a;  
327     margin-bottom: 2px;  
328     display: block;  
329     margin-left: 3px;  
330 }  
331  
332 .ui.options-cont label span {  
333     color: #555;  
334 }  
335  
336 .ui.options-cont label.checkcont {  
337     margin: 15px 0;  
338     font-size: 18px;  
339     padding-top: 3px;  
340     padding-bottom: 3px;  
341 }  
342  
343 .ui.options-cont label.checkcont {  
344     padding-top: 6px;  
345     color: #333;  
346 }  
347  
348 .ui.checkcont {  
349     display: block;  
350     position: relative;  
351     padding-left: 35px;  
352     margin-bottom: 12px;  
353     cursor: pointer;  
354     font-size: 22px;  
355     -webkit-user-select: none;  
356     -moz-user-select: none;  
357     -ms-user-select: none;  
358     user-select: none;  
359     -webkit-user-drag: none;  
360     user-drag: none;  
361 }  
362  
363 .ui.checkcont input {  
364     position: absolute;  
365     opacity: 0;  
366     cursor: pointer;  
367     height: 0;  
368     width: 0;  
369 }  
370  
371 .ui.checkcont .checkmark {  
372     position: absolute;  
373     top: 0;  
374     left: 0;  
375     height: 25px;  
376     width: 25px;
```



```
377     background-color: #fff ;
378     border: 2px solid #ccc ;
379     border-radius: 3px ;
380   }
381
382   .ui .checkcont:hover input ~ .checkmark {
383     background-color: #ccc ;
384   }
385
386   .ui .checkcont input:checked ~ .checkmark {
387     background-color: #2e85c7 ;
388   }
389
390   .ui .checkcont .checkmark:after {
391     content: "";
392     position: absolute;
393     display: none;
394   }
395
396   .ui .checkcont input:checked ~ .checkmark:after {
397     display: block;
398   }
399
400   .ui .checkcont .checkmark:after {
401     left: 9px;
402     top: 5px;
403     width: 5px;
404     height: 10px;
405     border: solid white;
406     border-width: 0 3px 3px 0;
407     -webkit-transform: rotate(45deg);
408     -ms-transform: rotate(45deg);
409     transform: rotate(45deg);
410   }
411
412   .ui .float-input input[type="text"] ,
413   .ui .float-input input[type="number"] ,
414   .ui .float-input input[type="submit"] ,
415   .ui .float-input input[type="password"] ,
416   .ui .float-input textarea ,
417   .ui .float-input select ,
418   .ui .float-input button {
419     font-family: 'Roboto';
420     border: 1px solid #ccc;
421     color: #333;
422     font-size: 16px;
423     padding-left: 10px;
424     padding-right: 10px;
425     padding-top: 8px;
426     padding-bottom: 8px;
427     width: calc(100% - 24px);
428     transition: all 0.2s linear;
429     -webkit-transition: all 0.2s linear;
430     border-radius: 3px;
431     -webkit-appearance: none;
```



```
432 background: #fff ;
433 font-family: 'Roboto' ;
434 -webkit-box-sizing:content-box ;
435 box-sizing:content-box ;
436 cursor: pointer ;
437 line-height: 18px ;
438 }
439
440 .ui.options-cont input[type="text"] ,
441 .ui.options-cont input[type="number"] ,
442 .ui.options-cont input[type="submit"] ,
443 .ui.options-cont input[type="password"] ,
444 .ui.options-cont textarea ,
445 .ui.options-cont select ,
446 .ui.options-cont button {
447 font-family: 'Roboto' ;
448 border: 2px solid #ccc ;
449 color: #333 ;
450 font-size: 18px ;
451 padding-left: 10px ;
452 padding-right: 10px ;
453 padding-top: 8px ;
454 padding-bottom: 8px ;
455 width: calc(100% - 24px) ;
456 margin-bottom: 10px ;
457 transition: all 0.2s linear ;
458 -webkit-transition: all 0.2s linear ;
459 font-weight: bold ;
460 border-radius: 3px ;
461 -webkit-appearance: none ;
462 background: #fff ;
463 font-family: 'Roboto' ;
464 -webkit-box-sizing:content-box ;
465 box-sizing:content-box ;
466 cursor: pointer ;
467 line-height: 22px ;
468 }
469
470 .ui.options-cont textarea {
471 resize: vertical ;
472 }
473
474 .ui.options-cont input[type="submit"] {
475 color: #fff ;
476 background: #2e85c7 ;
477 background-size: 30px 30px ;
478 border: 0 ;
479 margin-left: 2px ;
480 margin-bottom: 10px ;
481 cursor: pointer ;
482 text-transform: uppercase ;
483 border: 2px solid #2e85c7 ;
484 }
485
486 .ui.options-cont input[type="text"] :focus ,
```



```
487 .ui.options-cont input[type="number"]::focus,  
488 .ui.options-cont textarea::focus,  
489 .ui.options-cont select::focus {  
490   border: 2px solid #2e85c7;  
491 }  
492  
493 .ui.options-cont input[type="text"]::read-only:focus,  
494 .ui.options-cont input[type="number"]::read-only:focus {  
495   border: 2px solid #ccc;  
496 }  
497  
498 .ui.options-cont input[type="text"]::read-only:hover,  
499 .ui.options-cont input[type="number"]::read-only:hover {  
500   cursor: auto;  
501 }  
502  
503 .ui.options-cont input[type="submit"]::hover {  
504   color: #2e85c7;  
505   background: #fff;  
506 }  
507  
508 .ui.options-cont input[type="submit"]::disabled {  
509   background-image: linear-gradient(135deg, rgba(255, 255, 255, .15) 25%,  
510     transparent 25%,  
511     transparent 50%, rgba(255, 255, 255, .15) 50%, rgba(255, 255, 255, .15)  
512     75%,  
513     transparent 75%, transparent);  
514   background-size: 30px 30px;  
515   animation: animate-stripes 1s linear infinite;  
516   -webkit-animation: animate-stripes 1s linear infinite;  
517 }  
518  
519 .ui.options-cont button:not([disabled]):hover {  
520   color: #2e85c7;  
521   background: #fff;  
522 }  
523  
524 .ui.options-cont button {  
525   color: #fff;  
526   background: #2e85c7;  
527   background-size: 30px 30px;  
528   border: 0;  
529   cursor: pointer;  
530   text-transform: uppercase;  
531   border: 2px solid #2e85c7;  
532   margin-top: 15px;  
533 }  
534  
535 .ui.options-cont button:disabled {  
536   background-image: linear-gradient(135deg, rgba(255, 255, 255, .15) 25%,  
537     transparent 25%,  
538     transparent 50%, rgba(255, 255, 255, .15) 50%, rgba(255, 255, 255, .15)  
539     75%,  
540     transparent 75%, transparent);  
541   background-size: 30px 30px;
```



```
538 }
539
540 .ui .panel::-webkit-scrollbar ,
541 .ui .panel::-webkit-scrollbar-button ,
542 .ui .panel::-webkit-scrollbar-track ,
543 .ui .panel::-webkit-scrollbar-track-piece ,
544 .ui .panel::-webkit-scrollbar-thumb ,
545 .ui .panel::-webkit-scrollbar-corner ,
546 .ui .panel::-webkit-resizer {
547   background: transparent;
548 }
549
550 .ui .panel::-webkit-scrollbar {
551   width: 12px;
552   height: 12px;
553   -webkit-border-radius: 5px;
554   border-radius: 5px;
555 }
556
557 .ui .panel::-webkit-scrollbar-track-piece {
558   -webkit-border-radius: 5px;
559   border-radius: 5px;
560 }
561
562 .ui .panel::-webkit-scrollbar-thumb {
563   background: #ddd;
564   border-radius: 5px;
565   background-clip: content-box;
566   border: 2px solid transparent;
567 }
568
569 .ui .panel::-webkit-scrollbar-button {
570   width:0 ;
571   height:0 ;
572 }
573
574 .ui .tab-names {
575   display: flex;
576   gap: 5px;
577   margin-bottom: -1px;
578   padding-left: 10px;
579   padding-right: 10px;
580 }
581
582 .ui .tab-name {
583   padding: 5px 10px;
584   border: 1px solid #ccc;
585   border-bottom: 1px solid #ccc;
586   border-radius: 5px 5px 0 0;
587   background: #f5f5f5;
588   cursor: pointer;
589   user-select: none;
590 }
591
592 .ui .tab-name.active {
```

```
593     border-bottom: none;
594     background: #fff;
595 }
596
597 .ui.tab {
598     display: none;
599     border: 1px solid #ccc;
600     border-radius: 5px;
601     padding: 5px;
602     background: #fff;
603     margin: 0 5px 5px 5px;
604 }
605
606 .ui.divider {
607     border-bottom: 1px solid #ccc;
608     margin: 5px;
609 }
610
611 .ui.group {
612     border: 1px solid #ccc;
613     border-radius: 5px;
614     padding: 5px;
615     background: #fff;
616     margin: 5px;
617     margin-top: 10px;
618 }
619
620 .ui.group-title {
621     width: max-content;
622     padding: 5px 10px;
623     margin-top: -18px;
624     background: #fff;
625     margin-left: 5px;
626 }
627
628 .ui.columns-container {
629     display: flex;
630 }
631
632 .ui.columns-container .column {
633     flex: 1;
634     display: flex;
635     flex-direction: column;
636 }
637
638 .ui.columns-container .row {
639     flex: 1;
640     display: flex;
641     justify-content: center;
642     align-items: center;
643     padding: 0 5px;
644 }
645
646 .ui.text-center {
647     text-align: center;
```



```
648 }
649
650 .ui.float-input.text-center input {
651   text-align: center;
652 }
653
654 .ui.slider {
655   -webkit-appearance: none;
656   appearance: none;
657   height: 20px;
658   width: 300px;
659   outline: none;
660   background: #ddd;
661   border-radius: 5px;
662   z-index: 2;
663   position: relative;
664 }
665
666 .ui.slider.vertical {
667   writing-mode: vertical-lr;
668   height: 300px;
669   width: 20px;
670   display: inline-block;
671   transform: rotate(180deg);
672 }
673
674 .ui.slider::-webkit-slider-runnable-track {
675   cursor: pointer;
676 }
677
678 .ui.slider::-webkit-slider-thumb {
679   -webkit-appearance: none;
680   appearance: none;
681   width: 30px;
682   height: 60px;
683   background-color: #2e85c7;
684   cursor: pointer;
685   border-radius: 3px;
686   transition: all 0.4s;
687   border: 2px solid #fff;
688   background-image: linear-gradient(to right, transparent 0 2px, #fffff75 2px
689     4px, transparent 4px 6px, #fffff75 6px 8px, transparent 8px 10px, #
690     fffff75 10px 12px, transparent 12px 14px);
691   background-repeat: no-repeat;
692   background-size: 14px 90% ;
693   background-position: center;
694 }
695
696 .ui.slider.vertical::-webkit-slider-thumb {
697   width: 60px;
698   height: 30px;
699   background-image: linear-gradient(to bottom, transparent 0 2px, #fffff75 2
700     px 4px, transparent 4px 6px, #fffff75 6px 8px, transparent 8px 10px, #
701     fffff75 10px 12px, transparent 12px 14px);
702   background-size: 90% 14px;
```

```
699  }
700
701 .ui.slider:disabled::-webkit-slider-thumb {
702   background-image: linear-gradient(135deg, rgba(255, 255, 255, .15) 25%,
703   transparent 25%, transparent 50%, rgba(255, 255, 255, .15) 50%, rgba(255, 255, 255, .15)
704   75%, transparent 75%, transparent);
705   background-size: 30px 30px;
706   background-repeat: repeat;
707 }
708
709 .ui.slider-wrapper {
710   position: relative;
711   text-align: center;
712 }
713
714 .ui.slider-wrapper .slider-midpoint.vertical {
715   background: #999;
716   height: 4px;
717   width: 100%;
718   top: 50%;
719   left: 50%;
720   position: absolute;
721   transform: translateY(-50%) translateX(-50%);
722   z-index: 0;
723   border-radius: 2px;
724 }
```

Listing 29 - ui.css

5 html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Window - JSLAB | PR-DC</title>
6   <meta http-equiv="Content-Security-Policy" content="script-src * 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
7   <link rel="stylesheet" type="text/css" href="../css/basic.css" />
8   <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
9   <style id="custom-style"></style>
10 </head>
11 <body>
12 </body>
13 </html>
```

Listing 30 - blank.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
```

```

5   <title>Cesium Plot - JSLAB | PR-DC</title>
6   <meta http-equiv="Content-Security-Policy" content="script-src * 'unsafe-
7     inline' 'unsafe-eval' blob:;" />
8     <link rel="stylesheet" type="text/css" href="../css/basic.css" />
9     <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
10    <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
11    <link rel="stylesheet" type="text/css" href="../lib/Cesium-1.124/Widgets/
12      widgets.css">
13    <style>
14      .cesium-viewer-bottom, .cesium-viewer-toolbar, .cesium-viewer-
15        animationContainer, .cesium-viewer-fullscreenContainer {
16          display: none!important;
17        }
18    </style>
19    <style id="dynamic-style-rules"></style>
20  </head>
21  <body>
22    <div id="map-3d-cont"></div>
23
24    <script type="text/javascript" src="../lib/Cesium-1.124/Cesium.js"></
25      script>
26
27  </body>
28</html>
```

Listing 31 - cesium.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>D3 Canvas - JSLAB | PR-DC</title>
6      <meta http-equiv="Content-Security-Policy" content="script-src * 'self' ,
7        'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
8        <link rel="stylesheet" type="text/css" href="../css/basic.css" />
9        <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
10       <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
11
12       <style id="dynamic-style-rules"></style>
13    </head>
14    <body>
15      <svg id="d3-svg"></svg>
16      <canvas id="d3-canvas"></canvas>
17
18      <script type="text/javascript" src="../lib/d3-7.8.5/d3-7.8.5.min.js"></
19        script>
20
21  </body>
22</html>
```

Listing 32 - d3.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
```

```

5   <title>Editor - JSLAB | PR-DC</title>
6   <meta http-equiv="Content-Security-Policy" content="script-src * 'self' ,
7     unsafe-inline 'unsafe-eval'; worker-src 'self' blob:; />
8   <link rel="stylesheet" type="text/css" href="../css/tabs.css">
9   <link rel="stylesheet" type="text/css" href="../css/codemirror-notepadpp-
10    theme.css">
11  <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/
12    addon/fold/foldgutter.css">
13  <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/
14    addon/lint/lint.css">
15  <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/
16    addon/hint/show-hint.css">
17  <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/lib/
18    codemirror.css">
19  <link rel="stylesheet" type="text/css" href="../css/codemirror-editor-
20    custom.css">
21    <link rel="stylesheet" type="text/css" href="../css/editor.css
22      "/>
23  <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
24
25  <style id="dynamic-style-rules"></style>
26
27 </head>
28 <body>
29   <div id="editor-menu-container">
30     <div>
31       
32       <ul id="editor-menu">
33         <li id="new-script" title-str="26"><str sid="10"></str></li>
35         <li id="open-menu" title-str="27"><str
36           sid="11"></str></li>
37         <li id="save-menu" title-str="28"><str
38           sid="12"></str></li>
39         <li id="save-as-menu" title-str="29">
40           <str sid="13"></str></li>
41         <li id="run-menu" title-str="30"><str sid
42           ="14"></str></li>
43       </ul>
44       <ul id="window-controls">
45         <li id="win-minimize"></li>
46         <li id="win-restore"></li>
47         <li id="win-close"></li>
48       </ul>
49       <div id="app-title">JSLAB / EDITOR</div>
50       <div class="clear"></div>
51     </div>
52   </div>
53   <div class="tabs" style="--tab-content-margin: 9px">
54     <div class="tabs-content"></div>
55     
56     
57     <div class="tabs-bottom-bar"></div>
58   </div>
59   <div id="code"></div>
60   <div id="close-dialog-cont">
```

```

47 <div id="close-dialog">
48   <div id="close-dialog-header">
49     <span><str sid="47"></str></span>
50   </div>
51   <div id="close-dialog-msg">
52     <str sid="48"></str> <span id="close-file"></span> <str sid="49"></str>
53   </div>
54   <div id="close-dialog-buttons">
55     <button id="close-dialog-save"><str sid="50"></str></button>
56     <button id="close-dialog-discard"><str sid="51"></str></button>
57     <button id="close-dialog-cancel"><str sid="52"></str></button>
58   </div>
59   <br class="clear">
60 </div>
61 </div>
62
63 <div id="editor-more-popup" style="display: none;">
64   <ul>
65     <li id="search-dialog-menu" title-str="169"><str sid="168"></str></li>
66     <li id="compile-dialog-menu" class="arduino" title-str="227"><str sid="227"></str></li>
67     <li id="upload-dialog-menu" class="arduino" title-str="228"><str sid="228"></str></li>
68   </ul>
69   <div class="popup-triangle"></div>
70 </div>
71
72 <script>if (typeof module === 'object') {window.module = module; module = undefined;}</script>
73
74 <script type="text/javascript" src=".. / lib/draggabilly-2.3.0/draggabilly-2.3.0.min.js"></script>
75 <script type="text/javascript" src=".. / lib/PRDC_TABS/PRDC_TABS.js"></script>
76 <script type="text/javascript" src=".. / lib/jshint-2.13.0/jshint-2.13.0.js"></script>
77
78 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/lib/codemirror.js"></script>
79 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/selection/active-line.js"></script>
80 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/fold/foldcode.js"></script>
81 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/fold/foldgutter.js"></script>
82 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/fold/brace-fold.js"></script>
83 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/fold/indent-fold.js"></script>
84 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/fold/comment-fold.js"></script>
85 <script type="text/javascript" src=".. / lib/codemirror-5.49.2/addon/display/rulers.js"></script>

```

```

86   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/edit/
87     matchbrackets.js"></script>
88   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/lint/
89     lint.js"></script>
90   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/hint/
91     show-hint.js"></script>
92   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
93     searchcursor.js"></script>
94   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/scroll/
95     annotatescrollbar.js"></script>
96   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
97     matchesonscrollbar.js"></script>
98   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
99     jump-to-line.js"></script>
100  <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
101    match-highlighter.js"></script>
102  <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/comment/
103    /comment.js"></script>
104  <script type="text/javascript" src="../lib/codemirror-5.49.2/mode/
105    javascript/javascript.js"></script>
<script type="text/javascript" src="../lib/codemirror-5.49.2/mode/like/
  like.js"></script>

106  <script type="text/javascript" src="../js/code/custom-javascript-hint.js">
107    </script>
108  <script type="text/javascript" src="../js/code/dialog-search.js"></script>
109
110  <script type="text/javascript" src="../lib/jquery-3.7.0/jquery-3.7.0.min.
111    js"></script>
112
113  <script type="text/javascript" src="../js/editor/init-editor.js"></script>
114
115  </body>
</html>
```

Listing 33 - editor.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>Figure - JSLAB | PR-DC</title>
6      <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '
7        unsafe-inline 'unsafe-eval'; worker-src 'self' blob:;" />
8      <link rel="stylesheet" type="text/css" href="../css/figure.css"
9        "/>
10     <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
11     <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
12
13     <style id="dynamic-style-rules"></style>
14   </head>
15   <body>
16     <div id="figure-menu-button"></div>
17     <div id="figure-menu-container">
18       <div id="figure-menu">
19         <ul>
20           <li id="save-as-menu" title-str="152">
```

```

19   ><str sid="159"></str></li>
20   <li id="zoom-menu" class="specific -2d" title-str="153"><str sid="160"></str></li>
22   <li id="pan-menu" class="specific -2d" title-str="154"><str sid="161"></str></li>
24   <li id="rotate-menu" class="specific -2d" title-str="155"><str sid="162"></str></li>
26   <li id="zoom-in-menu" class="specific -2d" title-str="156"><str sid="163"></str></li>
28   <li id="zoom-out-menu" class="specific -2d" title-str="157"><str sid="164"></str></li>
30   <li id="fit-menu" class="specific -2d" title-str="158"><str sid="165"></str></li>
32   <li id="reset-menu" class="specific -2d" title-str="158"><str sid="221"></str></li>
34   <li id="pan-menu-3d" class="specific -3d" title-str="154"><str sid="161"></str></li>
36   <li id="rotate-menu-3d" class="specific -3d" title-str="155"><str sid="162"></str></li>
38   <li id="fit-menu-3d" class="specific -3d" title-str="158"><str sid="165"></str></li>
40   <li id="reset-menu-3d" class="specific -3d" title-str="158"><str sid="221"></str></li>
42   </ul>
43   <div class="clear"></div>
44 </div>
45 <div id="figure-content">
46 </div>
47
48 <script type="text/javascript" src=".../lib/plotly-2.24.2/plotly-2.24.2.min.
49   .js"></script>
50   <script type="text/javascript" src=".../js/windows/plot.js"></script>
51 </body>
52 </html>
```

Listing 34 - figure.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>%title%</title>
6      <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '
7        unsafe-inline 'unsafe-eval'; worker-src 'self' blob:; " />
8      <style>
9        html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre
10       , a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp,
11       small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li,
12       fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td,
13       article, aside, canvas, details, embed, figure, figcaption, footer, header,
14       hgroup, menu, nav, output, ruby, section, summary, time, mark, audio, video {
15         border:0; font-size:100%; font: inherit; vertical-align: baseline; margin:0;
16         padding:0} article, aside, details, figcaption, figure, footer, header, hgroup
17       , menu, nav, section{display: block} body{line-height:1} ol, ul{list-style:
18         none} blockquote, q{ quotes:none} blockquote:before, blockquote:after ,q:
```

```
before ,q:after {content:none}table{border-collapse:collapse;border-spacing:0}

9      #figure-content {
10     margin: 30px auto;
11     position: relative;
12     border: 1px solid #6f6f6f;
13     box-shadow: 0px 0px 20px #c1c1c1;
14   }
15
16
17   #figure-header {
18     background: #12568a;
19     background: linear-gradient(to left, #2e85c7, #12568a);
20     color: #fff;
21     user-select: none;
22     -webkit-user-drag: none;
23     user-drag: none;
24     max-height: 40px;
25     overflow: hidden;
26     transition: all .2s linear;
27     position: absolute;
28     left: 0;
29     right: 0;
30     top: 0;
31   }
32
33   #jslab-logo svg {
34     height: 30px;
35     float: left;
36     padding: 5px 10px;
37     user-select: none;
38     -webkit-user-drag: none;
39     user-drag: none;
40     width: 23px;
41     background: #00000047;
42   }
43
44   #company-logo svg {
45     filter: invert(1);
46     height: 30px;
47     float: left;
48     padding: 5px;
49     padding-left: 10px;
50     user-select: none;
51     -webkit-user-drag: none;
52     user-drag: none;
53     width: 54px;
54   }
55
56   #title {
57     font-family: 'Helvetica';
58     font-size: 20px;
59     padding: 3px 10px;
60     margin: 7px 0;
61     float: left;
```

```

62   font-weight: 400;
63   opacity: 0.8;
64   color: #ffffff;
65   border-left: 2px solid #fff;
66   margin-left: 5px;
67 }
68
69 #plot-cont {
70   padding-top: 40px;
71 }
72 </style>
73 </head>
74 <body>
75 <div id="figure-content">
76   <div id="figure-header">
77     <a href="https://pr-dc.com/jslab" title="JSLAB">
78       <div id="jslab-logo">
79         <svg
80           version="1.1"
81           viewBox="0 0 630 630"
82           xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
83           xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.
84           dtd"
85           xmlns="http://www.w3.org/2000/svg"
86           xmlns:svg="http://www.w3.org/2000/svg"
87           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
88           xmlns:cc="http://creativecommons.org/ns#"
89           xmlns:dc="http://purl.org/dc/elements/1.1/">
90         <metadata
91           id="metadata12">
92           <rdf:RDF>
93             <cc:Work
94               rdf:about="">
95               <dc:format>image/svg+xml</dc:format>
96               <dc:type
97                 rdf:resource="http://purl.org/dc/dcmitype/StillImage" />
98             </cc:Work>
99           </rdf:RDF>
100         </metadata>
101         <defs
102           id="defs10" />
103         <rect
104           id="background"
105             x="0"
106             y="0"
107             width="630"
108             height="630"
109             fill="#f7df1e"
110             rx="50"
111             ry="50" />
112         <path
d="m 132.3099,593 c 42.69727,0 71.96556,-22.72597
      71.96556,-72.65424 v -164.5911 h -48.2066 v 163.90243 c
      0,24.1033 -9.98565,30.30129 -25.82496,30.30129

```

```

113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
      -16.52798,0  -23.41463,-11.36298  -30.989963,-24.79196 L
      60.548.92539 C 71.362984,573.0287  93.744617,593
      132.3099,593 Z"
  style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
  id="path850" />
<path
  d="m 318.24908,593 c 45.79626,0 79.88522,-23.75897
    79.88522,-67.14491 0,-40.28695 -23.0703,-58.19225
    -64.04591,-75.75323 l -12.05165,-5.16499 c
    -20.65997,-8.95266 -29.61263,-14.80631
    -29.61263,-29.26829 0,-11.70732 8.95266,-20.65997
    23.0703,-20.65997 13.77332,0 22.72597,5.85365
    30.98996,20.65997 l 37.53228,-24.1033 C
    368.17734,363.67432 346.14004,353 315.49441,353 c
    -43.0416,0 -70.58823,27.54663 -70.58823,63.70158
    0,39.25394 23.0703,57.84791 57.84792,72.65422 l
    12.05165,5.165 c 22.0373,9.64132 35.12195,15.49498
    35.12195,32.02295 0,13.77332 -12.74032,23.75897
    -32.71162,23.75897 -23.75897,0 -37.18795,-12.39598
    -47.51793,-29.26829 L 230.4442,543.76039 C
    244.56185,571.65136 273.48581,593 318.24908,593 Z"
  style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
  id="path852" />
<path
  d="M 435.66613,589.901 H 589.92724 V 547.54805 H 483.87273 V
    355.75466 h -48.2066 z"
  style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
  id="path854" />
</svg>
</div>
</a>
<a href="https://pr-dc.com/" title="PR-DC Company">
  <div id="company-logo">
    <svg
      width="160mm"
      height="90mm"
      viewBox="0 0 160 90"
      version="1.1"
      id="svg869"
      xmlns="http://www.w3.org/2000/svg"
      xmlns:svg="http://www.w3.org/2000/svg"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cc="http://creativecommons.org/ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"/>
    <defs
      id="defs863" />

```

```

141 <metadata
142   id="metadata866">
143   <rdf:RDF>
144     <cc:Work
145       rdf:about=""
146       dc:format>image/svg+xml</dc:format>
147       <dc:type
148         rdf:resource="http://purl.org/dc/dcmitype/StillImage" />
149     </cc:Work>
150   </rdf:RDF>
151 </metadata>
152 <g
153   id="layer1">
154   <path
155     id="path847"
156     style="fill:#000000;fill-opacity:1;stroke:none;stroke-width:0.999999px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1"
157     d="M 483.13281 17.101562 C 483.13281 17.101562 455.74781
        17.701959 428.36328 17.615234 C 425.51072 17.606164
        422.57229 17.604649 419.5625 17.613281 C 374.41441
        17.745451 313.17548 18.456471 280.00586 28.585938 C
        219.34388 47.111121 185.96586 73.091557 158.50391
        103.79688 C 153.5363 109.42942 147.05989 117.97425
        139.72266 128.92188 C 143.64779 137.54078 145.61523
        147.65489 145.61523 159.27539 C 145.61523 172.166
        143.27648 183.00438 138.59766 191.78906 C 133.91885
        200.57382 127.9503 207.49589 120.69336 212.55664 C
        113.5319 217.52192 106.22722 220.81618 98.779297
        222.43945 C 95.807549 223.02819 92.396307 223.51965
        88.642578 223.93555 C 75.710747 253.86377 63.915338
        287.36997 55.78125 322.60352 L 449.07422 322.89844 C
        452.91989 288.9274 456.14589 259.97604 459.13281
        232.99414 C 454.31399 221.25319 451.90039 207.64528
        451.90039 192.16211 C 451.90039 165.94179 458.11321
        145.11257 470.5293 129.66602 C 474.28218 95.661149
        478.18692 60.495264 483.13281 17.101562 z M 539.49609
        115.06445 C 519.44401 115.06445 503.26564 121.67252
        490.96094 134.88867 C 478.65626 148.03972 472.50391
        166.52928 472.50391 190.35742 C 472.50391 212.88348
        478.62368 230.65689 490.86328 243.67773 C 503.10285
        256.63347 518.72784 263.11133 537.73828 263.11133 C
        553.10287 263.11133 565.76562 259.33529 575.72656
        251.7832 C 585.75262 244.16601 592.91407 232.54493
        597.21094 216.91992 L 569.18359 208.0332 C 566.77472
        218.51498 562.80339 226.19726 557.26953 231.08008 C
        551.73567 235.96289 545.12761 238.4043 537.44531
        238.4043 C 527.02866 238.4043 518.56513 234.56315
        512.05469 226.88086 C 505.54427 219.19856 502.28906
        206.30796 502.28906 188.20898 C 502.28906 171.15168
        505.57682 158.81446 512.15234 151.19727 C 518.79297
        143.58007 527.41927 139.77148 538.03125 139.77148 C
        545.71354 139.77148 552.22394 141.91992 557.5625
        146.2168 C 562.96614 150.51367 566.51431 156.37304

```

568.20703 163.79492 L 596.82031 156.95898 C 593.56511
145.50064 588.68231 136.71159 582.17188 130.5918 C
571.23438 120.24023 557.00914 115.06445 539.49609
115.06445 z M 14.216797 117.50586 L 14.216797 260.66992
L 43.123047 260.66992 L 43.123047 206.66602 L
61.970703 206.66602 C 75.05665 206.66602 85.050127
205.98242 91.951172 204.61523 C 97.0293 203.50846
102.00976 201.26237 106.89258 197.87695 C 111.8405
194.42643 115.9095 189.70638 119.09961 183.7168 C
122.28971 177.72721 123.88477 170.3379 123.88477
161.54883 C 123.88477 150.15559 121.11784 140.87826
115.58398 133.7168 C 110.05013 126.49023 103.18165
121.80273 94.978516 119.6543 C 89.639971 118.222
78.181651 117.50586 60.603516 117.50586 L 14.216797
117.50586 z M 157.9082 117.50586 L 218.74805 117.50586
C 234.04754 117.50586 245.14779 118.80794 252.04883
121.41211 C 259.01499 123.95116 264.58138 128.50844
268.74805 135.08398 C 272.91471 141.65949 274.99805
149.17908 274.99805 157.64258 C 274.99805 168.38479
271.84049 177.27149 265.52539 184.30273 C 259.21025
191.26886 249.77016 195.66342 237.20508 197.48633 C
243.45506 201.1321 248.59831 205.13606 252.63477
209.49805 C 256.73635 213.86 262.23763 221.60745
269.13867 232.74023 L 286.61914 260.66992 L 252.04883
260.66992 L 231.15039 229.51758 C 223.72849 218.38483
218.65039 211.38606 215.91602 208.52148 C 213.18164
205.59178 210.2845 203.60613 207.22461 202.56445 C
204.16471 201.45769 199.31446 200.9043 192.67383
200.9043 L 186.81445 200.9043 L 186.81445 260.66992 L
157.9082 260.66992 L 157.9082 117.50586 z M 322.95312
117.50586 L 375.78516 117.50586 C 387.69921 117.50586
396.78123 118.41729 403.03125 120.24023 C 411.4297
122.7142 418.62367 127.10873 424.61328 133.42383 C
430.60285 139.73889 435.16017 147.48634 438.28516
156.66602 C 441.41018 165.78061 442.97266 177.04361
442.97266 190.45508 C 442.97266 202.23893 441.50783
212.39517 438.57812 220.92383 C 434.99745 231.34047
429.88672 239.77153 423.24609 246.2168 C 418.23308
251.09964 411.46225 254.9082 402.93359 257.64258 C
396.55341 259.66081 388.02475 260.66992 377.34766
260.66992 L 322.95312 260.66992 L 322.95312 117.50586 z
M 43.123047 141.72461 L 57.087891 141.72461 C
67.504562 141.72461 74.43815 142.05013 77.888672
142.70117 C 82.576175 143.54753 86.449868 145.66341
89.509766 149.04883 C 92.569664 152.43425 94.099609
156.73112 94.099609 161.93945 C 94.099609 166.17123
92.99284 169.88216 90.779297 173.07227 C 88.630857
176.26237 85.636071 178.60612 81.794922 180.10352 C
77.953775 181.60091 70.336597 182.34961 58.943359
182.34961 L 43.123047 182.34961 L 43.123047 141.72461 z
M 186.81445 141.72461 L 186.81445 178.05273 L
208.20117 178.05273 C 222.06836 178.05273 230.72721
177.4668 234.17773 176.29492 C 237.62825 175.12304
240.33009 173.10485 242.2832 170.24023 C 244.23635
167.37565 245.21289 163.79492 245.21289 159.49805 C

```

245.21289 154.68032 243.91081 150.80665 241.30664
147.87695 C 238.76755 144.88217 235.15428 142.99412
230.4668 142.21289 C 228.12304 141.88747 221.0918
141.72461 209.37305 141.72461 L 186.81445 141.72461 z M
351.85938 141.72461 L 351.85938 236.54883 L 373.44141
236.54883 C 381.51433 236.54883 387.34115 236.09311
390.92188 235.18164 C 395.60935 234.00976 399.48306
232.02411 402.54297 229.22461 C 405.66796 226.42511
408.20709 221.8353 410.16016 215.45508 C 412.1133
209.00977 413.08984 200.25325 413.08984 189.18555 C
413.08984 178.11784 412.1133 169.62179 410.16016
163.69727 C 408.20709 157.77282 405.47267 153.15039
401.95703 149.83008 C 398.44139 146.50976 393.98179
144.26368 388.57812 143.0918 C 384.54168 142.18033
376.63151 141.72461 364.84766 141.72461 L 351.85938
141.72461 z M 275.35742 187.93945 L 311.72266 187.93945
L 311.72266 206.12305 L 275.35742 206.12305 L
275.35742 187.93945 z "
158   transform="scale(0.26458333)" />
159 <g
160   aria-label="PR DC"
161   id="text865"
162   style="font-style:normal;font-variant:normal;font-weight:
163     normal;font-stretch:normal;font-size:52.9167px;line-
164     height:1.25;font-family:arial;-inkscape-font-
165     specification:arial;letter-spacing:0px;word-spacing:0px
166     ;fill:#000000;fill-opacity:1;stroke:none;stroke-width
167     :0.264583"
168   transform="translate(-26.307505,-2.9264341)" />
169 </g>
170 </svg>
171 </div>
172 </a>
173 <div id="title">
174   %title%
175 </div>
176 <div id="plot-cont">
177   
178 </div>
179 </div>
180 <script>
181   window.onload = function () {
182     var figure = document.getElementById('figure');
183     document.getElementById('figure-content').style.width = figure.width +
184       'px';
185   };
186 </script>
187 </body>
188 </html>

```

Listing 35 - html_figure.html

```

1 <!DOCTYPE html>
2 <html>

```

```
3   <head>
4     <meta charset="UTF-8">
5     <title>%title%</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '
7       unsafe-inline 'unsafe-eval'; worker-src 'self' blob;" />
8     <style>
9       html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre
10      , a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp,
11      small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li,
12      fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td,
13      article, aside, canvas, details, embed, figure, figcaption, footer, header,
14      hgroup, menu, nav, output, ruby, section, summary, time, mark, audio, video{
15        border: 0; font-size: 100%; font: inherit; vertical-align: baseline; margin: 0;
16        padding: 0} article, aside, details, figcaption, figure, footer, header, hgroup
17      , menu, nav, section{display: block} body{line-height: 1} ol, ul{list-style:
18        none} blockquote, q{ quotes: none} blockquote: before, blockquote: after, q:
19        before, q: after{content: none} table{border-collapse: collapse; border-
20        spacing: 0}
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45 #figure-content {
    margin: 30px auto;
    position: relative;
    border: 1px solid #6f6f6f;
    box-shadow: 0px 0px 20px #c1c1c1;
    width: max-content;
}
#figure-header {
    background: #12568a;
    background: linear-gradient(to left, #2e85c7, #12568a);
    color: #fff;
    user-select: none;
    -webkit-user-drag: none;
    user-drag: none;
    max-height: 40px;
    overflow: hidden;
    transition: all .2s linear;
    position: absolute;
    left: 0;
    right: 0;
    top: 0;
}
#jslab-logo svg {
    height: 30px;
    float: left;
    padding: 5px 10px;
    user-select: none;
    -webkit-user-drag: none;
    user-drag: none;
    width: 23px;
    background: #00000047;
}
#company-logo svg {
```

```
46     filter: invert(1);  
47     height: 30px;  
48     float: left;  
49     padding: 5px;  
50     padding-left: 10px;  
51     user-select: none;  
52     -webkit-user-drag: none;  
53     user-drag: none;  
54     width: 54px;  
55 }  
56  
57 #title {  
58     font-family: 'Helvetica';  
59     font-size: 20px;  
60     padding: 3px 10px;  
61     margin: 7px 0;  
62     float: left;  
63     font-weight: 400;  
64     opacity: 0.8;  
65     color: #ffffff;  
66     border-left: 2px solid #fff;  
67     margin-left: 5px;  
68 }  
69  
70 #plot-cont {  
71     padding-top: 40px;  
72 }  
73 </style>  
74 </head>  
75 <body>  
76     <div id="figure-content">  
77         <div id="figure-header">  
78             <a href="https://pr-dc.com/jslab" title="JSLAB">  
79                 <div id="jslab-logo">  
80                     <svg  
81                         version="1.1"  
82                         viewBox="0 0 630 630"  
83                         xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"  
84                         xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.  
85                         dtd"  
86                         xmlns="http://www.w3.org/2000/svg"  
87                         xmlns:svg="http://www.w3.org/2000/svg"  
88                         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
89                         xmlns:cc="http://creativecommons.org/ns#"  
90                         xmlns:dc="http://purl.org/dc/elements/1.1/">  
91             <metadata  
92                 id="metadata12">  
93                 <rdf:RDF>  
94                     <cc:Work  
95                         rdf:about="">  
96                         <dc:format>image/svg+xml</dc:format>  
97                         <dc:type  
98                             rdf:resource="http://purl.org/dc/dcmitype/StillImage" />  
99                     </cc:Work>
```

```

99          </ rdf:RDF>
100         </metadata>
101         <defs
102           id="defs10" />
103           <rect
104             id="background"
105             x="0"
106             y="0"
107             width="630"
108             height="630"
109             fill="#f7df1e"
110             rx="50"
111             ry="50" />
112           <path
113             d="m 132.3099,593 c 42.69727,0 71.96556,-22.72597
114               71.96556,-72.65424 v -164.5911 h -48.2066 v 163.90243 c
115                 0,24.1033 -9.98565,30.30129 -25.82496,30.30129
116               -16.52798,0 -23.41463,-11.36298 -30.989963,-24.79196 L
117                 60,548.92539 C 71.362984,573.0287 93.744617,593
118               132.3099,593 Z"
119             style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
120             id="path850" />
121           <path
122             d="m 318.24908,593 c 45.79626,0 79.88522,-23.75897
123               79.88522,-67.14491 0,-40.28695 -23.0703,-58.19225
124                 -64.04591,-75.75323 l -12.05165,-5.16499 c
125                   -20.65997,-8.95266 -29.61263,-14.80631
126                     -29.61263,-29.26829 0,-11.70732 8.95266,-20.65997
127                       23.0703,-20.65997 13.77332,0 22.72597,5.85365
128                         30.98996,20.65997 l 37.53228,-24.1033 C
129                           368.17734,363.67432 346.14004,353 315.49441,353 c
130                             -43.0416,0 -70.58823,27.54663 -70.58823,63.70158
131                               0,39.25394 23.0703,57.84791 57.84792,72.65422 l
132                                 12.05165,5.165 c 22.0373,9.64132 35.12195,15.49498
133                                   35.12195,32.02295 0,13.77332 -12.74032,23.75897
134                                     -32.71162,23.75897 -23.75897,0 -37.18795,-12.39598
135                                       -47.51793,-29.26829 L 230.4442,543.76039 C
136                                         244.56185,571.65136 273.48581,593 318.24908,593 Z"
137             style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
138             id="path852" />
139           <path
140             d="M 435.66613,589.901 H 589.92724 V 547.54805 H 483.87273 V
141               355.75466 h -48.2066 z"
142             style="font-size:341.94px;line-height:1.05;font-family:'Neutra Text';-inkscape-font-specification:'Neutra Text';word-spacing:0px;fill:#000000;fill-opacity:1;stroke-width:8.60834"
143             id="path854" />
144         </svg>

```

```

125 </div>
126 </a>
127 <a href="https://pr-dc.com/" title="PR-DC Company">
128   <div id="company-logo">
129     <svg
130       width="160mm"
131       height="90mm"
132       viewBox="0 0 160 90"
133       version="1.1"
134       id="svg869"
135       xmlns="http://www.w3.org/2000/svg"
136       xmlns:svg="http://www.w3.org/2000/svg"
137       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
138       xmlns:cc="http://creativecommons.org/ns#"
139       xmlns:dc="http://purl.org/dc/elements/1.1/">
140     <defs
141       id="defs863" />
142     <metadata
143       id="metadata866">
144       <rdf:RDF>
145         <cc:Work
146           rdf:about="">
147           <dc:format>image/svg+xml</dc:format>
148           <dc:type
149             rdf:resource="http://purl.org/dc/dcmitype/StillImage" /
150             >
151           </cc:Work>
152         </rdf:RDF>
153       </metadata>
154     <g
155       id="layer1">
156       <path
157         id="path847"
158         style="fill:#000000;fill-opacity:1;stroke:none;stroke-width
159         :0.999999px;stroke-linecap:butt;stroke-linejoin:miter;
160         stroke-opacity:1"
161         d="M 483.13281 17.101562 C 483.13281 17.101562 455.74781
162           17.701959 428.36328 17.615234 C 425.51072 17.606164
163           422.57229 17.604649 419.5625 17.613281 C 374.41441
164           17.745451 313.17548 18.456471 280.00586 28.585938 C
165           219.34388 47.111121 185.96586 73.091557 158.50391
166           103.79688 C 153.5363 109.42942 147.05989 117.97425
167           139.72266 128.92188 C 143.64779 137.54078 145.61523
168           147.65489 145.61523 159.27539 C 145.61523 172.166
169           143.27648 183.00438 138.59766 191.78906 C 133.91885
170           200.57382 127.9503 207.49589 120.69336 212.55664 C
171           113.5319 217.52192 106.22722 220.81618 98.779297
172           222.43945 C 95.807549 223.02819 92.396307 223.51965
173           88.642578 223.93555 C 75.710747 253.86377 63.915338
174           287.36997 55.78125 322.60352 L 449.07422 322.89844 C
175           452.91989 288.9274 456.14589 259.97604 459.13281
176           232.99414 C 454.31399 221.25319 451.90039 207.64528
177           451.90039 192.16211 C 451.90039 165.94179 458.11321
178           145.11257 470.5293 129.66602 C 474.28218 95.661149
179           478.18692 60.495264 483.13281 17.101562 z M 539.49609

```

115.06445 C 519.44401 115.06445 503.26564 121.67252
490.96094 134.88867 C 478.65626 148.03972 472.50391
166.52928 472.50391 190.35742 C 472.50391 212.88348
478.62368 230.65689 490.86328 243.67773 C 503.10285
256.63347 518.72784 263.11133 537.73828 263.11133 C
553.10287 263.11133 565.76562 259.33529 575.72656
251.7832 C 585.75262 244.16601 592.91407 232.54493
597.21094 216.91992 L 569.18359 208.0332 C 566.77472
218.51498 562.80339 226.19726 557.26953 231.08008 C
551.73567 235.96289 545.12761 238.4043 537.44531
238.4043 C 527.02866 238.4043 518.56513 234.56315
512.05469 226.88086 C 505.54427 219.19856 502.28906
206.30796 502.28906 188.20898 C 502.28906 171.15168
505.57682 158.81446 512.15234 151.19727 C 518.79297
143.58007 527.41927 139.77148 538.03125 139.77148 C
545.71354 139.77148 552.22394 141.91992 557.5625
146.2168 C 562.96614 150.51367 566.51431 156.37304
568.20703 163.79492 L 596.82031 156.95898 C 593.56511
145.50064 588.68231 136.71159 582.17188 130.5918 C
571.23438 120.24023 557.00914 115.06445 539.49609
115.06445 z M 14.216797 117.50586 L 14.216797 260.66992
L 43.123047 260.66992 L 43.123047 206.66602 L
61.970703 206.66602 C 75.05665 206.66602 85.050127
205.98242 91.951172 204.61523 C 97.0293 203.50846
102.00976 201.26237 106.89258 197.87695 C 111.8405
194.42643 115.9095 189.70638 119.09961 183.7168 C
122.28971 177.72721 123.88477 170.3379 123.88477
161.54883 C 123.88477 150.15559 121.11784 140.87826
115.58398 133.7168 C 110.05013 126.49023 103.18165
121.80273 94.978516 119.6543 C 89.639971 118.222
78.181651 117.50586 60.603516 117.50586 L 14.216797
117.50586 z M 157.9082 117.50586 L 218.74805 117.50586
C 234.04754 117.50586 245.14779 118.80794 252.04883
121.41211 C 259.01499 123.95116 264.58138 128.50844
268.74805 135.08398 C 272.91471 141.65949 274.99805
149.17908 274.99805 157.64258 C 274.99805 168.38479
271.84049 177.27149 265.52539 184.30273 C 259.21025
191.26886 249.77016 195.66342 237.20508 197.48633 C
243.45506 201.1321 248.59831 205.13606 252.63477
209.49805 C 256.73635 213.86 262.23763 221.60745
269.13867 232.74023 L 286.61914 260.66992 L 252.04883
260.66992 L 231.15039 229.51758 C 223.72849 218.38483
218.65039 211.38606 215.91602 208.52148 C 213.18164
205.59178 210.2845 203.60613 207.22461 202.56445 C
204.16471 201.45769 199.31446 200.9043 192.67383
200.9043 L 186.81445 200.9043 L 186.81445 260.66992 L
157.9082 260.66992 L 157.9082 117.50586 z M 322.95312
117.50586 L 375.78516 117.50586 C 387.69921 117.50586
396.78123 118.41729 403.03125 120.24023 C 411.4297
122.7142 418.62367 127.10873 424.61328 133.42383 C
430.60285 139.73889 435.16017 147.48634 438.28516
156.66602 C 441.41018 165.78061 442.97266 177.04361
442.97266 190.45508 C 442.97266 202.23893 441.50783
212.39517 438.57812 220.92383 C 434.99745 231.34047
429.88672 239.77153 423.24609 246.2168 C 418.23308

```

251.09964 411.46225 254.9082 402.93359 257.64258 C
396.55341 259.66081 388.02475 260.66992 377.34766
260.66992 L 322.95312 260.66992 L 322.95312 117.50586 z
M 43.123047 141.72461 L 57.087891 141.72461 C
67.504562 141.72461 74.43815 142.05013 77.888672
142.70117 C 82.576175 143.54753 86.449868 145.66341
89.509766 149.04883 C 92.569664 152.43425 94.099609
156.73112 94.099609 161.93945 C 94.099609 166.17123
92.99284 169.88216 90.779297 173.07227 C 88.630857
176.26237 85.636071 178.60612 81.794922 180.10352 C
77.953775 181.60091 70.336597 182.34961 58.943359
182.34961 L 43.123047 182.34961 L 43.123047 141.72461 z
M 186.81445 141.72461 L 186.81445 178.05273 L
208.20117 178.05273 C 222.06836 178.05273 230.72721
177.4668 234.17773 176.29492 C 237.62825 175.12304
240.33009 173.10485 242.2832 170.24023 C 244.23635
167.37565 245.21289 163.79492 245.21289 159.49805 C
245.21289 154.68032 243.91081 150.80665 241.30664
147.87695 C 238.76755 144.88217 235.15428 142.99412
230.4668 142.21289 C 228.12304 141.88747 221.0918
141.72461 209.37305 141.72461 L 186.81445 141.72461 z M
351.85938 141.72461 L 351.85938 236.54883 L 373.44141
236.54883 C 381.51433 236.54883 387.34115 236.09311
390.92188 235.18164 C 395.60935 234.00976 399.48306
232.02411 402.54297 229.22461 C 405.66796 226.42511
408.20709 221.8353 410.16016 215.45508 C 412.1133
209.00977 413.08984 200.25325 413.08984 189.18555 C
413.08984 178.11784 412.1133 169.62179 410.16016
163.69727 C 408.20709 157.77282 405.47267 153.15039
401.95703 149.83008 C 398.44139 146.50976 393.98179
144.26368 388.57812 143.0918 C 384.54168 142.18033
376.63151 141.72461 364.84766 141.72461 L 351.85938
141.72461 z M 275.35742 187.93945 L 311.72266 187.93945
L 311.72266 206.12305 L 275.35742 206.12305 L
275.35742 187.93945 z "
  transform="scale(0.26458333)" />
<g
  aria-label="PR DC"
  id="text865"
  style="font-style: normal; font-variant: normal; font-weight:
    normal; font-stretch: normal; font-size: 52.9167px; line-
    height: 1.25; font-family: arial; -inkscape-font-
    specification: arial; letter-spacing: 0px; word-spacing: 0px
    ; fill: #000000; fill-opacity: 1; stroke: none; stroke-width
    : 0.264583"
  transform="translate(-26.307505, -2.9264341)" />
</g>
</svg>
</div>
</a>
<div id="title">
  %title%
</div>
</div>
<div id="plot-cont"></div>

```

```

174 </div>
175 <script src="https://cdn.plot.ly/plotly-2.24.2.min.js"></script>
176 <script>
177   var figure = %figure_data%;
178   Plotly.newPlot('plot-cont', figure);
179 </script>
180 </body>
181 </html>
```

Listing 36 - i_html_figure.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Leaflet Plot - JSLAB | PR-DC</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
7       <link rel="stylesheet" type="text/css" href="../css/basic.css" />
8     <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
9     <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
10    <link rel="stylesheet" type="text/css" href="../lib/leaflet-1.9.4/leaflet.css" />
11
12    <style id="dynamic-style-rules"></style>
13  </head>
14  <body>
15    <div id="map-cont"></div>
16
17    <script type="text/javascript" src="../lib/leaflet-1.9.4/leaflet.js"></script>
18    <script type="text/javascript" src="../lib/leaflet.rotatedMarker-0.2.0/leaflet.rotatedMarker.js"></script>
19  </body>
20 </html>
```

Listing 37 - leaflet.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>JSLAB | PR-DC</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
7       <link rel="stylesheet" type="text/css" href="../css/codemirror-notepadpp-theme.css">
8       <link rel="stylesheet" type="text/css" href="../css/highlight-notepadpp-theme.css">
9       <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/addon/lint/lint.css">
10      <link rel="stylesheet" type="text/css" href="../css/big-json-viewer-notepadpp-theme.css">
11      <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/addon/hint/show-hint.css">
```

```
12 <link rel="stylesheet" type="text/css" href="../lib/codemirror-5.49.2/lib/
13   codemirror.css">
<link rel="stylesheet" type="text/css" href="../css/codemirror-main-custom.
14   css">
15   <link type="text/css" rel="stylesheet" href="../css/main.css" />
16 <link type="text/css" rel="stylesheet" href="../css/terminal.css" />
17 <link type="text/css" rel="stylesheet" href="../font/roboto.css" />
18 <link type="text/css" rel="stylesheet" href="../font/RobotoMono.css" />
19 <style id="dynamic-style-rules"></style>
20 </head>
21 <body>
22 <div id="main-menu-container">
23   <div>
24     
25     
26     <ul id="main-menu">
27       <li id="editor-menu"><str sid="1"></
28         str></li>
29       <li id="help-menu"><str sid="2"></str></
30         li>
31       <li id="info-menu"><str sid="3"></str></
32         li>
33       <li id="settings-menu"><str sid=
34         = "9"></str></li>
35     </ul>
36     <ul id="window-controls">
37       <li id="win-minimize"></li>
38       <li id="win-restore"></li>
39       <li id="win-close"></li>
40     </ul>
41     <div id="app-title">JSLAB / MAIN</div>
42     <div class="clear"></div>
43   </div>
44 </div>
45 <div id="folder-navigation-container">
46   <div class="button float-left previous-folder disabled"></div>
48   <div class="button float-left next-folder disabled"></div>
50   <div class="button float-left up-folder disabled"></div>
52   <div class="button float-left open-folder"></div>
54   <div class="address-line-container">
55     <div class="folder-icon"></div>
56     <div class="folder-address">
57       <input class="address-line float-left" value="C:/ Electron/JSLAB">
58     <div class="current-address-cont">
59       <div class="current-address-wrap">
60         <div class="current-address">
61           <span class="folder">Local Disk (C:)</span>
62           <i class="i-next-folder"></i>
63     </div>
64   </div>
65 </div>
```

```
56      <span class="folder">Electron</span>
57      <i class="i-next-folder"></i>
58      <span class="folder">JSLAB</span>
59    </div>
60  </div>
61  </div>
62  <div id="save-path"><i title-str="35"></i></div>
63  </div>
64</div>
65 <div class="button float-right" id="paths-menu"></div>
67 <div class="clear"></div>
68</div>
69 <div id="panels-container">
70   <div id="left-panel">
71     <div id="left-top-panel">
72       <div class="cell-padding">
73         <div class="panel-container">
74           <div class="panel-title"><str sid="4"></str></div>
75
76           <div id="file-browser-options" class="options">
77             <div class="options-right">
78               <i class="refresh" title-str="37"></i>
79             </div>
80           </div>
81
82           <div id="file-browser" class="panel">
83             <div id="file-browser-cont"></div>
84           </div>
85         </div>
86       </div>
87     </div>
88
89   <div id="left-middle-panel" >
90     <div class="cell-padding">
91       <div class="panel-container">
92         <div class="panel-title"><str sid="5"></str></div>
93
94         <div id="workspace-options" class="options">
95           <div class="options-right">
96             <i class="clear" title-str="38"></i>
97           </div>
98         </div>
99
100        <div id="workspace-table-head">
101          <div class="row">
102            <div class="col col-1"><str sid="53"></str></div>
103            <div class="col col-2"><str sid="54"></str></div>
104            <div class="col col-3"><str sid="55"></str></div>
105          </div>
106        </div>
107
108        <div id="workspace" class="panel">
109          <div class="table"></div>
```

```
110      </div>
111      </div>
112      </div>
113      </div>
114
115      <div id="left-bottom-panel">
116          <div class="cell-padding">
117              <div class="panel-container">
118                  <div class="panel-title"><str sid="6"></str></div>
119
120                  <div id="command-history-options" class="options">
121                      <div class="options-right">
122                          <i class="clear" title-str="39"></i>
123                      </div>
124                  </div>
125
126                  <div id="command-history" class="panel">
127                      </div>
128                  </div>
129                  </div>
130                  </div>
131          </div>
132      </div>
133
134      <div id="right-panel">
135          <div class="cell-padding">
136              <div class="panel-container">
137                  <div class="panel-title"><str sid="7"></str></div>
138
139                  <div id="command-window-options" class="options">
140                      <div class="options-right">
141                          <i class="settings" title-str="40"></i>
142                          <i class="timestamp" title-str="41"></i>
143                          <i class="autoscroll active" title-str="42"></i>
144                          <i class="clear" title-str="43"></i>
145                          <i class="log" title-str="44"></i>
146                          <i class="to-bottom" title-str="45"></i>
147                      </div>
148                  </div>
149
150                  <div id="command-window" class="terminal-panel panel">
151                      <div id="command-window-messages" class="messages no-timestamp">
152                          </div>
153
154                      <div id="command-window-input-container">
155                          <div id="command-window-input-submit-cont">
156                              
157                              </div>
158                              <textarea id="command-window-input">cmd_help;</textarea>
159                          </div>
160                      </div>
161
162                      <div id="command-window-settings" class="terminal-dialog terminal-
settings options-panel panel" tabindex="0">
163                          <div class="options-cont">
```

```

163 <div class="options-header"><span><str sid="56"></str></span>
164 
166 </div>
167 <div class="float-input" title-str="17">
168   <input autocomplete="off" type="text" name="N-messages-max"
169     class="N-messages-max" placeholder="Infinity" value="">
170   <label class="float-label" for="N-messages-max"><str sid="17">
171     </str></label>
172 </div>
173 <button class="change-settings"><str sid="57"></str></button>
174 </div>
175 <div id="command-window-log" class="terminal-dialog terminal-log
176   options-panel panel" tabindex="0">
177   <div class="options-cont">
178     <div class="options-header">
179       <span><str sid="58"></str></span>
180       
182     </div>
183     <label class="checkcont"><str sid="59"></str><input class="
184       write-timestamps" type="checkbox" name="write-timestamps"
185         value="1" checked><span class="checkmark"></span></label>
186     <button class="save-log"><str sid="60"></str></button>
187   </div>
188 </div>
189 <div id="command-window-history" class="terminal-dialog history-
190   cont" tabindex="0">
191   <div class="history-header">
192     <span><str sid="61"></str></span>
193     
194   </div>
195   <ul class="history-panel panel" tabindex="0">
196     </ul>
197 </div>
198 <div id="script-path-container" class="main-dialog" tabindex="0">
199   <div class="page-cont">
200     <div class="page-header">
201       
203       <span><str sid="62"></str></span>
204     </div>
205     <div class="page-panel panel">
206       <div id="script-path-dialog-msg">
<str sid="63"></str> <span id="script-path"></span> <str sid="64">
</str>

```

```
207      </div>
208      <div id="script-path-dialog-buttons">
209          <button id="script-path-dialog-change-dir"><str sid="65"></str></button>
210          <button id="script-path-dialog-save"><str sid="66"></str></button>
211          <button id="script-path-dialog-run"><str sid="67"></str></button>
212      </div>
213      <br class="clear">
214      </div>
215  </div>
216
217
218  <div id="paths-container" class="main-dialog" tabindex="0">
219      <div class="page-cont wide">
220          <div class="page-header">
221              <span><str sid="68"></str></span>
222              
224          </div>
225          <div class="page-panel panel">
226              <ul></ul>
227          </div>
228      </div>
229
230  <div id="help-container" class="main-dialog" tabindex="0">
231      <div class="page-cont wide">
232          <div class="page-header">
233              <span><str sid="2"></str></span>
234              
236          </div>
237          <div class="page-panel panel">
238              <h1><str sid="69"></str></h1>
239              <table style="width:100%">
240                  <tr>
241                      <th><str sid="70"></str></th>
242                      <th><str sid="71"></str></th>
243                  </tr>
244                  <tr>
245                      <td>ESC</td>
246                      <td><str sid="72"></str></td>
247                  </tr>
248                  <tr>
249                      <td>Arrow Up</td>
250                      <td><str sid="73"></str></td>
251                  </tr>
252                  <tr>
253                      <td>Arrow Down</td>
254                      <td><str sid="74"></str></td>
255                  </tr>
256                  <tr>
257                      <td>Page Up</td>
258                      <td><str sid="75"></str></td>
259                  </tr>
```

```
259 <tr>
260   <td>Page Down</td>
261   <td><str sid="76"></str></td>
262 </tr>
263 <tr>
264   <td>Shift + Enter</td>
265   <td><str sid="77"></str></td>
266 </tr>
267 <tr>
268   <td>F3</td>
269   <td><str sid="78"></str></td>
270 </tr>
271 <tr>
272   <td>F7</td>
273   <td><str sid="79"></str></td>
274 </tr>
275 <tr>
276   <td>Alt + F7</td>
277   <td><str sid="80"></str></td>
278 </tr>
279 <tr>
280   <td>F8</td>
281   <td><str sid="81"></str></td>
282 </tr>
283 <tr>
284   <td>Ctrl + F</td>
285   <td><str sid="82"></str></td>
286 </tr>
287 <tr>
288   <td>Ctrl + H</td>
289   <td><str sid="83"></str></td>
290 </tr>
291 <tr>
292   <td>Ctrl + D</td>
293   <td><str sid="219"></str></td>
294 </tr>
295 <tr>
296   <td>Ctrl + S</td>
297   <td><str sid="84"></str></td>
298 </tr>
299 <tr>
300   <td>Ctrl + L</td>
301   <td><str sid="85"></str></td>
302 </tr>
303 </table>
304 </div>
305 </div>
306 </div>
307
308 <div id="info-container" class="main-dialog" tabindex="0">
309   <div class="page-cont panel">
310     <div class="page-header">
311       <span><str sid="3"></str></span>
312       
```

```
313 </div>
314 <div class="page-panel panel">
315     
316     <div class='app-name'>JavaScript LABoratory</div>
317     <div class='app-version'></div>
318     <div class='app-company'>by PR-DC company</div>
319     
320
321     <p><str sid="86"></str></p>
322     <ul>
323         <li><a href="https://www.electronjs.org/">Electron</a></li>
324         <li><a href="https://codemirror.net/">CodeMirror</a></li>
325         <li><a href="https://jquery.com/">JQuery</a></li>
326         <li><a href="https://github.com/adamschwartz/chrome-tabs">chrome-
327             tabs</a></li>
328         <li><a href="https://plotly.com/javascript/">Plotly.js</a></li>
329         <li><a href="https://mathjs.org/">Math.js</a></li>
330         <li><a href="https://www.mathjax.org/">MathJax</a></li>
331         <li><a href="https://highlightjs.org/">highlight.js</a></li>
332         <li><a href="https://github.com/dhcode/big-json-viewer/">big-json-
333             viewer</a></li>
334         <li><a href="https://threejs.org/">Three.js</a></li>
335         <li><a href="https://pyodide.org/en/stable/">pyodide</a></li>
336         <li><a href="https://www.sympy.org/en/index.html">SymPy</a></li>
337         <li><a href="https://www.cgal.org/">CGAL</a></li>
338         <li><a href="https://eigen.tuxfamily.org/index.php?title=Main_Page
339             ">Eigen</a></li>
340         <li><a href="https://github.com/benfred/fmin">fmin</a></li>
341         <li><a href="https://github.com.mozilla/pdf.js">PDF.js</a></li>
342         <li><a href="https://html2canvas.hertzen.com/">html2canvas</a></li
343             >
344             </ul>
345             </div>
346             </div>
347             </div>
348
349 <div id="settings-container" class="main-dialog" tabindex="0">
350     <div class="page-cont options-cont panel">
351         <div class="page-header">
352             <span><str sid="9"></str></span>
353             
355         </div>
356         <div class="page-panel panel">
357             <div class="float-select" title-str="25">
358                 <select name="set-langauge" class="set-langauge">
359                     <option value="en">English</option>
360                     <option value="rs">Srpski</option>
361                     <option value="rsc">Српски</option>
362                 </select>
363                 <label class="float-label" for="set-langauge"><str sid="16"></str>
364                     </label>
365             </div>
366             <div class="float-input" title-str="223">
367                 <input autocomplete="off" type="text" name="N-history-max" class="
368                     </input>
369             </div>
370         </div>
371     </div>
372 
```

```

362           N-history-max" placeholder="20" value="20">
363     <label class="float-label" for="N-history-max"><str sid="223"></
364       str></label>
365   </div>
366   <button class="change-settings"><str sid="57"></str></button>
367   </div>
368 </div>
369 <div id="sandbox-stats-popup" style="display: none;">
370   <div id="sandbox-stats-header">
371     <str sid="91"></str>
372   </div>
373   <div class="sandbox-stats-cont">
374     Required modules
375     <div class="sandbox-stats-val" id="sandbox-required-modules-num">0</
376       div>
377   </div>
378   <div class="sandbox-stats-cont">
379     Promises
380     <div class="sandbox-stats-val" id="sandbox-promises-num">0</div>
381   </div>
382   <div class="sandbox-stats-cont">
383     Timeouts
384     <div class="sandbox-stats-val" id="sandbox-timeouts-num">0</div>
385   </div>
386   <div class="sandbox-stats-cont">
387     Immediates
388     <div class="sandbox-stats-val" id="sandbox-immediates-num">0</div>
389   </div>
390   <div class="sandbox-stats-cont">
391     Intervals
392     <div class="sandbox-stats-val" id="sandbox-intervals-num">0</div>
393   </div>
394   <div class="sandbox-stats-cont">
395     Animation frames
396     <div class="sandbox-stats-val" id="sandbox-animation-frames-num">0</
397       div>
398   </div>
399   <div class="sandbox-stats-cont">
400     Idle callbacks
401     <div class="sandbox-stats-val" id="sandbox-idle-callbacks-num">0</div>
402   </div>
403   <div class="popup-triangle"></div>
404 </div>
405   <div id="status-container">
406     <div id="sandbox-stats-icon" class="ready"></div>
407
408     <div id="status">
409       <str sid="87"></str>
410     </div>
411     <div id="status-icons">
412       </div>

```

```

413   </div>
414   <script>if (typeof module === 'object') {window.module = module; module =
415     undefined;}</script>
416
417   <script type="text/javascript" src="../lib/jquery-3.7.0/jquery-3.7.0.min.js"></script>
418   <script type="text/javascript" src="../lib/jshint-2.13.0/jshint-2.13.0.js">
419     </script>
420   <script type="text/javascript" src="../lib/highlight-11.0.1/highlight-
421     -11.0.1.min.js"></script>
422   <script type="text/javascript" src="../lib/tex-mml-ctml-3.2.0/tex-mml-
423     ctml-3.2.0.js"></script>
424
425   <script type="text/javascript" src="../lib/codemirror-5.49.2/lib/
426     codemirror.js"></script>
427   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/edit/
428     matchbrackets.js"></script>
429   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/lint/
430     lint.js"></script>
431   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/hint/
432     show-hint.js"></script>
433   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
434     searchcursor.js"></script>
435   <script type="text/javascript" src="../lib/codemirror-5.49.2/addon/search/
436     match-highlighter.js"></script>
437   <script type="text/javascript" src="../lib/codemirror-5.49.2/mode/
438     javascript/javascript.js"></script>
439
440   <script type="text/javascript" src="../js/code/custom-javascript-hint.js">
441     </script>
442
443   <script type="text/javascript" src="../js/main/init-main.js"></script>
444 </body>
445 </html>
```

Listing 38 - main.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>Graph - JSLAB | PR-DC</title>
6    <meta http-equiv="Content-Security-Policy" content="script-src * 'self' ,
7      unsafe-inline 'unsafe-eval'; worker-src 'self' blob;" />
8    <link rel="stylesheet" type="text/css" href="../css/basic.css" />
9    <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
10   <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
11   <link rel="stylesheet" type="text/css" href="../css/svg-viewer.css" />
12   <link rel="stylesheet" type="text/css" href="../css/mermaid-graph.css" />
13   <style id="custom-style"></style>
14 </head>
15 <body>
16   <div id="graph" class="graph mermaid prdc-svg-viewer no-layers"></div>
17   <script>if (typeof module === 'object') {window.module = module; module =
18     undefined;}</script>
```

```

18 <script type="text/javascript" src="../lib/jquery-3.7.0/jquery-3.7.0.min.js">
19   </script>
20 <script type="text/javascript" src="../lib/jstree-3.3.17/jstree-3.3.17.min.
js"></script>
21 <script type="text/javascript" src="../lib/d3-7.8.5/d3-7.8.5.min.js"></
script>
22 <script type="text/javascript" src="../lib/PRDC_SVG_VIEWER/PRDC_SVG_VIEWER.
js"></script>
23 <script type="text/javascript" src="../lib/mermaid-11.4.1/mermaid-11.4.1.min
.js"></script>
24 <script type="module">
25   mermaid.initialize({
26     flowchart: {
27       useMaxWidth: false,
28       rankSpacing: 300
29     },
30     startOnLoad: false,
31     securityLevel: 'loose'
32   });
33 </script>
34 </body>
35 </html>
```

Listing 39 - mermaid_graph.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Plotlyjs Plot - JSLAB | PR-DC</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' "
unsafe-inline 'unsafe-eval'; worker-src 'self' blob:;>
7       <link rel="stylesheet" type="text/css" href="../css/basic.css" />
8       <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
9       <link rel="stylesheet" type="text/css" href="../font/LatinModern.css" />
10
11     <style id="dynamic-style-rules"></style>
12   </head>
13   <body>
14     <div id="plot-cont"></div>
15
16     <script type="text/javascript" src="../lib/plotly-2.24.2/plotly-2.24.2.min
.js"></script>
17   </body>
18 </html>
```

Listing 40 - plotlyjs.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Presentation editor - JSLAB | PR-DC</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' "
unsafe-inline 'unsafe-eval'; worker-src 'self' blob:;>
```

```

7   <link rel="stylesheet" type="text/css" href="../../css/tabs.css">
8   <link rel="stylesheet" type="text/css" href="../../css/codemirror-notepadpp-
  theme.css">
9   <link rel="stylesheet" type="text/css" href="../../lib/codemirror-5.49.2/
  addon/fold/foldgutter.css">
10  <link rel="stylesheet" type="text/css" href="../../lib/codemirror-5.49.2/
  addon/lint/lint.css">
11  <link rel="stylesheet" type="text/css" href="../../lib/codemirror-5.49.2/
  addon/hint/show-hint.css">
12  <link rel="stylesheet" type="text/css" href="../../lib/codemirror-5.49.2/lib/
  codemirror.css">
13  <link rel="stylesheet" type="text/css" href="../../css/codemirror-
  presentation-editor-custom.css">
14    <link rel="stylesheet" type="text/css" href="../../css/
  presentation-editor.css" />
15  <link rel="stylesheet" type="text/css" href="../../font/roboto.css" />
16
17  <style id="dynamic-style-rules"></style>
18</head>
19<body>
20  <div id="left-panel">
21    <div class="cell-padding">
22      <div class="panel-container">
23        <div class="panel-title"><str sid="242"></str></div>
24
25        <div id="left-panel-cont">
26          <div id="presentation-title"></div>
27          <div id="webview-wrap">
28            <webview id="preview" useragent="presentation-editor-preview"
              nodeintegration webpreferences="backgroundThrottling=no,
              contextIsolation=no"></webview>
29        </div>
30        <div id="slide-controls">
31          <div id="first-slide" class="button" title="First slide"> </
            div>
32          <div id="prev-slide" class="button" title="Previous"> </div>
33          <input id="set-slide" type="number" min="1" step="1">
34          <span id="total-slides">/ 0</span>
35          <div id="next-slide" class="button" title="Next"> </div>
36          <div id="last-slide" class="button" title="Last slide"> </div>
37        </div>
38      </div>
39    </div>
40  </div>
41
42  <div id="right-panel">
43    <div class="cell-padding">
44      <div class="panel-container">
45        <div class="panel-title"><str sid="243"></str></div>
46
47        <div id="right-panel-cont">
48          <div class="tabs" style="--tab-content-margin: 9px">
49            <div class="tabs-content"></div>
50            

```

```

52      
53      <div class="tabs-bottom-bar"></div>
54    </div>
55    <div id="code"></div>

56    <div id="editor-more-popup" style="display: none;">
57      <ul>
58        <li id="search-dialog-menu" title-str="169"><str sid="168"></str></li>
59        <li id="fold-slides"><str sid="245"></str></li>
60        <li id="unfold-slides"><str sid="246"></str></li>
61      </ul>
62      <div class="popup-triangle"></div>
63    </div>
64  </div>
65  </div>
66  </div>
67</div>
68</div>

69<script>if (typeof module === 'object') {window.module = module; module = undefined;}</script>
70
71<script type="text/javascript" src="../../lib/draggabilly-2.3.0/draggabilly-2.3.0.min.js"></script>
72<script type="text/javascript" src="../../lib/PRDC_TABS/PRDC_TABS.js"></script>
73<script type="text/javascript" src="../../lib/jshint-2.13.0/jshint-2.13.0.js"></script>
74
75<script type="text/javascript" src="../../lib/codemirror-5.49.2/lib/codemirror.js"></script>
76<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/selection/active-line.js"></script>
77<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/foldcode.js"></script>
78<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/foldgutter.js"></script>
79<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/xml-fold.js"></script>
80<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/brace-fold.js"></script>
81<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/indent-fold.js"></script>
82<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/fold/comment-fold.js"></script>
83<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/display/rulers.js"></script>
84<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/edit/matchbrackets.js"></script>
85<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/lint/lint.js"></script>
86<script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/hint/show-hint.js"></script>

```

```

88 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/search/
  searchcursor.js"></script>
89 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/scroll/
  annotatescrollbar.js"></script>
90 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/search/
  matchesonscrollbar.js"></script>
91 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/search/
  jump-to-line.js"></script>
92 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/search/
  match-highlighter.js"></script>
93 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/comment/
  /comment.js"></script>
94 <script type="text/javascript" src="../../lib/codemirror-5.49.2/mode/xml/xml.
  js"></script>
95 <script type="text/javascript" src="../../lib/codemirror-5.49.2/mode/css/css.
  js"></script>
96 <script type="text/javascript" src="../../lib/codemirror-5.49.2/mode/
  javascript/javascript.js"></script>
97 <script type="text/javascript" src="../../lib/codemirror-5.49.2/mode/
  htmlmixed/htmlmixed.js"></script>

98
99 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/hint/
  xml-hint.js"></script>
100 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/hint/
  javascript-hint.js"></script>
101 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/hint/
  css-hint.js"></script>
102 <script type="text/javascript" src="../../lib/codemirror-5.49.2/addon/hint/
  html-hint.js"></script>
103 <script type="text/javascript" src="../js/code/dialog-search.js"></script>
104
105 <script type="text/javascript" src="../../lib/jquery-3.7.0/jquery-3.7.0.min.
  js"></script>
106 <script type="text/javascript" src="../js/windows/presentation-editor.js">
  </script>
107 </body>
108 </html>
```

Listing 41 - presentation-editor.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Presentation - JSLAB | PR-DC</title>
6   <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '
  unsafe-inline 'unsafe-eval'; worker-src 'self' blob:;" />
7
8   <!-- CSS files begin -->
9   %presentation_stylesheets%
10  <link rel="stylesheet" type="text/css" href=".//res/internal/presentation.css
    " />
11  <link rel="stylesheet" type="text/css" href=".//main.css" />
12  <!-- CSS files end -->
13
14  <style id="dynamic-style-rules"></style>
```

```

15 </head>
16 <body>
17 <div id="slides-cont">
18 <!-- Slides begin -->
19 <slide>
20   Slide <span class="slide-number">${{presentation.slideNumber()}} / ${{
21     presentation.slideCount()}}</span>
22 </slide>
23
24 <slide>
25   Slide <span class="slide-number">${{presentation.slideNumber()}} / ${{
26     presentation.slideCount()}}</span>
27 </slide>
28
29 <slide>
30   Slide <span class="slide-number">${{presentation.slideNumber()}} / ${{
31     presentation.slideCount()}}</span>
32 </slide>
33 <!-- Slides end -->
34 </div>
35
36 <!-- JS files begin -->
37 %presentation_scripts%
38 <script type="text/javascript" src=". / res/internal/presentation.js"></script>
39 <script type="text/javascript" src=". / main.js"></script>
40 <!-- JS files end -->
41 </body>
42 </html>
```

Listing 42 - presentation.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5   </head>
6   <body>
7     <script type="text/javascript" src=". / lib/luxon-3.4.4/luxon-3.4.4.min.js">
8       </script>
9     <script type="text/javascript" src=". / lib/math-11.8.2/math-11.8.2.min.js">
10      </script>
11     <script type="text/javascript" src=". / lib/sprintf-1.1.3/sprintf-1.1.3.min.js">
12       </script>
13     <script type="text/javascript" src=". / lib/Cesium-1.124/Cesium.js"></
14       script>
15
16     <script>window. process. browser = 'Electron';</script>
17     <script type="text/javascript" src=". / lib/sympy-0.26.2/pyodide.js"></
18       script>
19     <script type="text/javascript" src=". / node_modules/fmin/build/fmin.min.js">
20       </script>
21     <script type="text/javascript" src=". / js/sandbox/init-sandbox-worker.js">
22       </script>
23   </body>
```

```
17 </html>
```

Listing 43 - sandbox-worker.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
6   </head>
7   <body>
8     <script type="text/javascript" src="../lib/luxon-3.4.4/luxon-3.4.4.min.js"></script>
9     <script type="text/javascript" src="../lib/math-11.8.2/math-11.8.2.min.js"></script>
10    <script type="text/javascript" src="../lib/sprintf-1.1.3/sprintf-1.1.3.min.js"></script>
11    <script type="text/javascript" src="../lib/Cesium-1.124/Cesium.js"></script>
12
13  <script>window.process.browser = 'Electron';</script>
14  <script type="text/javascript" src="../lib/sympy-0.26.2/pyodide.js"></script>
15  <script type="text/javascript" src="../node_modules/fmin/build/fmin.min.js"></script>
16  <script type="text/javascript" src="../js/sandbox/init-sandbox.js"></script>
17 </body>
18 </html>
```

Listing 44 - sandbox.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Serial Terminal - JSLAB | PR-DC</title>
6     <meta http-equiv="Content-Security-Policy" content="script-src * 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;" />
7     <link rel="stylesheet" type="text/css" href="../css/basic.css" />
8     <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
9     <style>
10       html, body {
11         margin: 0;
12         padding: 0;
13         height: 100%;
14       }
15
16       body {
17         height: calc(100% - 6px);
18       }
19
20       .clear {
21         clear: both;
22     }
```



```
23
24 .container {
25   display: flex;
26   flex-direction: column;
27   height: 100%;
28 }
29
30 #messages-container {
31   flex: 1;
32   overflow-y: auto;
33   padding: 10px;
34   border-bottom: 1px solid #ccc;
35 }
36
37 #input-container {
38   display: flex;
39   align-items: center;
40   padding: 10px;
41   background: #eee;
42 }
43
44 #input-container textarea {
45   flex: 1;
46   margin-right: 40px;
47   padding: 5px;
48   font-family: inherit;
49   font-size: 14px;
50   resize: none;
51   border-radius: 5px;
52   padding-bottom: 0px;
53 }
54
55 #send-button {
56   position: absolute;
57   right: 10px;
58   z-index: 3;
59   background: #f7df1e;
60   border-radius: 100%;
61   width: 30px;
62   height: 30px;
63   cursor: pointer;
64   bottom: 20px;
65 }
66
67 #send-button:hover {
68   opacity: 0.6;
69 }
70
71 #send-button img {
72   width: 18px;
73   padding: 6px;
74 }
75
76 .options .options-right {
77   float: right;
```

```
78     }
79
80     .options i#settings {
81       background: url(../img/settings.svg) no-repeat center;
82     }
83
84     .options i#timestamp {
85       background: url(../img/timestamp.svg) no-repeat center;
86     }
87
88     .options i#autoscroll {
89       background: url(../img/autoscroll.svg) no-repeat center;
90     }
91
92     .options i#clear {
93       background: url(../img/clear.svg) no-repeat center;
94     }
95
96     .options i#log {
97       background: url(../img/save-log.svg) no-repeat center;
98     }
99
100    .options i#to-bottom {
101      background: url(../img/to-bottom.svg) no-repeat center;
102    }
103
104    .options i {
105      width: 16px;
106      height: 18px;
107      display: block;
108      background-size: 16px !important;
109      float: left;
110      clear: none;
111      padding: 3px 5px;
112      opacity: 0.3;
113      user-select: none;
114      -webkit-user-drag: none;
115    }
116
117    .options i:hover {
118      opacity: 1;
119      cursor: pointer;
120    }
121
122    .options i.active {
123      opacity: 0.8;
124    }
125
126    #save-log, #change-settings {
127      margin-bottom: 0px;
128    }
129
130    #terminal-options {
131      margin: 5px 10px;
132      margin-bottom: 0px;
```



```
133     color: #666;
134     padding-bottom: 3px;
135     border-bottom: 1px solid #f7df1e;
136     font-weight: 300;
137     font-size: 14px;
138     user-select: none;
139     -webkit-user-drag: none;
140     user-drag: none;
141     white-space: nowrap;
142     overflow: hidden;
143     text-overflow: ellipsis;
144     width: calc(100% - 20px);
145     right: 0;
146 }
147
148 #messages-container > div {
149     padding-bottom: 5px;
150     padding-top: 5px;
151     line-height: 22px;
152     padding-left: 105px;
153     padding-right: 10px;
154     position: relative;
155     white-space: pre-wrap;
156     min-height: 32px;
157 }
158
159 #messages-container.no-timestamp > div {
160     padding-left: 5px;
161 }
162
163 #messages-container div:hover {
164     background-color: #f9f9f9;
165 }
166
167 #messages-container.no-timestamp div span.timestamp {
168     display: none;
169 }
170
171 #messages-container div:hover span.timestamp {
172     color: #2e85c7
173 }
174
175 #messages-container div span.timestamp {
176     color: #999;
177     display: block;
178     position: absolute;
179     left: 0px;
180 }
181
182 #terminal-title {
183     float: left;
184     padding-top: 5px;
185 }
186
187 </style>
```

```

188 </head>
189 <body>
190   <div class="container">
191     <div id="terminal-options" class="options ui">
192       <div id="terminal-title"></div>
193       <div class="options-right">
194         <i id="settings" title-str="40"></i>
195         <i id="timestamp" class="active" title-str="41"></i>
196         <i id="autoscroll" class="active" title-str="42"></i>
197         <i id="clear" title-str="43"></i>
198         <i id="log" title-str="44"></i>
199         <i id="to-bottom" title-str="45"></i>
200         <br class="clear">
201       </div>
202     </div>
203
204   <div id="messages-container" class="panel ui"></div>
205
206   <div id="input-container">
207     <textarea id="message-input" str="234" placeholder=""></textarea>
208     <div id="send-button">
209       
210     </div>
211   </div>
212
213   <div id="settings-dialog" class="terminal-dialog options-panel panel ui" tabindex="0">
214     <div class="options-cont ui">
215       <div class="options-header ui"><span><str sid="56"></str></span>
216       
217     </div>
218     <div class="float-input ui" title-str="17">
219       <input autocomplete="off" type="text" name="N-messages-max" id="N-
220         messages-max" placeholder="Infinity" value="">
221       <label class="float-label" for="N-messages-max"><str sid="17"></str></
222         label>
223     </div>
224     <button class="change-settings ui blue"><str sid="57"></str></button>
225   </div>
226
227   <div id="log-dialog" class="terminal-dialog options-panel panel ui" tabindex="0">
228     <div class="options-cont ui">
229       <div class="options-header ui">
230         <span><str sid="58"></str></span>
231         
232       </div>
233       <label class="checkcont ui"><str sid="59"></str><input id="write-
234         timestamps" type="checkbox" name="write-timestamps" value="1"
235           checked><span class="checkmark"></span></label>
236         <button id="save-log" class="ui blue"><str sid="60"></str></button>
237       </div>
238     </div>
239   </div>

```

```
237 </body>
238 </html>
```

Listing 45 - serial_terminal.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>THREE - JSLAB | PR-DC</title>
6    <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '' unsafe-inline ''unsafe-eval'; worker-src 'self' blob:;" />
7    <link rel="stylesheet" type="text/css" href="../css/three.css" />
8    <link rel="stylesheet" type="text/css" href="../font/roboto.css" />
9  </head>
10 <body>
11
12   <div id="scene-cont"></div>
13
14   <script type="text/javascript" src="../lib/hammer-2.0.8/hammer.min.js"></script>
15   <script type="text/javascript" src="../lib/anime-3.2.1/anime-3.2.1.min.js"></script>
16   <script type="text/javascript" src="../lib/tween.js-23.1.1/tween-23.1.1.js"></script>
17   <script type="text/javascript" src="../lib/inflate-0.3.1/inflate-0.3.1.min.js"></script>
18
19   <script>if (typeof module === 'object') { window.module = module; module = undefined; }</script>
20
21   <script type="importmap">
22   {
23     "imports": {
24       "three": "../lib/three.js-r162/build/three.module.js",
25       "three/addons/": "../lib/three.js-r162/examples/jsm/"
26     }
27   }
28   </script>
29 </body>
30 </html>
```

Listing 46 - three.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>URL - JSLAB | PR-DC</title>
6    <meta http-equiv="Content-Security-Policy" content="script-src * 'self' '' unsafe-inline ''unsafe-eval'; worker-src 'self' blob:;" />
7    <style>
8      html, body {
9        height: 100%;
10       margin: 0;
11     }
```

```

12      webview {
13          position: fixed ;
14          inset: 0;
15      }
16      </style>
17  </head>
18  <body>
19      <webview id="webview" nodeintegration webpreferences="backgroundThrottling=
19          no, contextIsolation=no"></webview>
20  </body>
21 </html>
```

Listing 47 - url.html

6 js

```

1  /**
2   * @file Javascript helper functions
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  var { shell } = require('electron');
9
10 /**
11  * Call function when document is ready
12  * @param {function} fn - function which is called when document is ready.
13  */
14 global.ready = function(fn) {
15     if(document.readyState != 'loading') {
16         fn();
17     } else {
18         document.addEventListener('DOMContentLoaded', fn);
19     }
20 };
21
22 /**
23  * Prevents redirect
24  */
25 global.preventDefault = function() {
26     var links = $('a');
27     links.each(function() {
28         if(!$(this).hasClass('external-link')) {
29             $(this).addClass('external-link');
30             $(this).click(function(e) {
31                 e.preventDefault();
32                 shell.openExternal(e.target.href);
33                 return false;
34             });
35         });
36     });
37};
```

```

38
39 /**
40 * Get process arguments
41 */
42 if(process.type === 'browser' || global.is_worker) {
43     global.process_arguments = process.argv;
44 } else {
45     var { ipcRenderer } = require('electron');
46     global.process_arguments =
47         ipcRenderer.sendSync("sync-message", "get-process-arguments");
48 }

```

Listing 48 - helper.js

```

1 /**
2 * @file Init config
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 * @version 0.0.1
7 */
8 "use strict";
9
10 const { PRDC_APP_CONFIG } = require("../config/config.js");
11
12 // Global variables
13 global.config = new PRDC_APP_CONFIG();
14
15 // Conditional variables
16 if(typeof global.process_arguments !== 'undefined' && Array.isArray(global.
17     process_arguments)) {
18     var args = global.process_arguments.map(function(e) { return e.toLowerCase()
19         ; });
20     if(args.includes("--debug-app")) {
21         global.config.DEBUG = true;
22     }
23     if(args.includes("--test-app")) {
24         global.config.TEST = true;
25     }
26     if(args.includes("--sign-build")) {
27         global.config.SIGN_BUILD = true;
28     }
29 }

```

Listing 49 - init-config.js

```

1 /**
2 * @file JSLAB init app
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7 "use strict";
8
9 const { app } = require('electron');
10 global.app_path = app.getAppPath().replace(/\js\\?$/ , '' );

```

```

11
12 const helper = require("./helper.js");
13 require("./init-config.js");
14
15 const { PRDC_JSLAB_MAIN } = require("./main");
16
17 // Start main
18 const main = new PRDC_JSLAB_MAIN();

```

Listing 50 - init.js

```

1 /**
2  * @file JSLAB language module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const fs = require('fs');
9 const Store = require('electron-store');
10
11 const store = new Store();
12
13 /**
14  * Class for JSLAB language.
15 */
16 class PRDC_JSLAB_LANGUAGE {
17
18 /**
19  * Create JSLAB language object.
20 */
21 constructor() {
22   // Variables
23   var obj = this;
24
25   this.lang_index = store.get('lang_index');
26   if(!this.lang_index) {
27     this.lang_index = 0;
28   }
29
30   this.onLanguageChange = function() {};
31
32   var style = document.createElement('style');
33   document.head.appendChild(style);
34   this.lang_styles = style.sheet;
35
36   // Get language strings
37   this.s = JSON.parse(fs.readFileSync(app_path + '/config/lang.json'));
38
39   document.querySelectorAll('str').forEach(function(el) {
40     var id = el.getAttribute('sid');
41     el.innerHTML = obj.string(id);
42   });
43
44   this.lang = config.langs[this.lang_index];
45   this.set(this.lang);

```

```

46   }
47
48  /**
49   * Sets the application's current language and updates the UI accordingly.
50   * Saves the new language preference for future sessions.
51   * @param {string} lang The language code to set as the current language.
52   */
53 set(lang) {
54   var idx = config.langs.findIndex(function(e) {
55     return e === lang;
56   });
57   if(idx >= 0) {
58     this.lang = lang;
59     this.lang_index = idx;
60     if(this.lang_styles.cssRules.length) {
61       this.lang_styles.deleteRule(0);
62     }
63     this.lang_styles.insertRule("lang."+lang+" { display: initial }", 0);
64     this.update('html', false);
65
66     // Save language
67     store.set('lang_index', this.lang_index);
68     this.onLanguageChange(lang);
69   }
70 }
71
72 /**
73  * Dynamically updates text strings within the specified HTML container to
74  * the current language.
75  * Can optionally update placeholders and titles for input and option
76  * elements.
77  * @param {String} cont The selector for the container whose text strings
78  * will be updated. Defaults to 'html'.
79  * @param {boolean} flag If true, updates the container and child elements
80  * with dynamic language strings.
81  */
82 update(cont = 'html', flag = true) {
83   var obj = this;
84
85   if(flag) {
86     document.querySelectorAll(cont + ' str').forEach(function(el) {
87       var id = el.getAttribute('sid');
88       el.innerHTML = obj.string(id);
89     });
90   }
91
92   document.querySelectorAll(cont + ' textarea[str]').forEach(function(el) {
93     var id = el.getAttribute('str');
94     if(id in obj.s) {
95       el.setAttribute('placeholder', obj.s[id][obj.lang]);
96     }
97   });
98
99   document.querySelectorAll(cont + ' input[str]').forEach(function(el) {
100    var id = el.getAttribute('str');
101  });

```

```

97     if(id in obj.s) {
98         el.setAttribute('placeholder', obj.s[id][obj.lang]);
99     }
100 });
101
102 document.querySelectorAll(cont + ' option[' + str + ']).forEach(function(el) {
103     var id = el.getAttribute('str');
104     if(id in obj.s) {
105         el.textContent = obj.s[id][obj.lang];
106     }
107 });
108
109 document.querySelectorAll(cont + ' [title=' + str + ']).forEach(function(el) {
110     var id = el.getAttribute('title');
111     if(id in obj.s) {
112         el.setAttribute('title', obj.s[id][obj.lang]);
113     }
114 });
115 }
116
117 /**
118 * Retrieves the specified language string in all available languages,
119 * wrapped in language-specific <lang> tags.
120 * @param {number} id The identifier of the string to retrieve.
121 * @returns {HTML} HTML string containing the text in all available
122 * languages.
123 */
124 string(id) {
125     var obj = this;
126     var msg = '';
127     if(id in this.s) {
128         config.langs.forEach(function(lang) {
129             msg += '<lang class="' + lang + '">' + obj.s[id][lang] + '</lang>';
130         });
131     } else {
132         config.langs.forEach(function(lang) {
133             msg += '<lang class="' + lang + '"></lang>';
134         });
135     }
136     return msg;
137 }
138
139 /**
140 * Retrieves the current language string for the specified identifier.
141 * @param {number} id The identifier of the string to retrieve.
142 * @returns {String} The string corresponding to the current language.
143 */
144 currentString(id) {
145     if(id in this.s) {
146         return this.s[id][this.lang];
147     } else {
148         return '';
149     }

```

```

150  /**
151   * Sets a callback function to be executed when the language is changed.
152   * @param {Function} callback The callback function to be executed on
153   * language change.
154   */
155  setOnLanguageChange(callback) {
156    if(typeof callback == 'function') {
157      this.onLanguageChange = callback;
158    }
159  }
160
161 exports.PRDC_JSLAB_LANGUAGE = PRDC_JSLAB_LANGUAGE;

```

Listing 51 - language.js

```

1  /**
2   * @file JSLAB main script
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  const { app, BrowserWindow, ipcMain, dialog, powerSaveBlocker, shell,
9       MenuItem, desktopCapturer, screen, webContents } = require('electron');
10
11 const contextMenu = require('electron-context-menu');
12 const fs = require('fs');
13 const os = require('os');
14 const Store = require('electron-store');
15
16 const { PRDC_APP_LOGGER } = require('../lib/PRDC_APP_LOGGER/PRDC_APP_LOGGER');
17
18 /**
19  * Class for flight control app.
20  */
21 class PRDC_JSLAB_MAIN {
22
23 /**
24  * Create app.
25  */
26 constructor() {
27   var obj = this;
28
29   this.store = new Store();
30   this.debounce_save_win_time = config.WIN_SAVE_DEBOUNCE_TIME;
31   this.debounce_save_win_bounds = [];
32
33   this.heartbeat_interval;
34   this.win_main;
35   this.win_editor;
36   this.win_sandbox;
37   this.win_opacity = 0;
38   this.editor_close_ready = false;
39   this.stop_loop_in = false;

```

```

41     this.app_icon = app_path + '/icons/icon.ico'; // png to ico https://icoconvert.com/
42     this.is_app_quitting = false;
43     this.known_paths = [ 'home', 'appData', 'userDat', 'sessionData',
44         'temp', 'exe', 'module', 'desktop', 'documents', 'downloads',
45         'music', 'pictures', 'videos', 'recent', 'logs', 'crashDumps' ];
46
47     if(os.platform() == 'linux') {
48         this.app_icon = app_path + '/icons/icon.png';
49     }
50
51     // Create folder for app
52     this.app_folder = app.getPath('documents')+ '\\ '+app.getName();
53     if(!fs.existsSync(this.app_folder)) {
54         fs.mkdirSync(this.app_folder);
55     }
56
57     // Start log
58     this.log_file = this.app_folder+'\\ '+app.getName()+' .log';
59     this.app_logger = new PRDC_APP_LOGGER(this.log_file);
60
61     if(config.REPORT_CRASH) {
62         const Bugsnag = require('@bugsnag/electron');
63         Bugsnag.start({ apiKey: config.BUGSNAG_API_KEY });
64     }
65
66     // Context menu
67     contextMenu({
68         append: function(default_actions, p, win) {
69             return [
70                 new MenuItem({
71                     label: 'Toggle Comment',
72                     click: function() {
73                         obj.win_editor.send('EditorWindow', 'toggle-comment');
74                     },
75                     visible: p.formControlType == 'text-area' &&
76                         p.pageURL.endsWith('/editor.html')
77                 }),
78                 new MenuItem({
79                     label: 'Go To Code',
80                     click: function() {
81                         win.webContents.send('PresentationEditorWindow', 'go-to-code');
82                     },
83                     visible: p.pageURL.endsWith('/presentation-editor.html') &&
84                         p.frameURL != p.pageURL
85                 }),
86                 new MenuItem({
87                     label: 'Go To Slide',
88                     click: function() {
89                         win.webContents.send('PresentationEditorWindow', 'go-to-slide');
90                     },
91                     visible: p.formControlType == 'text-area' &&
92                         p.titleText == 'html' &&
93                         p.pageURL.endsWith('/presentation-editor.html')
94                 })
95             ];
96         }
97     });

```

```

94
95   }) ,
96   new MenuItem({
97     role: 'selectAll',
98     label: 'Select All',
99     visible: p.editFlags.canSelectAll
100  }) ,
101  new MenuItem({
102    role: 'delete',
103    label: 'Delete',
104    visible: p.editFlags.canDelete
105  })
106 },
107 showLookUpSelection: false,
108 showSearchWithGoogle: false,
109 showCopyImage: false,
110 showInspectElement: false
111 });
112
113 // Command line switches
114 app.commandLine.appendSwitch('disable-renderer-backgrounding');
115 app.commandLine.appendSwitch('disable-background-timer-throttling');
116 app.commandLine.appendSwitch('disable-backgrounding-occluded-windows');
117 app.commandLine.appendSwitch("disable-http-cache");
118 app.commandLine.appendSwitch('max-active-webgl-contexts', config.
119   MAX_ACTIVE_WEBGL_CONTEXTS);
120
121 // Prevent sleep
122 powerSaveBlocker.start('prevent-app-suspension');
123
124 // This method will be called when Electron has finished
125 // initialization and is ready to create browser windows.
126 // Some APIs can only be used after this event occurs.
127 app.whenReady().then(function() {
128   obj.createWindows();
129
130   app.on('activate', function() {
131     // On macOS it's common to re-create a window in the app when the
132     // dock icon is clicked and there are no other windows open.
133     if(BrowserWindow.getAllWindows().length === 0) obj.createWindows();
134   });
135
136   // Quit when all windows are closed, except on macOS. There, it's common
137   // for applications and their menu bar to stay active until the user quits
138   // explicitly with Cmd + Q.
139   app.on('window-all-closed', function() {
140     if(process.platform !== 'darwin') app.quit();
141   });
142   app.allowRendererProcessReuse = false;
143 }
144
145 /**
146 * Create all windows
147 */

```

```
148 createWindows() {
149     // Create the main window
150     this.creatMainWindow();
151
152     // Create the sandbox window
153     this.createSandboxWindow();
154
155     // Create the editor window
156     this.createEditorWindow();
157
158     // Set handlers
159     this.setHandlers();
160
161     // Handle IPC messages
162     this.handleMessages();
163 }
164
165 /**
166 * Create main window
167 */
168 creatMainWindow() {
169     var obj = this;
170     var options = {
171         title: 'JSLAB',
172         minWidth: 720,
173         minHeight: 500,
174         icon: this.app_icon, // png to ico https://icoconvert.com/
175         show: false,
176         frame: false,
177         backgroundColor: '#ffffff',
178         opacity: this.win_opacity,
179         webPreferences: {
180             nodeIntegration: true,
181             nodeIntegrationInWorker: true,
182             contextIsolation: false,
183             backgroundThrottling: false
184         }
185     };
186     options = this.getWindowBounds('mainWinBounds', options);
187     this.win_main = new BrowserWindow(options);
188
189     // Hide menu
190     this.win_main.setMenu(null);
191
192     // Maximize
193     if(options.maximize) {
194         this.win_main.maximize();
195     }
196
197     // and load the main.html of the app
198     this.win_main.loadFile(app_path + '/html/main.html');
199
200     // Events
201     this.win_main.on("resize", function() { obj.saveWindowBounds(obj.win_main,
202         'mainWinBounds'); });
203 }
```

```

202   this.win_main.on("move", function() { obj.saveWindowBounds(obj.win_main, 'mainWinBounds'); });
203   this.win_main.on("maximize", function() { obj.saveWindowBounds(obj.win_main, 'mainWinBounds'); });
204   this.win_main.on("unmaximize", function() { obj.saveWindowBounds(obj.win_main, 'mainWinBounds'); });

205 // Show window when ready
206 this.win_main.once('ready-to-show', function() {
207   obj.win_main.show();
208   if(config.DEBUG) {
209     obj.openDevTools(obj.win_main);
210   }
211 });
212 });

213 this.win_main.webContents.on('render-process-gone', function(event,
214   details) {
215   obj.app_logger.logMessage(config.DEBUG_RENDER_GONE_ERROR, config,
216     LOG_RENDER_GONE_ERROR, config.LOG_CODES['render-gone-error'],
217     Render gone error, 'Main window render gone error:' + JSON.
218     stringify(event) + ' ' + JSON.stringify(details));
219   app.exit();
220   app.relaunch();
221 });
222 });

223 // Close all windows
224 this.win_main.on('close', function(e) {
225   e.preventDefault();
226   obj.win_main.webContents.executeJavaScript('win_main.close();');
227 });
228 });

229 /**
230 * Create sandbox window
231 */
232 createSandboxWindow() {
233   var obj = this;
234   this.win_sandbox = new BrowserWindow({
235     title: 'Sandbox | JSLAB',
236     minWidth: 720,
237     minHeight: 500,
238     icon: this.app_icon, // png to ico https://icoconvert.com/
239     show: false,
240     backgroundColor: '#ffffff',
241     opacity: this.win_opacity,
242     webPreferences: {
243       nodeIntegration: true,
244       nodeIntegrationInWorker: true,
245       contextIsolation: false,
246       backgroundThrottling: false
247     }
248   });
249 });

250 // Hide menu
251 this.win_sandbox.setMenu(null);

```

```

250
251 // and load the index.html of the app
252 this.win_sandbox.loadFile(app_path + '/html/sandbox.html');
253
254 if(config.DEBUG) {
255   // Show dev tools
256   this.openDevTools(this.win_sandbox);
257 }
258
259 // Sub windows
260 this.win_sandbox.webContents.setWindowOpenHandler(function(handler) {
261   return {
262     action: 'allow',
263     overrideBrowserWindowOptions: {
264       minWidth: 250,
265       minHeight: 50,
266       icon: obj.app_icon, // png to ico https://icoconvert.com/
267       show: false,
268       backgroundColor: '#ffffff',
269       opacity: obj.win_opacity,
270       webPreferences: {
271         nodeIntegration: true,
272         nodeIntegrationInWorker: true,
273         contextIsolation: false,
274         backgroundThrottling: false,
275         webviewTag: true
276       }
277     }
278   };
279 });
280
281 this.sandbox_sub_wins = {};
282 this.win_sandbox.webContents.on('did-create-window', function(sub_win,
283   details) {
284   var wid = details.frameName;
285   obj.sandbox_sub_wins[wid] = sub_win;
286
287   // Hide menu
288   sub_win.setMenu(null);
289
290   // Show window
291   sub_win.once('ready-to-show', function() {
292     sub_win.show();
293     sub_win.focus();
294     obj.fadeWindowIn(sub_win, 0.1, 10);
295
296     if(config.DEBUG) {
297       obj.openDevTools(sub_win);
298     }
299   });
300
301   sub_win.webContents.on('did-attach-webview', (_e, wc) => {
302     if(wc.getUserAgent() == 'presentation-editor-preview') {
303       contextMenu({
304         window: wc,

```

```

304     append: function( default _ actions , p , win) {
305       return [
306         new MenuItem({
307           label: 'Go To Slide',
308           click: function() {
309             sub _ win . webContents . send( 'PresentationEditorWindow' , 'go-
310               to - code' );
311             },
312             visible: true
313           })
314         }
315       );
316     }
317   );
318
319   // Close all windows
320   sub _ win . on( 'close' , function(e) {
321     e . preventDefault();
322     obj . win _ sandbox . webContents . executeJavaScript( 'jsl . windows .
323       _closedWindow( '+wid+' );' );
324     sub _ win . hide();
325     sub _ win . onClose = true;
326   });
327
328
329   this . win _ sandbox . webContents . on( 'render - process - gone' , function(event ,
330     details) {
331     obj . app _ logger . logMessage( config . DEBUG_RENDER_GONE_ERROR , config .
332       LOG_RENDER_GONE_ERROR , config . LOG_CODES[ 'render-gone-error' ] , ,
333       Render gone error , 'Sandbox window render gone error:' + JSON .
334       stringify( event ) + ' ' + JSON . stringify( details ) );
335     app . exit();
336     app . relaunch();
337   });
338
339
340   this . win _ sandbox . webContents . on( 'did - finish - load' , function() {
341     if (!obj . store . get( 'shown - getting - started' )) {
342       obj . store . set( 'shown - getting - started' , 1 );
343       obj . win _ sandbox . webContents . executeJavaScript( 'openDocumentation("
344         about") ;' );
345     }
346     if (obj . sandbox _ reload _ active) {
347       obj . win _ main . webContents . executeJavaScript( 'win_main . onSandboxReset() ;
348         ');
349       obj . sandbox _ reload _ active = false;
350     }
351   });
352
353
354   /**
355    * Create editor window
356    */
357   createEditorWindow() {

```

```
351     var obj = this;
352     var options = {
353         title: 'Editor | JSLAB',
354         minWidth: 720,
355         minHeight: 500,
356         icon: this.app_icon, // png to ico https://icoconvert.com/
357         show: false,
358         frame: false,
359         backgroundColor: '#ffffff',
360         opacity: this.win_opacity,
361         webPreferences: {
362             nodeIntegration: true,
363             nodeIntegrationInWorker: true,
364             contextIsolation: false,
365             backgroundThrottling: false
366         }
367     };
368     options = this.getWindowBounds('editorWinBounds', options);
369     this.win_editor = new BrowserWindow(options);
370
371     // Hide menu
372     this.win_editor.setMenu(null);
373
374     // Maximize
375     if(options.maximize) {
376         this.win_editor.maximize();
377     }
378
379     // Hide menu
380     this.win_editor.setMenu(null);
381
382     // and load the index.html of the app
383     this.win_editor.loadFile(app_path + '/html/editor.html');
384
385     // Maximize
386     if(options.maximize) {
387         this.win_editor.maximize();
388     }
389
390     // Events
391     this.win_editor.on("resize", function() { obj.saveWindowBounds(obj.
392         win_editor, 'editorWinBounds'); });
393     this.win_editor.on("move", function() { obj.saveWindowBounds(obj.
394         win_editor, 'editorWinBounds'); });
395     this.win_editor.on("maximize", function() { obj.saveWindowBounds(obj.
396         win_editor, 'editorWinBounds'); });
397     this.win_editor.on("unmaximize", function() { obj.saveWindowBounds(obj.
398         win_editor, 'editorWinBounds'); });
399
400     // Hide window when ready
401     this.win_editor.hide();
402
403     this.win_editor.on('close', function(e) {
404         if(!obj.is_app_quitting) {
405             e.preventDefault();
```

```

402         obj.fadeWindowOut(obj.win_editor, 0.1, 10);
403         setTimeout(function() {
404             obj.win_editor.hide();
405         }, 100);
406     } else if (!obj.editor_close_ready) {
407         e.preventDefault();
408     }
409 });
410
411 this.win_editor.webContents.on('render-process-gone', function(event,
412     details) {
413     obj.app_logger.logMessage(config.DEBUG_RENDER_GONE_ERROR, config.
414         LOG_RENDER_GONE_ERROR, config.LOG_CODES['render-gone-error'], ,
415         'Render gone error', 'Editor window render gone error:' + JSON.
416         stringify(event) + ' ' + JSON.stringify(details));
417 });
418
419 /**
420 * Handle IPC messages
421 */
422 handleMessages() {
423     var obj = this;
424
425     // For MainProcess
426     ipcMain.handle('get-completions', function(e, data) {
427         obj.win_sandbox.send('SandboxWindow', 'get-completions', data);
428         return new Promise(function(resolve) {
429             ipcMain.once('completions-' + data[0], function(e, data) {
430                 resolve(data);
431             });
432         });
433     });
434
435     ipcMain.handle('dialog', function(e, method, params) {
436         if (!params || params && !params.hasOwnProperty('icon')) {
437             if (!params) {
438                 params = {};
439             }
440             params.icon = obj.app_icon;
441         }
442         return dialog[method](params);
443     });
444
445     ipcMain.on('get-desktop-sources', async function(e) {
446         e.returnValue = await desktopCapturer.getSources({ types: ['screen', ,
447             'window'] });
448     });
449
450     ipcMain.handle('print-to-pdf', async function(e, options) {
451         return await e.sender.printToPDF(options);
452     });
453
454     ipcMain.handle('print-sub-win-to-pdf', async function(e, wid, options) {
455         return await obj.sandbox_sub_wins[wid].webContents.printToPDF(options);
456     });

```

```
452 });
453
454 ipcMain.on('dialog', function(e, method, params) {
455   if(!params || params && !params.hasOwnProperty('icon')) {
456     if(!params) {
457       params = {};
458     }
459     params.icon = obj.app_icon;
460   }
461   e.returnValue = dialog[method](params);
462 });
463
464 ipcMain.on('sync-message', function(e, action, data) {
465   var retval;
466   switch(action) {
467     case 'get-app':
468       retval = { 'name': app.getName(), 'version': app.getVersion(), 'path':
469                 app_path, 'exe_path': app.getPath('exe') };
470       break;
471     case 'get-app-name':
472       retval = app.getName();
473       break;
474     case 'get-app-path':
475       retval = app_path;
476       break;
477     case 'get-app-version':
478       retval = app.getVersion();
479       break;
480     case 'get-platform':
481       retval = os.platform();
482       break;
483     case 'get-debug-flag':
484       retval = config.DEBUG;
485       break;
486     case 'get-path':
487       if(obj.known_paths.includes(data)) {
488         retval = app.getPath(data);
489       } else if(data === 'root') {
490         retval = app_path;
491       } else if(data === 'includes') {
492         retval = app_path + '\\includes';
493       } else {
494         retval = true;
495       }
496       break;
497     case 'get-log-file':
498       retval = obj.log_file;
499       break;
500     case 'is-maximized-win':
501       retval = BrowserWindow.fromWebContents(e.sender).isMaximized();
502       break;
503     case 'check-stop-loop':
504       retval = obj.stop_loop_in;
505       break;
506     case 'reset-stop-loop':
```

```
506     obj.stopLoopIn = false;
507     retval = true;
508     break;
509   case 'get-process-arguments':
510     retval = process.argv;
511     break;
512   case 'call-sub-win-method':
513     var [id, method, ...args] = data;
514     if(obj.sandbox_sub_wins.hasOwnProperty(id)) {
515       retval = obj.sandbox_sub_wins[id][method](...args);
516     } else {
517       retval = false;
518     }
519     break;
520   case 'open-sub-win-devtools':
521     if(obj.sandbox_sub_wins.hasOwnProperty(data)) {
522       obj.openDevTools(obj.sandbox_sub_wins[data]);
523       retval = true;
524     } else {
525       retval = false;
526     }
527     break;
528   case 'get-sub-win-source-id':
529     if(obj.sandbox_sub_wins.hasOwnProperty(data)) {
530       retval = obj.sandbox_sub_wins[data].getMediaSourceId();
531     } else {
532       retval = false;
533     }
534     break;
535   case 'reset-app':
536     app.exit();
537     app.relaunch();
538     break;
539   case 'reset-sandbox':
540     obj.sandbox_reload_active = true;
541     obj.win_sandbox.destroy();
542     Object.keys(obj.sandbox_sub_wins).forEach(function(wid) {
543       obj.sandbox_sub_wins[wid].destroy();
544       delete obj.sandbox_sub_wins[wid];
545     });
546     obj.createSandboxWindow();
547     break;
548   default:
549     retval = true;
550     break;
551   }
552   e.returnValue = retval;
553 });
554
555 ipcMain.on('MainProcess', function(e, action, data) {
556   switch(action) {
557     case 'show-dev-tools':
558       // Show DevTools
559       obj.openDevTools(BrowserWindow.fromWebContents(e.sender));
560       break;
```

```
561     case 'show-sandbox-dev-tools':  
562         // Show Sandbox DevTools  
563         obj.openDevTools(obj.win_sandbox);  
564         break;  
565     case 'open-dir':  
566     case 'open-folder':  
567     case 'show-dir':  
568     case 'show-folder':  
569         shell.openPath(data);  
570         break;  
571     case 'show-file-in-folder':  
572     case 'show-file-in-dir':  
573         shell.showItemInFolder(data);  
574         break;  
575     case 'focus-win':  
576         e.sender.focus();  
577         break;  
578     case 'fade-in-win':  
579         obj.fadeWindowIn(BrowserWindow.fromWebContents(e.sender), 0.1, 10);  
580         break;  
581     case 'fade-out-win':  
582         obj.fadeWindowIn(BrowserWindow.fromWebContents(e.sender), 0.1, 10);  
583         break;  
584     case 'close-win':  
585         BrowserWindow.fromWebContents(e.sender).close();  
586         e.sender.destroy();  
587         break;  
588     case 'close-app':  
589         if(obj.win_main !== undefined && !obj.win_main.isDestroyed()) {  
590             obj.win_main.destroy();  
591         }  
592         if(obj.win_editor !== undefined && !obj.win_editor.isDestroyed()) {  
593             obj.is_app_quitting = true;  
594             obj.win_editor.send('EditorWindow', 'close-all');  
595         }  
596         if(obj.win_sandbox !== undefined && !obj.win_sandbox.isDestroyed()) {  
597             obj.win_sandbox.destroy();  
598         }  
599         break;  
600     case 'set-fullscreen':  
601         BrowserWindow.fromWebContents(e.sender).setFullScreen(data);  
602         BrowserWindow.fromWebContents(e.sender).maximize();  
603         break;  
604     case 'maximize-win':  
605         BrowserWindow.fromWebContents(e.sender).maximize();  
606         break;  
607     case 'restore-win':  
608         BrowserWindow.fromWebContents(e.sender).restore();  
609         break;  
610     case 'minimize-win':  
611         BrowserWindow.fromWebContents(e.sender).minimize();  
612         break;  
613     case 'show-editor':  
614         // Show editor
```

```

615     obj.showEditor();
616     break;
617   case 'close-editor':
618     // Close editor
619     obj.editor_close_ready = true;
620     obj.win_editor.close();
621     break;
622   case 'capture-page':
623     obj.win_main.webContents.capturePage(undefined, {
624       stayHidden: true,
625       stayAwake: true
626     }).then(function(img) {
627       var size = img.getSize();
628       obj.win_main.webContents.send('streamer', 'captured-page', {buffer
629         : img.toBitmap().buffer, width: size.width, height: size.
630         height});
631     });
632     break;
633   case 'take-screenshot':
634     obj.win_main.webContents.capturePage(undefined, {
635       stayHidden: true,
636       stayAwake: true
637     }).then(function(img) {
638       if(data) {
639         img = img.crop(data);
640       }
641       obj.win_main.webContents.send('gui', 'screenshot', {buffer: img.
642         toPNG()});
643     });
644     break;
645   case 'close-log':
646     obj.app_logger.closeLog();
647     break;
648   case 'code-evaluated':
649     Object.keys(obj.sandbox_sub_wins).forEach(function(wid) {
650       var win = obj.sandbox_sub_wins[wid];
651       if(win.forClose) {
652         setTimeout(function() {
653           win.destroy();
654           delete obj.sandbox_sub_wins[wid];
655         }, 500);
656       }
657     });
658     break;
659   case 'set-win-size':
660     obj.win_main.setSize(data[0], data[1]);
661     break;
662   case 'app-relaunch':
663     app.exit();
664     app.relaunch();
665     break;
666   }
667 });
668 // For MainWindow

```

```

667   ipcMain.on('MainWindow', function(e, action, data) {
668     if (!obj.win_main.isDestroyed()) {
669       switch(action) {
670         default:
671           obj.win_main.send('MainWindow', action, data);
672           break;
673         }
674       }
675     });
676
677   // For EditorWindow
678   ipcMain.on('EditorWindow', function(e, action, data) {
679     switch(action) {
680       case 'open-script':
681         // Open file in editor
682         obj.showEditor();
683         obj.win_editor.send('EditorWindow', 'open-script', data);
684         break;
685       default:
686         // Other actions
687         obj.win_editor.send('EditorWindow', action, data);
688         break;
689       }
690     });
691
692   // For SandboxWindow
693   ipcMain.on('SandboxWindow', function(e, action, data) {
694     if(action == 'stop-loop') {
695       obj.stop_loop_in = data;
696     }
697     switch(action) {
698       default:
699         obj.win_sandbox.send('SandboxWindow', action, data);
700         break;
701       }
702     });
703   }
704
705 /**
706  * Show editor window
707  */
708 showEditor() {
709   this.win_editor.show();
710   this.win_editor.focus();
711   this.fadeWindowIn(this.win_editor, 0.1, 10);
712
713   if(config.DEBUG) {
714     if(!this.win_editor.devtools_win || !this.win_editor.devtools_win.
715       isVisible()) {
716       this.openDevTools(this.win_editor);
717     }
718   }
719 /**
720

```

```

721  * Get window bounds
722  * @param {string} store_key - key used for storage.
723  * @param {object} options - options for window.
724  * @returns {object} window bounds.
725  */
726 getWindowBounds(store_key, options) {
727   var stored_options = this.store.get(store_key);
728   if(stored_options) {
729     var bounds = stored_options.bounds;
730     var area = screen.getDisplayMatching(bounds).workArea;
731     options.maximize = stored_options.maximize;
732     if(
733       bounds.x >= area.x &&
734       bounds.y >= area.y &&
735       bounds.x + bounds.width <= area.x + area.width &&
736       bounds.y + bounds.height <= area.y + area.height
737     ) {
738       options.x = bounds.x;
739       options.y = bounds.y;
740     }
741     if(bounds.width <= area.width || bounds.height <= area.height) {
742       options.width = bounds.width;
743       options.height = bounds.height;
744     }
745   }
746   return options;
747 }
748
749 /**
750  * Save window bounds
751  * @param {BrowserWindow} win - browser window.
752  * @param {string} store_key - key used for storage.
753  */
754 saveWindowBounds(win, store_key) {
755   var obj = this;
756   if(this.debounce_save_win_bounds[store_key]) {
757     clearTimeout(this.debounce_save_win_bounds[store_key]);
758   }
759   this.debounce_save_win_bounds[store_key] = setTimeout(function() {
760     obj.debounce_save_win_bounds[store_key] = undefined;
761     var options = {};
762     options.bounds = win.getNormalBounds();
763     options.maximize = win.isMaximized();
764     obj.store.set(store_key, options);
765   }, this.debounce_save_win_time);
766 }
767
768 /**
769  * Configures session-wide handlers for certificate verification, permission
770  * checks,
771  * device permission handling, and USB protected classes handling. These
772  * handlers ensure
773  * the application's security and user privacy.
774  */

```

```

774     setHandlers() {
775       this.win_main.webContents.session.setCertificateVerifyProc(function(
776         request, callback) {
777           callback(0);
778         });
779 
780       this.win_main.webContents.session.setPermissionCheckHandler(function() {
781         return true;
782       });
783 
784       this.win_main.webContents.session.setDevicePermissionHandler(function() {
785         return true;
786       });
787 
788       this.win_main.webContents.session.setUSBProtectedClassesHandler(function()
789         {
790           return [];
791         });
792     }
793 
794   /**
795    * Opens the Developer Tools for the specified Electron BrowserWindow in a
796    * detached window.
797    * @param {BrowserWindow} win - The Electron 'BrowserWindow' instance for
798    * which to open the Developer Tools.
799   */
800   openDevTools(win) {
801     if(win.webContents.isDevToolsOpened()) {
802       win.webContents.closeDevTools();
803     }
804     win.webContents.openDevTools({mode: 'undocked'});
805     win.webContents.once('devtools-opened', function() {
806       win.devToolsWebContents.focus();
807     });
808   }
809 
810   /**
811    * Gradually increases the opacity of a given window until it is fully
812    * opaque.
813    * @param {BrowserWindow} win - The browser window to fade in.
814    * @param {number} step - The incremental step of opacity change. Default is
815    * 0.1.
816    * @param {number} dt - The time interval in milliseconds between opacity
817    * changes. Default is 10.
818    * @returns {number} The interval identifier for the fade-in operation.
819   */
820   fadeWindowIn(win, step = 0.1, dt = 10) {
821     if(win) {
822       // Get the opacity of the window.
823       var opacity = win.getOpacity();
824       var interval = [];
825 
826       if(opacity != 1) {
827         // Increase the opacity of the window by 'step' every 'dt' ms
828         interval = setInterval(function() {
829           win.setOpacity(opacity + step);
830           if(opacity + step > 1) {
831             clearInterval(interval);
832           }
833         }, dt);
834       }
835     }
836   }
837 
```

```

822         // Stop fading if window's opacity is 1 or greater.
823         if(opacity >= 1) {
824             clearInterval(interval);
825             opacity = 1;
826         }
827         win.setOpacity(opacity);
828         opacity += step;
829     }, dt);
830 }
831
832     // Return the interval. Useful if we want to stop fading at will.
833     return interval;
834 }
835     return false;
836 }
837
838 /**
839 * Gradually decreases the opacity of a given window until it is fully
840 * transparent.
841 * @param {BrowserWindow} win - The browser window to fade out.
842 * @param {number} step - The decremental step of opacity change. Default is
843 * 0.1.
844 * @param {number} dt - The time interval in milliseconds between opacity
845 * changes. Default is 10.
846 * @returns {number} The interval identifier for the fade-out operation.
847 */
848 fadeWindowOut(win, step = 0.1, dt = 10) {
849     if(win) {
850         // Get the opacity of the window.
851         var opacity = win.getOpacity();
852         var interval = [];
853
854         if(opacity != 0) {
855             // Reduce the opacity of the window by 'step' every 'dt' ms
856             interval = setInterval(function() {
857                 // Stop fading if window's opacity is 0 or lesser.
858                 if(opacity <= 0) {
859                     clearInterval(interval);
860                     opacity = 0;
861                 }
862                 win.setOpacity(opacity);
863                 opacity -= step;
864             }, dt);
865         }
866
867         // Return the interval. Useful if we want to stop fading at will.
868         return interval;
869     }
870
871 exports.PRDC_JSLAB_MAIN = PRDC_JSLAB_MAIN;

```

Listing 52 - main.js

```

1  /**
2   * @file Javascript tester module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  if(jsl) {
9    require = jsl._require;
10 }
11
12 const fs = require("fs");
13
14 /**
15  * Class for application testing.
16  */
17 class PRDC_JSLAB_TESTER {
18
19 /**
20  * Initializes the tester with the specified folder containing test modules.
21  * Loads all test modules and prepares them for execution.
22  * @param {string} folder The folder relative to the project directory where
23  * test modules are located.
24  */
25 constructor(folder) {
26   var obj = this;
27   this.path = app_path+/js/+folder;
28   this.modules = [];
29   this.tests = [];
30   this.total_tests = 0;
31
32 // Define function for data display
33 if(jsl) {
34   this.disp = jsl._console.log;
35 } else {
36   this.disp = console.log;
37 }
38
39 // Find all test modules
40 try {
41   var files = fs.readdirSync(this.path);
42   files.forEach(function(file) {
43     if(file.endsWith(".test.js")) {
44       obj.modules.push(obj.path+/'+file);
45     }
46   });
47 } catch(err) {
48   if(err) {
49     this.disp(language.currentString(92) + ': ' + err);
50   }
51
52 // Find all tests defined in test modules
53 obj.modules.forEach(function(module) {
54   var { MODULE_TESTS } = require(module);

```

```

55     obj.total_tests += MODULE_TESTS.testsNumber();
56     obj.tests.push(...MODULE_TESTS.get());
57   });
58 }
59
60 /**
61 * Executes all loaded tests, reports successes and failures, and logs the
62 * results.
63 */
64 runTests() {
65   var obj = this;
66   if(this.total_tests > 0) {
67     this.disp(language.currentString(93)+' '+this.total_tests+' '+language.
68       currentString(94)+'.');
69     var i = 1;
70     var passed = 0;
71     var failed = 0;
72     this.tests.forEach(function(test) {
73       var name = test.name;
74       var result = false;
75       var error = '';
76       try {
77         result = test.run();
78       } catch(e) {
79         error = e;
80       }
81       if(!result) {
82         failed += 1;
83         if(error != '') {
84           obj.disp([' '+i+'/'+obj.total_tests+' '+language.currentString
85             (95)+'',+name+' - '+language.currentString(96)+': '+error]
86             );
87         } else {
88           obj.disp([' '+i+'/'+obj.total_tests+' '+language.currentString
89             (95)+'',+name+' - '+language.currentString(97)+'.']);
90         }
91       }
92     });
93     this.disp(language.currentString(99)+':');
94     this.disp(' '+language.currentString(100)+': '+passed);
95     this.disp(' '+language.currentString(101)+': '+failed);
96   } else {
97     this.disp(language.currentString(102));
98   }
99 }
100 exports.PRDC_JSLAB_TESTER = PRDC_JSLAB_TESTER;
101
102 /**
103 * Represents a collection of tests for a specific module or functionality

```

```

        within the JSLAB application.

104   /**
105  class PRDC_JSLAB_TESTS {
106
107  /**
108  * Creates an instance to manage and store individual tests.
109  */
110 constructor() {
111     this.tests = [];
112 }
113
114 /**
115 * Adds a new test to the collection.
116 * @param {string} name The name of the test.
117 * @param {Function} fun The test function to execute.
118 */
119 add(name, fun) {
120     this.tests.push(new PRDC_JSLAB_TEST(name, fun));
121 }
122
123 /**
124 * Returns all tests added to this collection.
125 * @returns {Array} An array of all tests within this collection.
126 */
127 get() {
128     return this.tests;
129 }
130
131 /**
132 * Returns the number of tests in the collection.
133 * @returns {number} The total number of tests.
134 */
135 testsNumber() {
136     return this.tests.length;
137 }
138 }

139 exports.PRDC_JSLAB_TESTS = PRDC_JSLAB_TESTS;
140
141 /**
142 * Represents an individual test within a test suite.
143 */
144 class PRDC_JSLAB_TEST {
145
146 /**
147 * Initializes a new test with a name and a test function.
148 * @param {string} name The name of the test.
149 * @param {Function} fun The function to execute as the test.
150 */
151 constructor(name, fun) {
152     this.name = name;
153     this.fun = fun;
154 }
155
156 /**
157 */

```

```

158  * Executes the test function and returns the result .
159  * @returns {boolean} The result of the test function execution , true for
160  * pass and false for fail .
161  */
162  run() {
163   return this.fun();
164 }
165
166 exports.PRDC_JSLAB_TEST = PRDC_JSLAB_TEST;

```

Listing 53 - tester.js

6.1 code

```

1 /**
2  * @file Edited CodeMirror javascript-hints.js
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  *
7  * CodeMirror, copyright (c) by Marijn Haverbeke and others
8  * Distributed under an MIT license: https://codemirror.net/LICENSE
9  *
10 */
11
12 (function(mod) {
13   if(typeof exports == "object" && typeof module == "object") // CommonJS
14     mod(require("../lib/codemirror"));
15   else if(typeof define == "function" && define.amd) // AMD
16     define(["../lib/codemirror"], mod);
17   else // Plain browser env
18     mod(CodeMirror);
19 })(function(CodeMirror) {
20   var Pos = CodeMirror.Pos;
21
22   function scriptHint(editor, keywords, getToken, options) {
23
24     // Find the token at the cursor
25     var cur = editor.getCursor(), token = getToken(editor, cur);
26     if(/\b(?:string|comment)\b/.test(token.type)) return;
27     var innerMode = CodeMirror.innerMode(editor.getMode(), token.state);
28     if(innerMode.mode.helperType === "json") return;
29     token.state = innerMode.state;
30
31     // If it's not a 'word-style' token, ignore the token.
32     if(!/^[\w$]*/$.test(token.string)) {
33       token = {start: cur.ch, end: cur.ch, string: "", state: token.state,
34               type: token.string === "." ? "property" : null};
35     } else if(token.end > cur.ch) {
36       token.end = cur.ch;
37       token.string = token.string.slice(0, cur.ch - token.start);
38     }
39

```

```

40   var tprop = token;
41   // If it is a property, find out what it is a property of.
42   while(tprop.type == "property") {
43     tprop = getToken(editor, Pos(cur.line, tprop.start));
44     if(tprop.string != ".") return;
45     tprop = getToken(editor, Pos(cur.line, tprop.start));
46     if(!context) var context = [];
47     context.push(tprop);
48   }
49   return getCompletions(token, context, keywords, options, cur);
50 }
51
52 function javascriptHint(editor, callback, options) {
53   return scriptHint(editor, javascriptKeywords,
54                     function(e, cur) { return e.getTokenAt(cur); },
55                     options);
56 }
57 CodeMirror.registerHelper("hint", "javascript", javascriptHint);
58
59
60 var javascriptKeywords = ("break case catch class const continue debugger
61   default delete do else export extends false finally for function "
62   "if in import instanceof new null return super switch this
63   throw true try typeof var void while with yield").split(
64   " ");
65
66 function getCompletions(token, context, keywords, options, cur) {
67   return new Promise(function(resolve) {
68     ipcRenderer.invoke('get-completions', [token.string, JSON.stringify(
69       context), keywords]).then(function(found) {
70       resolve({list: found,
71                 from: Pos(cur.line, token.start),
72                 to: Pos(cur.line, token.end)}));
73     });
74   });
75 }
76 });

```

Listing 54 - custom-javascript-hint.js

```

1 /**
2 * @file Edited CodeMirror search.js
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 *
7 * CodeMirror, copyright (c) by Marijn Haverbeke and others
8 * Distributed under an MIT license: https://codemirror.net/LICENSE
9 *
10 */
11
12 (function(mod) {
13   if(typeof exports == "object" && typeof module == "object") // CommonJS
14     mod(require("../lib/codemirror"), require("./searchcursor"));
15   else if(typeof define == "function" && define.amd) // AMD
16     define(["../lib/codemirror", "./searchcursor"], mod);

```

```
17     else // Plain browser env
18     mod(CodeMirror);
19 })(function(CodeMirror) {
20   "use strict";
21
22   function searchOverlay(query, caseInsensitive) {
23     if(typeof query == "string")
24       query = new RegExp(query.replace(/[-[\]\/\{\}\(\)\*\+\?\.\\\^\$\|]/g,
25                                         "\$\\$JSLAB_SOURCE_CODE_DATA$"), caseInsensitive ? "gi" : "g");
26     else if(!query.global)
27       query = new RegExp(query.source, query.ignoreCase ? "gi" : "g");
28
29     return {token: function(stream) {
30       query.lastIndex = stream.pos;
31       var match = query.exec(stream.string);
32       if(match && match.index == stream.pos) {
33         stream.pos += match[0].length || 1;
34         return "searching";
35       } else if(match) {
36         stream.pos = match.index;
37       } else {
38         stream.skipToEnd();
39       }
40     }};
41
42   function SearchState() {
43     this.posFrom = this.posTo = this.lastQuery = this.query = null;
44     this.overlay = null;
45   }
46
47   function getSearchState(cm) {
48     return cm.state.search || (cm.state.search = new SearchState());
49   }
50
51   function queryCaseInsensitive(query) {
52     return typeof query == "string" && query == query.toLowerCase();
53   }
54
55   function getSearchCursor(cm, query, pos) {
56     // Heuristic: if the query string is all lowercase, do a case insensitive
57     // search.
58     return cm.getSearchCursor(query, pos, {caseFold: queryCaseInsensitive(
59       query), multiline: true});
60
61   function persistentDialog(cm, text, deflt, onEnter, onKeyDown) {
62     cm.openDialog(text, onEnter, {
63       value: deflt,
64       selectValueOnOpen: true,
65       closeOnEnter: false,
66       onClose: function() { clearSearch(cm); },
67       onKeyDown: onKeyDown
68     });
69   }
70 }
```

```

69
70  function dialog(cm, text, shortText, deflt, f) {
71    if(cm.openDialog) cm.openDialog(text, f, { value: deflt, selectValueOnOpen
72      : true });
73    else f(prompt(shortText, deflt));
74  }
75
76  function confirmDialog(cm, text, shortText, fs) {
77    if(cm.openConfirm) cm.openConfirm(text, fs);
78    else if(confirm(shortText)) fs[0]();
79  }
80
81  function parseString(string) {
82    return string.replace(/\n\([nr\\]\)/g, function(match, ch) {
83      if(ch == "n") return "\n";
84      if(ch == "r") return "\r";
85      if(ch == "t") return "\t";
86      if(ch == "\\") return "\\";
87      return match;
88    });
89  }
90
91  function parseQuery(query) {
92    var isRE = query.match(/^\//(.*)\//([a-z]*$));
93    if(isRE) {
94      try { query = new RegExp(isRE[1], isRE[2].indexOf("i") == -1 ? "" : "i")
95        ;
96      }
97      catch(e) {}
98    } else {
99      query = parseString(query);
100    }
101    if(typeof query == "string" ? query == "" : query.test(""))
102      query = /x^/;
103    return query;
104  }
105
106  function startSearch(cm, state, query) {
107    state.queryText = query;
108    state.query = parseQuery(query);
109    cm.removeOverlay(state.overlay, queryCaseInsensitive(state.query));
110    state.overlay = searchOverlay(state.query, queryCaseInsensitive(state.
111      query));
112    cm.addOverlay(state.overlay);
113    if(cm.showMatchesOnScrollbar) {
114      if(state.annotate) { state.annotate.clear(); state.annotate = null; }
115      state.annotate = cm.showMatchesOnScrollbar(state.query,
116        queryCaseInsensitive(state.query));
117    }
118  }
119
120  function showSearchDialog(cm) {
121    cm.display.wrapper.querySelector(".CodeMirror-search-dialog")
122      .style.display = 'block';
123    cm.display.wrapper.querySelector(".CodeMirror-search-find").focus();
124  }

```

```

120
121  function hideSearchDialog(cm) {
122    cm.display.wrapper.querySelector(".CodeMirror-search-dialog")
123      .style.display = 'none';
124  }
125
126  function doSearch(cm, rev, persistent, immediate) {
127    var state = getSearchState(cm);
128    if(state.query) return findNext(cm, rev);
129    var q = cm.getSelection() || state.lastQuery;
130    if(q instanceof RegExp && q.source == "x^") q = null;
131    if(persistent && cm.openDialog) {
132      var hiding = null;
133      var searchNext = function(query, event) {
134        CodeMirror.e_stop(event);
135        if(!query) return;
136        if(query != state.queryText) {
137          startSearch(cm, state, query);
138          state.posFrom = state.posTo = cm.getCursor();
139        }
140        if(hiding) hiding.style.opacity = 1;
141        findNext(cm, event.shiftKey, function(_, to) {
142          var dialog;
143          if(to.line < 3 && document.querySelector &&
144              (dialog = cm.display.wrapper.querySelector(".CodeMirror-dialog"))
145              ) &&
146              dialog.getBoundingClientRect().bottom - 4 > cm.cursorCoords(to,
147                "window").top)
148            (hiding = dialog).style.opacity = 0.4;
149        });
150        persistentDialog(cm, getQueryDialog(cm), q, searchNext, function(event,
151          query) {
152          var keyName = CodeMirror.keyName(event);
153          var extra = cm.getOption('extraKeys'), cmd = (extra && extra[keyName])
154            || CodeMirror.keyMap[cm.getOption("keyMap")][keyName];
155          if(cmd == "findNext" || cmd == "findPrev" ||
156            cmd == "findPersistentNext" || cmd == "findPersistentPrev") {
157            CodeMirror.e_stop(event);
158            startSearch(cm, getSearchState(cm), query);
159            cm.execCommand(cmd);
160          } else if(cmd == "find" || cmd == "findPersistent") {
161            CodeMirror.e_stop(event);
162            searchNext(query, event);
163          }
164        });
165        if(immediate && q) {
166          startSearch(cm, state, q);
167          findNext(cm, rev);
168        }
169      } else {
170        dialog(cm, getQueryDialog(cm), "Search for:", q, function(query) {
171          if(query && !state.query) cm.operation(function() {
172            startSearch(cm, state, query);
173            state.posFrom = state.posTo = cm.getCursor();

```

```

171     findNext(cm, rev);
172   });
173 }
174 }
175 }

176 function findNext(cm, rev, callback) {cm.operation(function() {
177   var state = getSearchState(cm);
178   var cursor = getSearchCursor(cm, state.query, rev ? state.posFrom : state.
179     posTo);
180   if(!cursor.find(rev)) {
181     cursor = getSearchCursor(cm, state.query, rev ? CodeMirror.Pos(cm.
182       lastLine()) : CodeMirror.Pos(cm.firstLine(), 0));
183   if(!cursor.find(rev)) return;
184   }
185   cm.setSelection(cursor.from(), cursor.to());
186   cm.scrollIntoView({from: cursor.from(), to: cursor.to()}, 20);
187   state.posFrom = cursor.from(); state.posTo = cursor.to();
188   if(callback) callback(cursor.from(), cursor.to());
189 })};

190 function clearSearch(cm) {cm.operation(function() {
191   var state = getSearchState(cm);
192   state.lastQuery = state.query;
193   if(!state.query) return;
194   state.query = state.queryText = null;
195   cm.removeOverlay(state.overlay);
196   if(state.annotate) { state.annotate.clear(); state.annotate = null; }
197 })};

198

199 function getQueryDialog(cm) {
200   return '<span class="CodeMirror-search-label">' + cm.phrase("Search:") +
201     '</span> <input type="text" style="width: 10em" class="CodeMirror-
202     search-field"/> <span style="color: #888" class="CodeMirror-search-
203     hint">' + cm.phrase("(Use /re/ syntax for regexp search)") + '</span>',
204 }
205 function getReplaceQueryDialog(cm) {
206   return '<input type="text" style="width: 10em" class="CodeMirror-search-
207     field"/> <span style="color: #888" class="CodeMirror-search-hint">' +
208     cm.phrase("(Use /re/ syntax for regexp search)") + '</span>';
209 }
210 function getReplacementQueryDialog(cm) {
211   return '<span class="CodeMirror-search-label">' + cm.phrase("With:") + '</
212     span> <input type="text" style="width: 10em" class="CodeMirror-search-
213     field"/>';

214 }
215 function getDoReplaceConfirm(cm) {
216   return '<span class="CodeMirror-search-label">' + cm.phrase("Replace?") +
217     '</span> <button>' + cm.phrase("Yes") + '</button> <button>' + cm.
218     phrase("No") + '</button> <button>' + cm.phrase("All") + '</button> <
219     button>' + cm.phrase("Stop") + '</button> ';
220 }

221 }
```

```

213   function replaceAll(cm, query, text) {
214     cm.operation(function() {
215       for(var cursor = getSearchCursor(cm, query); cursor.findNext();) {
216         if(typeof query != "string") {
217           var match = cm.getRange(cursor.from(), cursor.to()).match(query);
218           cursor.replace(text.replace(/\\$\\d/g, function(_, i) { return match
219             [i]; }));
220         } else cursor.replace(text);
221       });
222     });
223   }
224
224   function replace(cm, all) {
225     if(cm.getOption("readOnly")) return;
226     var query = cm.getSelection() || getSearchState(cm).lastQuery;
227     var dialogText = '<span class="CodeMirror-search-label">' + (all ? cm.
228       phrase("Replace all:") : cm.phrase("Replace:")) + '</span>';
229     dialog(cm, dialogText + getReplaceQueryDialog(cm), dialogText, query,
230       function(query) {
231         if(!query) return;
232         query = parseQuery(query);
233         dialog(cm, getReplacementQueryDialog(cm), cm.phrase("Replace with:"), ""
234           , function(text) {
235             text = parseString(text);
236             if(all) {
237               replaceAll(cm, query, text);
238             } else {
239               clearSearch(cm);
240               var cursor = getSearchCursor(cm, query, cm.setCursor("from"));
241               var advance = function() {
242                 var start = cursor.from(), match;
243                 if(!(match = cursor.findNext())) {
244                   cursor = getSearchCursor(cm, query);
245                   if(!(match = cursor.findNext()) ||
246                     (start && cursor.from().line == start.line && cursor.from().
247                       ch == start.ch)) return;
248                 }
249                 cm.setSelection(cursor.from(), cursor.to());
250                 cm.scrollIntoView({from: cursor.from(), to: cursor.to()});
251                 confirmDialog(cm, getDoReplaceConfirm(cm), cm.phrase("Replace?"),
252                   [function() {doReplace(match);}, advance,
253                     function() {replaceAll(cm, query, text)}]);
254               };
255               var doReplace = function(match) {
256                 cursor.replace(typeof query == "string" ? text :
257                   text.replace(/\\$\\d/g, function(_, i) { return
258                     match[i]; }));
259                 advance();
260               };
261             });
262           });
263         });
264       });
265     });
266   }

```

```

262 CodeMirror.commands.showSearchDialog = function(cm) { clearSearch(cm);  

263   showSearchDialog(cm); };  

264 CodeMirror.commands.hideSearchDialog = function(cm) { clearSearch(cm);  

265   hideSearchDialog(cm); };  

266 CodeMirror.commands.find = function(cm) { clearSearch(cm); doSearch(cm); };  

267 CodeMirror.commands.findPersistent = function(cm) { clearSearch(cm);  

268   doSearch(cm, false, true); };  

269 CodeMirror.commands.findPersistentNext = function(cm) { doSearch(cm, false,  

270   true, true); };  

271 CodeMirror.commands.findPersistentPrev = function(cm) { doSearch(cm, true,  

272   true, true); };  

273 CodeMirror.commands.findNext = doSearch;  

274 CodeMirror.commands.findPrev = function(cm) { doSearch(cm, true); };  

275 CodeMirror.commands.clearSearch = clearSearch;  

276 CodeMirror.commands.replace = replace;  

277 CodeMirror.commands.replaceAll = function(cm) { replace(cm, true); };  

278 });

```

Listing 55 - custom-search.js

```

1 /**
2  * @file Search dialog based on CodeMirror search.js
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  *
7  * CodeMirror, copyright (c) by Marijn Haverbeke and others
8  * Distributed under an MIT license: https://codemirror.net/LICENSE
9  */
10
11 (function(mod) {
12   if(typeof exports == "object" && typeof module == "object") // CommonJS
13     mod(require("../lib/codemirror"), require("./searchcursor"));
14   else if(typeof define == "function" && define.amd) // AMD
15     define(["../../../lib/codemirror", "./searchcursor"], mod);
16   else // Plain browser env
17     mod(CodeMirror);
18 }) (function(CodeMirror) {
19   "use strict";
20
21   CodeMirror.defineOption('searchDialog', false, function(cm, val) {
22     if(val) {
23       var div = document.createElement('div');
24       div.className = 'CodeMirror-search-dialog';
25       div.innerHTML = '<input class="CodeMirror-search-find" value=""  

26         placeholder="Find" autocomplete="off" spellcheck="false" title="Find"><i class="CodeMirror-search-find-prev-btn" title="Previous match"></i><i class="CodeMirror-search-find-next-btn" title="Next match"></i><br class="clear"/><input class="CodeMirror-search-replace" value="" placeholder="Replace" autocomplete="off" spellcheck="false" title="Replace"><i class="CodeMirror-search-replace-btn" title="Replace"></i><i class="CodeMirror-search-replace-all-btn" title="Replace All"></i><br class="clear"/><i class="CodeMirror-search-case-btn" title="Match Case"></i><i class="CodeMirror-search-regex-btn" title="Use Regular Expression"></i><div class="CodeMirror-search-

```

```

bottom-right"><span class="CodeMirror-search-current-match">0</span>
  of <span class="CodeMirror-search-total-matches">0</span><i class="CodeMirror-search-close-btn" title="Close Search Dialog"></i></div><br class="clear"/>';
27  div.setAttribute('tabindex', 0);

28
29  var find_input = div.querySelector('.CodeMirror-search-find');
30  var replace_input = div.querySelector('.CodeMirror-search-replace');
31  var find_prev_btn = div.querySelector('.CodeMirror-search-find-prev-btn');
32    );
33  var find_next_btn = div.querySelector('.CodeMirror-search-find-next-btn');
34    );
35  var replace_btn = div.querySelector('.CodeMirror-search-replace-btn');
36  var replace_all_btn = div.querySelector('.CodeMirror-search-replace-all-btn');
37
38  var match_case_btn = div.querySelector('.CodeMirror-search-case-btn');
39  var regex_btn = div.querySelector('.CodeMirror-search-regex-btn');
40  var close = div.querySelector('.CodeMirror-search-close-btn');

41
42  div.addEventListener('keydown', function(e) {
43    if(e.key == 'Escape') {
44      // ESC
45      if(window.jQuery) {
46        $(div).slideUp(200);
47      } else {
48        div.style.display = 'none';
49      }
50      cm.focus();
51    }
52  });

53
54  replace_input.addEventListener('keydown', function(e) {
55    if(e.key == 'Enter') {
56      // Replace
57      replace(cm, replace_input.value, false);
58    }
59  });

60
61  find_input.addEventListener('keyup', function() {
62    // Find match
63    clearSearch(cm);
64    var state = getSearchState(cm);
65    state.query = this.value;
66    doSearch(cm);
67  });

68
69  find_prev_btn.addEventListener('click', function() {
70    // Show previous match
71    doSearch(cm, true);
72  });

73
74  find_next_btn.addEventListener('click', function() {

```

```

75      // Show next match
76      doSearch(cm, false);
77  });
78
79  replace_btn.addEventListener('click', function() {
80      // Replace
81      replace(cm, replace_input.value, false)
82  });
83
84  replace_all_btn.addEventListener('click', function() {
85      // Replace all
86      replace(cm, replace_input.value, true)
87  });
88
89  match_case_btn.addEventListener('click', function() {
90      if(this.classList.contains('active')) {
91          cm.search_match_case = false;
92          this.classList.remove('active');
93      } else {
94          cm.search_match_case = true;
95          this.classList.add('active');
96      }
97  });
98
99  regex_btn.addEventListener('click', function() {
100     if(this.classList.contains('active')) {
101         cm.search_regex = false;
102         this.classList.remove('active');
103     } else {
104         cm.search_regex = true;
105         this.classList.add('active');
106     }
107  });
108
109 close.addEventListener('click', function() {
110     if(window.jQuery) {
111         $(div).slideUp(200);
112     } else {
113         div.style.display = 'none';
114     }
115     cm.focus();
116  });
117
118     cm.display.wrapper.appendChild(div);
119 }
120 });
121
122 function searchOverlay(query, caseInsensitive) {
123     if(typeof query === "string")
124         query = new RegExp(query.replace(/[\-\[\]\]/g, '$'), caseInsensitive ? "gi" : "g");
125     else if(!query.global)
126         query = new RegExp(query.source, query.ignoreCase ? "gi" : "g");
127
128     return {token: function(stream) {

```

```

129     query.lastIndex = stream.pos;
130     var match = query.exec(stream.string);
131     if(match && match.index == stream.pos) {
132         stream.pos += match[0].length || 1;
133         return "searching";
134     } else if(match) {
135         stream.pos = match.index;
136     } else {
137         stream.skipToEnd();
138     }
139     return false;
140 };
141 }
142
143 function SearchState() {
144     this.posFrom = this.posTo = this.lastQuery = this.query = null;
145     this.overlay = null;
146 }
147
148 function getSearchState(cm) {
149     return cm.state.search || (cm.state.search = new SearchState());
150 }
151
152 function queryCaseInsensitive(cm, query) {
153     return !cm.search_match_case && typeof query == "string";
154 }
155
156 function getSearchCursor(cm, query, pos) {
157     return cm.getSearchCursor(query, pos, {caseFold: queryCaseInsensitive(cm,
158         query), multiline: true});
159 }
160
161 function parseString(string) {
162     return string.replace(/\[\nrt\]/g, function(match, ch) {
163         if(ch == "n") return "\n";
164         if(ch == "r") return "\r";
165         if(ch == "t") return "\t";
166         if(ch == "\\") return "\\";
167         return match;
168     });
169 }
170
171 function parseQuery(cm, query) {
172     if(cm.search_regex) {
173         var isRE = query.match(/^ \/(.*) \/( [a-z]* )$/);
174         if(isRE) {
175             try { query = new RegExp(isRE[1], isRE[2].indexOf("i") == -1 ? "" : "i");
176             } catch(e) {}
177         } else {
178             query = parseString(query);
179         }
180         if(typeof query == "string" ? query == "" : query.test(""))
181             query = /x^/;
```

```
182     return query;
183 }
184
185 function startSearch(cm, state, query) {
186     state.queryText = query;
187     state.query = parseQuery(cm, query);
188     cm.removeOverlay(state.overlay, queryCaseInsensitive(cm, state.query));
189     state.overlay = searchOverlay(state.query, queryCaseInsensitive(cm, state.
190         query));
191     cm.addOverlay(state.overlay);
192     if(state.annotate) { state.annotate.clear(); state.annotate = null; }
193     state.annotate = cm.showMatchesOnScrollbar(state.query,
194         queryCaseInsensitive(cm, state.query));
195 }
196
197 function showSearchDialog(cm) {
198     var div = cm.display.wrapper.querySelector('.CodeMirror-search-dialog');
199     if(window.jQuery) {
200         $(div).slideDown(200);
201     } else {
202         div.style.display = 'block';
203     }
204     var query = cm.getSelection();
205     var find_input = cm.display.wrapper.querySelector('.CodeMirror-search-find');
206     if(query) {
207         find_input.value = query;
208         clearSearch(cm);
209         var state = getSearchState(cm);
210         state.query = query;
211         doSearch(cm);
212     }
213     find_input.focus();
214
215 function hideSearchDialog(cm) {
216     var div = cm.display.wrapper.querySelector('.CodeMirror-search-dialog');
217     if(window.jQuery) {
218         $(div).slideUp(200);
219     } else {
220         div.style.display = 'none';
221     }
222
223 function doSearch(cm, rev) {
224     var state = getSearchState(cm);
225     var q = state.query;
226     if(q != state.queryText) {
227         startSearch(cm, state, q);
228         state.posFrom = state.posTo = cm.getCursor();
229     }
230     if(q) {
231         findNext(cm, rev);
232     }
233     if(state.annotate) {
```

```

234     var current_match_index = state.annotate.matches.findIndex(function(e) {
235       return e.from.line === state.posFrom.line &&
236         e.from.ch === state.posFrom.ch && e.to.line === state.posTo.line &&
237         e.to.ch === state.posTo.ch;
238     })+1;
239     var total_matchs = cm.display.wrapper.querySelector('.CodeMirror-search-
240       total-matchs');
241     total_matchs.innerText = state.annotate.matches.length;
242     var current_match = cm.display.wrapper.querySelector('.CodeMirror-search-
243       -current-match');
244     current_match.innerText = current_match_index;
245   }
246
247   function findNext(cm, rev, callback) {cm.operation(function() {
248     var state = getSearchState(cm);
249     var cursor = getSearchCursor(cm, state.query, rev ? state.posFrom : state.
250       posTo);
251     if(!cursor.find(rev)) {
252       cursor = getSearchCursor(cm, state.query, rev ? CodeMirror.Pos(cm.
253         lastLine()) : CodeMirror.Pos(cm.firstLine(), 0));
254     if(!cursor.find(rev)) return;
255     }
256     cm.setSelection(cursor.from(), cursor.to());
257     cm.scrollIntoView({from: cursor.from(), to: cursor.to()}, 20);
258     state.posFrom = cursor.from(); state.posTo = cursor.to();
259     if(callback) callback(cursor.from(), cursor.to());
260   });
261
262   function clearSearch(cm) {cm.operation(function() {
263     var state = getSearchState(cm);
264     state.lastQuery = state.query;
265     if(!state.query) return;
266     state.query = state.queryText = null;
267     cm.removeOverlay(state.overlay);
268     if(state.annotate) { state.annotate.clear(); state.annotate = null; }
269   });
270
271   function replaceAll(cm, query, text) {
272     clearSearch(cm);
273     cm.operation(function() {
274       for(var cursor = getSearchCursor(cm, query); cursor.findNext();) {
275         if(typeof query != "string") {
276           var match = cm.getRange(cursor.from(), cursor.to()).match(query);
277           cursor.replace(text.replace(/\$(\d)/g, function(_, i) { return match
278             [i]; }));
279         } else cursor.replace(text);
280       }
281     });
282
283   function replace(cm, text, all) {
284     if(cm.getOption("readOnly")) return;
285     var query = getSearchState(cm).query;
286     if(query) {

```

```

284     text = parseString(text);
285     if(all) {
286         replaceAll(cm, query, text);
287     } else {
288         var cursor = getSearchCursor(cm, query, cm.getCursor());
289         var start = cursor.from(), match;
290         if(!(match = cursor.findNext()))) {
291             cursor = getSearchCursor(cm, query);
292             if(!(match = cursor.findNext())) ||
293                 (start && cursor.from().line == start.line && cursor.from().ch ==
294                  start.ch)) return;
295         }
296         cm.setSelection(cursor.from(), cursor.to());
297         cursor.replace(typeof query == "string" ? text :
298             text.replace(/\$(\d)/, function(w, i) {return match[i];}));
299     }
300 }
301
302 CodeMirror.commands.showSearchDialog = function(cm) { clearSearch(cm);
303     showSearchDialog(cm);};
304 CodeMirror.commands.hideSearchDialog = function(cm) { clearSearch(cm);
305     hideSearchDialog(cm);};
306 CodeMirror.commands.find = function(cm) { clearSearch(cm); doSearch(cm);};
307 CodeMirror.commands.findNext = function(cm) {doSearch(cm, false);};
308 CodeMirror.commands.findPrev = function(cm) {doSearch(cm, true);};
309 CodeMirror.commands.clearSearch = clearSearch;
310 CodeMirror.commands.replace = replace;
311 CodeMirror.commands.replaceAll = function(cm) {replace(cm, true);};
312 });

```

Listing 56 - dialog-search.js

6.2 dev

```

1  /**
2   * @file Native modules build
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   * @version 0.0.1
7   */
8
9 // Import the filesystem module
10 const fs = require('fs');
11
12 global.process_arguments = process.argv;
13 require('../init-config.js');
14
15 console.log('[build-configure.js] Started');
16 var t = performance.now();
17
18 // Change package.json
19 console.log('Changing package.json...');


```

```

20 var data = JSON.parse(fs.readFileSync('package.json'));
21 var filename = data.name + '_' + data.version;
22 if(process.argv.length > 3) {
23   var action = process.argv[3];
24   if(action === 'dist-portable') {
25     filename += '_portable';
26   }
27 }
28
29 data.build.artifactName = filename + '.' + ext;
30
31 if(config.SIGN_BUILD) {
32   data.build.win.sign = "js/dev/build-sign.js";
33 } else {
34   delete data.build.win.sign;
35 }
36
37 fs.writeFileSync('package.json', JSON.stringify(data, null, 2));
38
39 // Creating binding.gyp
40 console.log('Creating binding.gyp...');
41 var binding_data = {targets: []};
42
43 // - Native module
44 if(fs.existsSync('cpp/binding.gyp')) {
45   var data = JSON.parse(fs.readFileSync('cpp/binding.gyp'));
46   if(!Array.isArray(data)) {
47     data = [data];
48   }
49   for(var i = 0; i < data.length; i++) {
50     binding_data.targets[i] = data[i];
51     binding_data.targets[i].defines = ["NAPI_DISABLE_CPP_EXCEPTIONS"];
52   }
53 }
54
55 // - Libs binding.gyp
56 config.COMPILE_LIBS.forEach(function(lib) {
57   var data = JSON.parse(fs.readFileSync('lib/' + lib + '/binding.gyp'));
58   binding_data.targets.push(data);
59 });
60
61 // - Save binding.gyp
62 fs.writeFileSync('binding.gyp', JSON.stringify(binding_data, null, 2));
63
64 // Create bin folder
65 if(!fs.existsSync('bin/')) {
66   fs.mkdirSync('bin/');
67 }
68
69 console.log('[build-configure.js] ' + ((performance.now() - t) / 1000).toFixed(3)
70   + ' s');
```

Listing 57 - build-configure.js

```

1 /**
2  * @file Build sign
```

```

3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7  */
8
9 require('../init-config.js');
10
11 console.log('[build-sign.js] Started');
12 var t = performance.now();
13
14 if (!process.env.COMPANY_NAME) {
15   console.error('Environment variable COMPANY_NAME must be defined!');
16 }
17 if (!process.env.TIMESTAMP_SERVER) {
18   console.error('Environment variable TIMESTAMP_SERVER must be defined!');
19 }
20 if (!process.env.SIGN_TOOL_PATH) {
21   console.error('Environment variable SIGN_TOOL_PATH must be defined!');
22 }
23
24 const { execSync } = require('child_process');
25
26 /**
27  * Signs a specified file using the digital signature tool and parameters
28  * defined in the application's configuration. The
29  * function reads the configuration to obtain the path to the signing tool,
30  * the company name for the signature, and other
31  * necessary parameters. It then constructs the command to execute the signing
32  * process and runs it using 'execSync'.
33  *
34  * @param {Object} data - An object containing parameters for the signing
35  * process.
36  * @param {string} data.path - The path to the file that needs to be signed.
37  * @param {string} data.hash - The hash algorithm to use for signing (e.g., 'sha256').
38  */
39 exports.default = function(data) {
40   console.info('Signing '+data.path+' with '+data.hash+' to '+config.
41   COMPANY_NAME);
42
43   const sha256 = data.hash === 'sha256';
44   const appendCert = sha256 ? '/as' : null;
45   const timestamp = sha256 ? '/tr' : '/t';
46   const appendTd = sha256 ? '/td sha256' : null;
47
48   let args = [
49     ''+config.SIGN_TOOL_PATH+'',
50     'sign',
51     '/debug',
52     '/n',
53     ''+config.COMPANY_NAME+'',
54     '/a',
55     appendCert,

```

```

52     '/fd',
53     data.hash,
54     timestamp,
55     config.TIMESTAMP_SERVER,
56     appendTd,
57     '/v',
58     '"${data.path}"'
59   ];
60
61   try {
62     const { stdout } = execSync(args.join(' '));
63     console.log(stdout);
64   } catch(err) {
65     throw err;
66   }
67
68   console.log(`[ build-sign.js ] ${((performance.now() - t) / 1000).toFixed(3)} s`);
69 }

```

Listing 58 - build-sign.js

```

1  /**
2  * @file Clear app data
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7  */
8
9 // Import modules
10 const fs = require('fs');
11 const os = require('os');
12 const rimraf = require('rimraf');
13 const readline = require('readline').createInterface({
14   input: process.stdin,
15   output: process.stdout
16 });
17
18 var data = JSON.parse(fs.readFileSync('package.json'));
19
20 console.log(`[ clear-app-data.js ] Started`);
21 var t = performance.now();
22
23 // Variables
24 var confirm = process.argv.includes('--confirm');
25
26 if(confirm) {
27   readline.question("Clear app data? yes/[no]: ", function(answer) {
28     if(answer === 'yes') {
29       main();
30     } else {
31       console.log('Action aborted.');
32       console.log(`[ clear-app-data.js ] Execution done in ${((performance.now() - t) / 1000).toFixed(3)} s`);
33       process.exit();
34     }
35   });
36 }

```

```

34      }
35      readline.close();
36  });
37 } else {
38   main();
39 }
40
41 function main() {
42   var path = os.homedir()+'\\AppData\\Roaming\\'+data.name;
43   console.log('Clearing app data from '+path);
44   rimraf.sync(path);
45
46   console.log('[ clear-app-data.js ] Execution done in ' + ((performance.now()-t
47     )/1000).toFixed(3) + ' s');
48   process.exit();
49 }
```

Listing 59 - clear-app-data.js

```

1 /**
2  * @file Download libs
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7 */
8
9 // Import modules
10 const fs = require('fs');
11 const rimraf = require('rimraf');
12 const readline = require('readline').createInterface({
13   input: process.stdin,
14   output: process.stdout
15 });
16 const path = require('path');
17 const { extractFull } = require('node-7z');
18 const bin_path = require('7zip-bin').path7za;
19
20 require('../init-config.js');
21
22 console.log('[ dev-download-libs.js ] Started');
23 var t = performance.now();
24
25 // Variables
26 var force = process.argv.includes('--force');
27 var confirm = process.argv.includes('--confirm');
28
29 if(confirm) {
30   readline.question("Download libs? yes/[no]: ", function(answer) {
31     if(answer === 'yes') {
32       main();
33     } else {
34       console.log('Libs download aborted.');
35       console.log('[ dev-download-libs.js ] Execution done in ' + ((performance.
36         now() - t) / 1000).toFixed(3) + ' s');
37       process.exit();
38     }
39   });
40 }
```

```

37     }
38     readline.close();
39   });
40 } else {
41   main();
42 }
43
44 async function main() {
45   if(force) {
46     console.log('Deleting all libs... ');
47     rimraf.sync('./lib');
48   }
49
50   for(const lib of config.USED_LIBS) {
51     const libDir = path.join('./lib', lib);
52     const serverLibDir = path.join(config.SERVER_LIBS_PATH, lib);
53     const serverLib7z = path.join(config.SERVER_LIBS_PATH, `${lib}.7z`);
54     const lib7z = path.join('./lib', `${lib}.7z`);

55     // Check if lib folder exists locally
56     if(!fs.existsSync(libDir)) {
57       if(fs.existsSync(serverLibDir)) {
58         process.stdout.write(`Downloading ${lib}... `);
59         await fs.promises.cp(serverLibDir, libDir, { recursive: true });
60         process.stdout.clearLine(0);
61         process.stdout.cursorTo(0);
62         console.log(`Downloading of ${lib} complete.`);
63       } else if(fs.existsSync(serverLib7z)) {
64         await copyWithProgress(lib, serverLib7z, lib7z);
65         await extractWithProgress(lib, lib7z, './lib');
66       } else {
67         console.log(`[ERROR] Neither folder nor .7z found for ${lib}. `);
68       }
69     } else {
70       console.log(`Lib ${lib} already downloaded.`);
71     }
72   }
73 }

74 console.log(`\nAll libraries downloaded.`);
75 console.log(`[download-libs.js] Execution done in ${((performance.now() - t) / 1000).toFixed(3)} s`);
76 process.exit();
77 }

78 // Function to copy files with progress
79 async function copyWithProgress(lib, src, dest) {
80   return new Promise((resolve, reject) => {
81     const total_size = fs.statSync(src).size;
82     let copied = 0;

83     const read_stream = fs.createReadStream(src);
84     const write_stream = fs.createWriteStream(dest);

85     read_stream.on('data', (chunk) => {
86       copied += chunk.length;
87     })
88   })
89 }
90 
```

```

91   const progress = ((copied / total_size) * 100).toFixed(2);
92
93   process.stdout.clearLine(0);
94   process.stdout.cursorTo(0);
95   process.stdout.write(`Downloading ${lib}: ${progress}% (${copied}/${total_size})`);
96 });
97
98   read_stream.pipe(write_stream);
99
100  write_stream.on('finish', () => {
101    process.stdout.clearLine(0);
102    process.stdout.cursorTo(0);
103    console.log(`Downloading of ${lib} complete.`);
104    resolve();
105  });
106
107  read_stream.on('error', reject);
108  write_stream.on('error', reject);
109});
110}
111
112 // Function to extract .7z with progress
113 async function extractWithProgress(lib, archive, dest) {
114   return new Promise((resolve, reject) => {
115     const extractor = extractFull(archive, dest, {
116       $bin: bin_path,
117       $progress: true
118     });
119
120     extractor.on('progress', (progress) => {
121       const percent = progress.percent.toFixed(0);
122       process.stdout.clearLine(0);
123       process.stdout.cursorTo(0);
124       process.stdout.write(`Extracting ${lib}: ${percent}%`);
125     });
126
127     extractor.on('end', () => {
128       process.stdout.clearLine(0);
129       process.stdout.cursorTo(0);
130       console.log(`Extraction of ${lib} complete.`);
131       fs.unlinkSync(archive);
132       resolve();
133     });
134
135     extractor.on('error', (err) => {
136       console.error(`Extraction error for ${lib}:`, err);
137       fs.unlinkSync(archive);
138       reject(err);
139     });
140   });
141 }

```

Listing 60 - download-libs.js

```
2  * @file Generate sandbox documentation for JSLAB
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7  */
8
9 const jsdoc = require('jsdoc-api');
10 const fs = require('fs');
11 const cp = require("child_process");
12 const rimraf = require('rimraf');
13
14 require('../init-config.js');
15
16 console.log('[make-doc.js] Started');
17 var t = performance.now();
18
19 var package_data = JSON.parse(fs.readFileSync('package.json'));
20 var app_version = package_data.version;
21 var year = new Date().getFullYear();
22
23 var jslab_doc = {
24   'global': {},
25   'lib': {}
26 };
27
28 async function processDoc(module) {
29   var docs_out;
30   if(config.OUTPUT_COMPLETE_JSDOC) {
31     docs_out = [];
32   } else {
33     docs_out = {};
34   }
35   var docs = await jsdoc.explain({ files: 'js/sandbox/' + module.file + '.js' });
36   docs.forEach(function(doc) {
37     if(config.OUTPUT_COMPLETE_JSDOC) {
38       docs_out.push(doc);
39     } else if((doc.kind === 'function' || doc.kind === 'member') &&
40       doc.memberof === module.className &&
41       !doc.name.startsWith('_') &&
42       !doc.name.startsWith('#') &&
43       doc.name !== 'jsl' &&
44       doc.description) {
45       var doc_output = {};
46       doc_output.name = doc.name;
47       doc_output.kind = doc.kind;
48       doc_output.description = doc.description;
49       if(doc.params) {
50         doc_output.params = doc.params;
51       }
52       if(doc.returns) {
53         doc_output.returns = doc.returns;
54       }
55       if(doc.async) {
56         doc_output.async = doc.async;
```

```

57     }
58     if(doc.meta) {
59         doc_output.source_filename = doc.meta.filename;
60         doc_output.source_lineno = doc.meta.lineno;
61         doc_output.source_range = doc.meta.range;
62     }
63     docs_out[doc.name] = doc_output;
64 }
65 });
66 return docs_out;
67 }
68
69 (async function main() {
70     config.SUBMODULES['builtin'].push(...config.DOC_SUBMODULES_ADDITIONAL);
71     for(var module of config.SUBMODULES['builtin']) {
72         console.log(' Generating documentation for: global/' + module.name);
73         jslab_doc['global'][module.name] = await processDoc(module);
74     }
75     for(var lib of config.SUBMODULES['lib']) {
76         console.log(' Generating documentation for: lib/' + lib.name);
77         jslab_doc['lib'][lib.name] = await processDoc(lib);
78     }
79     var jslab_doc_flat = Object.values(jslab_doc).reduce((acc, innerObj) => {
80         return { ...acc, ...innerObj };
81     }, {});
82     if(config.OUTPUT_COMPLETE_JSDOC) {
83         return;
84     }
85
86     // Make JSON documentation
87     console.log(' Making documentation.json');
88     fs.writeFileSync('docs/documentation.json', JSON.stringify(jslab_doc, null,
89                     2));
90
91     // Make HTML documentation
92     console.log(' Making documentation.html');
93     var html_template = fs.readFileSync('dev/documentation_template.html', 'utf8
94                     ');
95     html_template = html_template.replaceAll('$JSLAB_CODE_DATA
96
97     % -----
98     % End of document
99     % -----
100    \end{document}, JSON.stringify(jslab_doc_flat, null, 2));
101    html_template = html_template.replaceAll('$JSLAB_APP_VERSION
102
103    % -----
104    % End of document
105    % -----
106    \end{document}, "'v'+app_version+''");
107    html_template = html_template.replaceAll('$JSLAB_PUBLISH_YEAR
108
109    % -----
110    % End of document
111    % -----
```

```

110 \end{document}, ''+year+'');
111 fs.writeFileSync('docs/documentation.html', html_template);
112
113 // Make tex
114 console.log(' Making documentation.tex');
115 var latex_template = fs.readFileSync('dev/documentation_template.tex', 'utf8');
116 latex_template = latex_template.replaceAll('$JSLAB_APP_VERSION'
117
118 % -----
119 % End of document
120 % -----
121 \end{document}, 'v'+app_version+'');
122 latex_template = latex_template.replaceAll('$JSLAB_PUBLISH_YEAR'
123
124 % -----
125 % End of document
126 % -----
127 \end{document}, year);
128
129 function getLatexByCodeCategory(category) {
130   // Helper function to escape LaTeX special characters
131   function escapeLatex(string) {
132     if(typeof string !== 'string') {
133       return string;
134     }
135     return string
136       .replace(/\$/g, '\\textbackslash{}')
137       .replace(/\&/g, '\\&')
138       .replace(/\%/g, '\\%')
139       .replace(/\$/g, '\\'
140
141 % -----
142 % End of document
143 % -----
144 \end{document})
145       .replace(/\#/g, '\\#')
146       .replace(/\_ /g, '\\_')
147       .replace(/\{/g, '\\{')
148       .replace(/\}/g, '\\}')
149       .replace(/\~/g, '\\textasciitilde{}')
150       .replace(/\^/g, '\\textasciicircum{}')
151       .replace(/\`/g, '\\textasciigrave{}');
152   }
153
154   let latex = '';
155
156   // Iterate over each item in the specified category
157   latex += '\\subsection{$\\{escapeLatex(category)}$}\\n';
158
159   for(const item_name in jslab_doc_flat[category]) {
160     const item = jslab_doc_flat[category][item_name];
161
162     if(item.kind === 'function') {
163       // Generate the function signature

```

```

164   const params = item.params
165   ? item.params
166     .map(param => {
167       if(param.type === 'Object' && param.properties) {
168         // Generate nested object structure for 'opts'
169         const nestedSignature = param.properties
170           .map(prop =>
171             prop.optional
172               ? `[$ ${escapeLatex(prop.name)} ]`
173               : `${escapeLatex(prop.name)}`
174           )
175           .join(', ');
176
177         // Format the 'opts' parameter
178         return param.optional
179           ? `[$ ${escapeLatex(param.name)} ]\\{ ${nestedSignature} } ]`
180           : `${escapeLatex(param.name)} ]\\{ ${nestedSignature} }`;
181     } else {
182       // Handle simple parameters with optionality and default
183       // values
184       return param.optional
185         ? `[$ ${escapeLatex(param.name)} ]${param.default ? `=${
186           escapeLatex(param.default)}` : ''}`
187         : `${escapeLatex(param.name)} ]${param.default ? `=${
188           escapeLatex(param.default)}` : ''}`;
189     }
190   }
191
192   // Add a title for the function
193   latex += '\\vspace{5mm}\\n\\noindent \\codeBlock{\\texttt{${escapeLatex(
194     item.name)}(${params})}}\\color{jsl-gray}\\vspace{2mm}\\hrule\\
195   \\vspace{4mm}}\\n\\n';
196
197   // Add metadata if available
198   if(item.since) {
199     latex += '\\textbf{Added in:} \\${escapeLatex(item.since)}\\\\\\n';
200   }
201
202   // Add parameters section
203   if(item.params && item.params.length > 0) {
204     latex += '\\n\\noindent \\textbf{Parameters:}\\n\\begin{itemize}\\n';
205     item.params.forEach(param => {
206       const type = param.type && param.type.names && param.type.names[0]
207         ? param.type.names[0] : 'Unknown';
208       latex += '\\\\item \\texttt{${escapeLatex(param.name)}} \\texttt{<
209         ${escapeLatex(type)}>}: ${escapeLatex(param.description || '')}\\n';
210     });
211     latex += '\\\\end{itemize}\\n';
212   }
213
214   // Add returns section
215   if(item.returns && item.returns.length > 0) {

```

```

211     const returnType = item.returns[0].type && item.returns[0].type.
212     names && item.returns[0].type.names[0] ? item.returns[0].type.
213     names[0] : 'Unknown';
214     latex += '\n\\noindent \\textbf{Returns:} \\texttt{<${escapeLatex(
215       returnType)}>}: ${escapeLatex(item.returns[0].description || '')}
216   }'
217
218   // Add description
219   if(item.description) {
220     latex += '\n\\noindent ${escapeLatex(item.description)}\n\n';
221   }
222
223   // Add examples if available
224   if(item.examples && item.examples.length > 0) {
225     latex += '\\begin{verbatim}\n${escapeLatex(item.examples[0])}\n\\end
226   {verbatim}\n\n';
227 }
228
229   // Add a title for the member
230   if(item.kind === 'member') {
231     latex += '\\vspace{5mm}\\noindent \\code{\\texttt{$
232     ${escapeLatex(item.name)}}}\\color{jsl-gray}\\vspace{2mm}\\hrule\\vspace{4mm}\n
233   ';
234
235   // Add type information
236   const type = item.type && item.type.names && item.type.names[0] ? item
237     .type.names[0] : 'Unknown';
238   latex += '\n\\noindent \\textbf{Type:} \\texttt{<${escapeLatex(type)
239     }>}\n';
240
241   // Add description
242   if(item.description) {
243     latex += '\n\\noindent ${escapeLatex(item.description)}\n\n';
244   }
245
246   return latex;
247 }
248
249 var latex_text = '';
250 for(const category in jslab_doc_flat) {
251   latex_text += getLatexByCodeCategory(category);
252 }
253 latex_template = latex_template.replaceAll('$JSLAB_CODE_DATA
254 % -----
```

```

257 % End of document
258 % -----
259 \end{document}, latex_text);
260 fs.writeFileSync('docs/documentation.tex', latex_template);
261
262 // Make pdf
263 console.log(' Making documentation.pdf');
264
265 var dir = __dirname + '/../../dev/latex/';
266 var file = dir + 'documentation.tex';
267 fs.mkdirSync(dir, { recursive: true });
268 fs.writeFileSync('docs/documentation.tex', file);
269 fs.cpSync('docs/resources/', dir + 'resources/', { recursive: true });
270
271 for(var i = 0; i < config.DOC_LATEX_RERUNS_NUMBER; i++) {
272   console.log(' Run ' + (i+1) + '/' + config.DOC_LATEX_RERUNS_NUMBER);
273   try {
274     cp.execSync('cd ' + dir + '& pdflatex documentation.tex --interaction=nonstopmode', { cwd: dir });
275   } catch(e) {
276     if (!fs.existsSync(dir + 'documentation.pdf') || (e.output &&
277       e.output.toString().trim().split('\n').pop().toLowerCase().includes(
278         'fatal'))) {
279       console.log(' LaTeX compile failed! Add path to pdflatex to
280                 environment variables and try again with MiKTeX 24.1 or later.
281                 Output: ');
282       if (e.output) {
283         console.log(e.output.toString());
284       } else {
285         console.log(e);
286       }
287       break;
288     }
289   }
290   if (fs.existsSync(dir + 'documentation.pdf')) {
291     fs.writeFileSync(dir + 'documentation.pdf', 'docs/documentation.pdf');
292   }
293   rimraf.sync(dir);
294   console.log('[make-doc.js] ' + ((performance.now()-t)/1000).toFixed(3) + ' s
295   ');
296 })();

```

Listing 61 - make-doc.js

```

1 /**
2 * @file Generate source code book for JSLAB
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 * @version 0.0.1
7 */
8
9 const fs = require('fs');
10 const path = require('path');

```

```

11 const cp = require("child_process");
12 const rimraf = require('rimraf');
13
14 require('../init-config.js');
15
16 console.log(`[make-source-code-book.js] Started`);
17 var t = performance.now();
18
19 var package_data = JSON.parse(fs.readFileSync('package.json'));
20 var app_version = package_data.version;
21 var year = new Date().getFullYear();
22
23 var levels = ['\\section', '\\subsection', '\\subsubsection'];
24 var latex = ',' ;
25
26 var SOURCE_CODE_BOOK_FILES_EXCLUDE = config.SOURCE_CODE_BOOK_FILES_EXCLUDE.map
27     (p => path.resolve(p));
28
29 // Helper function for excluding
30 function isExcluded(absolutPath) {
31     if(SOURCE_CODE_BOOK_FILES_EXCLUDE.includes(absolutPath)) return true;
32     return SOURCE_CODE_BOOK_FILES_EXCLUDE.some(dir => absolutPath.startsWith(
33         dir + path.sep));
34 }
35
36 // Helper function to escape LaTeX special characters
37 function escapeLatex(string) {
38     if(typeof string !== 'string') {
39         return string;
40     }
41     return string
42         .replace(/\\/g, '\\textbackslash{}')
43         .replace(/\&/g, '\\&')
44         .replace(/\%/g, '\\%')
45         .replace(/\$/g, '\\'
46 % -----
47 % End of document
48 % -----
49 \end{document}
50     .replace(/\#/g, '\\#')
51     .replace(/\_g, '\\_')
52     .replace(/\{/g, '\\{')
53     .replace(/\}/g, '\\}')
54     .replace(/\~/g, '\\textasciitilde{}')
55     .replace(/\^/g, '\\textasciicircum{}')
56     .replace(/\`/g, '\\textasciigrave{}');
57 }
58 /**
59 * Recursively traverses directories and reads file contents.
60 * @param {string[]} pathsList - List of file or folder paths to process.
61 * @returns {Promise<Object>} - An object representing the directory structure
62     with file details.
63 */

```



```
63  async function buildFileStructure(pathsList) {
64    const result = {root:{}};
65
66    for(const itemPath of pathsList) {
67      const absolutePath = path.resolve(itemPath);
68      if(isExcluded(absolutePath)) continue;
69
70      const stats = await fs.promises.lstat(absolutePath);
71
72      if(stats.isDirectory()) {
73        // Recursively process the directory
74        result[path.basename(absolutePath)] = await traverseDirectory(
75          absolutePath);
76      } else if(stats.isFile()) {
77        result['root'][path.basename(absolutePath)] = await fs.promises.readFile(
78          (absolutePath, 'utf-8'));
79      }
80    }
81  }
82
83 /**
84 * Helper function to traverse a directory recursively.
85 * @param {string} dirPath - The directory path to traverse.
86 * @returns {Promise<Object>} - An object representing the directory structure
87 */
88 async function traverseDirectory(dirPath) {
89   if(isExcluded(dirPath)) return {};
90
91   const dirContents = await fs.promises.readdir(dirPath, { withFileTypes: true
92     });
93   const dirObject = {};
94
95   // Separate files and directories
96   const files = dirContents.filter(dirent => dirent.isFile());
97   const directories = dirContents.filter(dirent => dirent.isDirectory());
98
99   // Process files first
100  for(const file of files) {
101    const fullPath = path.join(dirPath, file.name);
102    if(isExcluded(fullPath)) continue;
103    dirObject[file.name] = await fs.promises.readFile(fullPath, 'utf-8');
104  }
105
106  // Then process directories
107  for(const directory of directories) {
108    const fullPath = path.join(dirPath, directory.name);
109    if(isExcluded(fullPath)) continue;
110    dirObject[directory.name] = await traverseDirectory(fullPath);
111  }
112
113  return dirObject;
}
```

```

114
115  /**
116   * Determines the programming language based on the file extension.
117   * @param {string} filename - The name of the file (e.g., "main.cpp").
118   * @returns {string|null} - The corresponding language name or null if unknown
119   */
120  function getLanguage(filename) {
121    const extensionMatch = filename.match(/\.([^.]+)$/);
122    if (!extensionMatch) {
123      return null;
124    }
125
126    const extension = extensionMatch[1].toLowerCase();
127
128    // Mapping of extensions to languages
129    const extensionToLanguageMap = {
130      'cpp': 'C++',
131      'h': 'C++',
132      'js': 'JavaScript',
133      'json': 'JavaScript',
134      'gyp': 'JavaScript',
135      'css': 'CSS',
136      'html': 'HTML',
137    };
138
139    return extensionToLanguageMap[extension] || null;
140  }
141
142  /**
143   * Recursively traverses an object and performs actions based on the type of
144   * each property.
145   * @param {Object} obj - The object to traverse.
146   * @param {Function} callback - A function to execute on each property.
147   * @param {number} [level=0] - The current recursion level (used internally).
148  */
149  function traverseObject(obj, level = 0) {
150    for(const key in obj) {
151      if(obj.hasOwnProperty(key)) {
152        if(typeof obj[key] === 'object' && obj[key] !== null) {
153          let latex += '\n%' + '-----'.repeat(4-level) + '\n' + levels[level]
154            + '{' + escapeLatex(key) + '}\\n\\n';
155          traverseObject(obj[key], level + 1);
156        } else {
157          latex += '\\begin{lstlisting}[style=${getLanguage(key)}Style, caption={$escapeLatex(key)}]'
158          ${obj[key]}
159          '\\end{lstlisting}'
160        }
161      }
162    }
163  }
164

```

```

165  (async function main() {
166    // Make tex
167    console.log(' Making source-code-book.tex');
168    var latex_template = fs.readFileSync('dev/source_code_template.tex', 'utf8')
169    ;
170    latex_template = latex_template.replaceAll('$JSLAB_APP_VERSION'
171    % -----
172    % End of document
173    %
174    \end{document}, 'v'+app_version+'');
175    latex_template = latex_template.replaceAll('$JSLAB_PUBLISH_YEAR'
176    %
177    %
178    % End of document
179    %
180    \end{document}, year);
181
182    var app_path = __dirname + '/../../';
183    var fileStructure = await buildFileStructure(config.SOURCE_CODE_BOOK_FILES);
184    traverseObject(fileStructure);
185    latex_template = latex_template.replaceAll('$JSLAB_SOURCE_CODE_DATA'
186    %
187    %
188    % End of document
189    %
190    \end{document}, latex);
191    fs.writeFileSync('docs/source-code-book.tex', latex_template);
192
193    // Make pdf
194    console.log(' Making source-code-book.pdf');
195
196    var dir = __dirname + '/../../dev/latex/';
197    var file = dir + 'source-code-book.tex';
198    fs.mkdirSync(dir, { recursive: true });
199    fs.writeFileSync('docs/source-code-book.tex', file);
200
201    for(var i = 0; i < config.SOURCE_CODE_BOOK_LATEX_RERUNS_NUMBER; i++) {
202      console.log(' Run ' + (i+1) + '/' + config.
203                  SOURCE_CODE_BOOK_LATEX_RERUNS_NUMBER);
204      try {
205        cp.execSync('cd ' + dir + '& pdflatex source-code-book.tex --
206                    interaction=nonstopmode', { cwd: dir });
207      } catch(e) {
208        if(!fs.existsSync(dir + 'source-code-book.pdf') || (e.output &&
209          e.output.toString().trim().split('\n').pop().toLowerCase().includes(
210            'fatal'))) {
211          console.log(' LaTeX compile failed! Add path to pdflatex to
212                      environment variables and try again with MiKTeX 24.1 or later.
213                      Output: ');
214          if(e.output) {
215            console.log(e.output.toString());
216          } else {
217            console.log(e);
218          }
219        }
220      }
221    }

```

```

214         break;
215     }
216   }
217 }
218 if(fs.existsSync(dir + 'source-code-book.pdf')) {
219   fs.copyFileSync(dir + 'source-code-book.pdf', 'docs/source-code-book.pdf')
220   ;
221 }
222 rimraf.sync(dir);
223 console.log('[make-source-code-book.js.js] ' + ((performance.now() - t) / 1000).
224  toFixed(3) + ' s');
225 })();

```

Listing 62 - make-source-code-book.js

```

1 /**
2  * @file Prepare libs
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7 */
8
9 // Import modules
10 const fs = require('fs');
11 const rimraf = require('rimraf');
12 const path = require('path');
13 const { extractFull } = require('node-7z');
14 const bin_path = require('7zip-bin').path7za;
15
16 require('../init-config.js');
17
18 console.log('[prepare-libs.js] Started');
19 var t = performance.now();
20
21 async function main() {
22   for(const lib of config.COMPRESSED_LIBS) {
23     const lib_dir = path.join('./lib', lib);
24     const lib_7z = path.join('./lib', `${lib}.7z`);
25
26     // Check if lib folder exists locally
27     if(!fs.existsSync(lib_dir)) {
28       await extractWithProgress(lib, lib_7z, './lib');
29     } else {
30       console.log(`Lib ${lib} already uncompressed.`);
31     }
32   }
33
34   console.log('\nAll libraries ready.');
35   console.log(`[prepare-libs.js] Execution done in ${((performance.now() - t) / 1000).toFixed(3)} s`);
36 }
37
38 main();
39

```

```

40 // Function to extract .7z with progress
41 async function extractWithProgress(lib, archive, dest) {
42     return new Promise((resolve, reject) => {
43         const extractor = extractFull(archive, dest, {
44             $bin: bin_path,
45             $progress: true
46         });
47
48         extractor.on('progress', (progress) => {
49             const percent = progress.percent.toFixed(0);
50             process.stdout.clearLine(0);
51             process.stdout.cursorTo(0);
52             process.stdout.write(`Extracting ${lib}: ${percent}%`);
53         });
54
55         extractor.on('end', () => {
56             process.stdout.clearLine(0);
57             process.stdout.cursorTo(0);
58             console.log(`Extraction of ${lib} complete.`);
59             fs.unlinkSync(archive);
60             resolve();
61         });
62
63         extractor.on('error', (err) => {
64             console.error(`Extraction error for ${lib}:`, err);
65             fs.unlinkSync(archive);
66             reject(err);
67         });
68     });
69 }

```

Listing 63 - prepare-libs.js

```

1 /**
2  * @file Upload source code
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7 */
8
9 // Import modules
10 const fs = require('fs');
11 const dircompare = require('dir-compare');
12 const path = require('path');
13
14 require('../init-config.js');
15
16 console.log('[upload-source-code.js] Started');
17 var start = performance.now();
18
19 if (!process.env.SERVER_PATH) {
20     console.error('Environment variable SERVER_PATH must be defined!');
21 }
22 if (!process.env.SERVER_LIBS_PATH) {
23     console.error('Environment variable SERVER_LIBS_PATH must be defined!');

```

```

24 }
25
26 // Variables
27 const options = {
28   compareSize: config.UPLOAD_COMPARE_SIZE,
29   compareContent: config.UPLOAD_COMPARE_CONTENT,
30   compareDate: config.UPLOAD_COMPARE_DATE,
31   excludeFilter: config.SOURCE_UPLOAD_EXCLUDE.join(' ', '')
32 };
33
34 // Compare directories
35 console.log('Comparing directories... ');
36 var result = dircompare.compareSync('..', config.SERVER_SOURCE_PATH, options);
37
38 console.log(`Statistics: \n Equal entries: ${result.equal}, \n Distinct entries: ${result.distinct}, \n Left only entries: ${result.left}, \n Right only entries: ${result.right}, \n Differences: ${result.differences}`);
39
40 if(result.differences) {
41   // Analyzing different files
42   console.log('Analyzing different files... ');
43   result.diffSet.forEach(function(dif) {
44     if(dif.type1 === 'file' || dif.type2 === 'file') {
45       if(dif.state !== 'equal') {
46         switch(dif.state) {
47           case 'distinct':
48             if(!config.UPLOAD_COMPARE_SIZE_ON_DISTINCT || dif.date1 >= dif.date2) {
49               console.log(`File ${path.join(dif.relativePath, dif.name1)} changed.`);
50             } else {
51               console.log(`File ${path.join(dif.relativePath, dif.name1)} last modified on server.`);
52             }
53             break;
54           case 'left':
55             console.log(`File ${path.join(dif.relativePath, dif.name1)} missing on server.`);
56             break;
57           case 'right':
58             console.log(`File ${path.join(dif.relativePath, dif.name2)} exists only on server.`);
59             break;
60           default:
61             console.log('Unhandled state for: ');
62             console.log(dif);
63           }
64         }
65       if(['distinct', 'left'].includes(dif.state)) {
66         if(!dif.date2 || !config.UPLOAD_COMPARE_SIZE_ON_DISTINCT || dif.date1 >= dif.date2) {
67           console.log('Uploading file to server... ');
68         }
69         if(!dif.date2) {
70

```

```

71         fs.mkdirSync(path.join(config.SERVER_SOURCE_PATH, dif.
72             relativePath), { recursive: true });
73         fs.writeFileSync(path.join(dif.path1, dif.name1),
74             path.join(config.SERVER_SOURCE_PATH, dif.relativePath, dif.
75                 name1));
76     } else {
77         fs.mkdirSync(dif.path2, { recursive: true });
78         fs.writeFileSync(path.join(dif.path1, dif.name1),
79             path.join(dif.path2, dif.name2));
80     }
81     //console.log(dif);
82 }
83 );
84 );
85 } else {
86     console.log('No different files ...');
87 }
88 let d = new Date(), t = ((performance.now() - start) / 1000).toFixed(3);
89 console.log('[upload-source-code.js] Execution done in ${t}s - ${'0'+d.
90     getHours()}.slice(-2)}:${('0'+d.getMinutes()).slice(-2)}:${('0'+d.
91     getSeconds()}.slice(-2)}.${('00'+d.getMilliseconds()).slice(-3)} ${('0'+d.
92     getDate()}.slice(-2)}/${('0' + (d.getMonth() + 1)).slice(-2)}/${d.getFullYear
93     ()}');
```

Listing 64 - upload-source-code.js

6.3 editor

```

1 /**
2  * @file JSLAB editor module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB editor.
10 */
11 class PRDC_JSLAB_EDITOR {
12
13 /**
14  * Initializes the editor, setting up event listeners for UI interactions
15  * and window controls.
16  * @param {object} win The window object representing the current Electron
17  * window.
18 */
19 constructor(win) {
20     var obj = this;
21     this.win = win;
22
23     this.fullscreen = false;
```

```

22 // On menu click
23 $("#save-menu").click(function() { obj.win.script_manager.saveScript(); });
24 ;
25 $("#save-as-menu").click(function() { obj.win.script_manager.saveAsScript();
26 () });
27 $("#open-menu").click(function() { obj.win.script_manager.openScriptFile();
28 });
29 ;
30 $("#run-menu").click(function() { obj.win.script_manager.runScript(); });
31 $("#new-tab").click(function() { obj.win.script_manager.createScript();
32 });
33 ;
34 $("#new-script").click(function() { obj.win.script_manager.createScript();
35 });
36 ;
37 $("#close-dialog-save").click(function() { obj.win.script_manager.
38 closingDialogButton(2); });
39 $("#close-dialog-discard").click(function() { obj.win.script_manager.
40 closingDialogButton(1); });
41 $("#close-dialog-cancel").click(function() { obj.win.script_manager.
42 closingDialogButton(0); });
43 ;
44 $("#search-dialog-menu").click(function() {
45     obj.win.script_manager.openSearchDialog();
46     obj.win.editor_more_popup.close();
47 });
48 ;
49 // Keydown actions
50 document.addEventListener("keydown", function(e) {
51     if(e.key === 'F11') {
52         obj.toggleFullscreen();
53     } if(e.key === 'F12') {
54         ipcRenderer.send('MainProcess', 'show-dev-tools');
55     } else if(e.ctrlKey && e.key.toLowerCase() === "n") {
56         obj.win.script_manager.createScript();
57     } else if(e.ctrlKey && e.key === "F4") {
58         obj.win.script_manager.removeScriptByTab(obj.win.script_manager.
59             active_tab);
60     } else if(e.key === "F5") {
61         obj.win.script_manager.runScript();
62     } else if(e.ctrlKey && e.key.toLowerCase() === "o") {
63         obj.win.script_manager.openScriptFile();
64     } else if(e.ctrlKey && e.key.toLowerCase() === "s" && !e.shiftKey) {
65         obj.win.script_manager.saveScript();
66     } else if(e.ctrlKey && e.key.toLowerCase() === "s" && e.shiftKey) {
67         obj.win.script_manager.saveAsScript();
68 }

```



```

120  * Displays a message to the user through the application's main window. Can
121   * optionally request to focus the window.
122  * @param {string} msg The message to display.
123  * @param {boolean} [focus=true] Optional. Whether to focus the application
124   * window when displaying the message. Defaults to true.
125  */
126 disp(msg, focus = true) {
127   ipcRenderer.send("MainWindow", "editor-disp", [msg, focus]);
128 }
129 /**
130  * Displays an internal message within the application's main window. Can
131   * optionally request to focus the window.
132  * @param {string} msg - The internal message to display.
133  * @param {boolean} [focus=true] - Optional. Whether to focus the
134   * application window when displaying the message. Defaults to true.
135  */
136 dispInternal(msg, focus = true) {
137   ipcRenderer.send("MainWindow", "disp-internal", [msg, focus]);
138 }
139 /**
140  * Displays an error message through the application's main window.
141   * Typically used for internal errors.
142  * @param {string} msg The error message to display.
143  */
144 errorInternal(msg) {
145   ipcRenderer.send("MainWindow", "internal-error", msg);
146 }
147
148 exports.PRDC_JSLAB_EDITOR = PRDC_JSLAB_EDITOR;

```

Listing 65 - editor.js

```

1 /**
2  * @file JSLAB editor window init file
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 // Modules
9 // -----
10 const { ipcRenderer } = require('electron');
11
12 const helper = require('../js/helper.js');
13 require('../js/init-config.js');
14 const { PRDC_APP_LOGGER } = require('../lib/PRDC_APP_LOGGER/PRDC_APP_LOGGER');
15 const { PRDC_JSLAB_LANGUAGE } = require('../js/language');
16
17 global.app_path = process.argv.find(e => e.startsWith('--app-path=')).split('=')
18   [1].replace(/\js\?$/,'');
19 const { PRDC_JSLAB_WIN_EDITOR } = require('../js/editor/win-editor');
20

```



```
21 // Start log
22 const log_file = ipcRenderer.sendSync('sync-message', 'get-log-file');
23 const app_logger = new PRDC_APP_LOGGER(log_file);
24
25 // Global variables
26 var language = new PRDC_JSLAB_LANGUAGE();
27 var win_editor = new PRDC_JSLAB_WIN_EDITOR();
28
29 // When document is ready
30 // -----
31 ready(function() {
32     // Jquery ready
33     $(document).ready(function() {
34         win_editor.onReady();
35     });
36 });


```

Listing 66 - init-editor.js

```
1 /**
2  * @file JSLAB editor script manager module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const { PRDC_JSLAB_EDITOR_SCRIPT } = require('./script');
9
10 const fs = require('fs');
11 const { pathEqual } = require('path-equal');
12 const Store = require('electron-store');
13 const { ESLint } = require("eslint");
14 const path = require("path");
15
16 const store = new Store();
17
18 /**
19 * Class for JSLAB editor script .
20 */
21 class PRDC_JSLAB_EDITOR_SCRIPT_MANAGER {
22
23 /**
24 * Initializes the script manager with references to the application window
25 * and sets up event listeners for tab interactions.
26 * @param {object} win The application window where the editor is hosted.
27 */
28 constructor(win) {
29     var obj = this;
30     this.win = win;
31
32     this.scripts = [];
33     this.last_script_paths = [];
34     this.last_active_script;
35     this.active_tab;
36     this.close_window = false;
37     this.tabs = new PRDC_TABS();
```

```
37
38     this.eslint = new ESLint(config.LINT_OPTIONS);
39
40     // Tabs
41     this.tabs_cont = document.querySelector(".tabs");
42     this.tabs.init(this.tabs_cont);
43
44     // On tab add
45     this.tabs_cont.addEventListener("tabAdd", function({ detail }) {
46         $("#close-dialog-cont").hide();
47         detail.tabEl.onmousedown = function(e) {
48             if(e && (e.which === 2 || e.button === 4)) {
49                 obj.removeScriptByTab(detail.tabEl);
50             }
51         };
52     });
53
54     // On tab remove
55     this.tabs_cont.addEventListener("tabRemove", function() {
56         if(obj.scripts.length === 0) {
57             if(obj.close_window) {
58                 store.set("last_script_paths", obj.last_script_paths);
59                 store.set("last_active_script", obj.last_active_script);
60                 ipcRenderer.send("MainProcess", "close-editor");
61             } else {
62                 obj.createScript();
63             }
64         }
65     });
66
67     // On tab close
68     this.tabs_cont.addEventListener("tabClose", function({ detail }) {
69         obj.removeScriptByTab(detail.tabEl);
70     });
71
72     // On tab change
73     this.tabs_cont.addEventListener("activeTabChange", function({ detail }) {
74         if(obj.active_tab !== undefined) {
75             // Update code to last active_tab
76             var [script] = obj.getScriptByTab(obj.active_tab);
77             if(script !== undefined) {
78                 script.update();
79             }
80         }
81         obj.active_tab = detail.tabEl;
82         var [script] = obj.getScriptByTab(obj.active_tab);
83         if(script !== undefined) {
84             if(!script.closing) {
85                 $("#close-dialog-cont").hide();
86             }
87             script.show();
88         }
89         obj.updateActiveExtension(script);
90     });
91
```

```

92
93 // On start up
94 var argv = ipcRenderer.sendSync('sync-message', 'get-process-arguments');
95 var file_path = argv.find(arg => arg.endsWith('.jsl')); // path as
96   argument
97 var script_paths = store.get("last_script_paths"); // last opened scripts
98 if(file_path) {
99   if(script_paths) {
100     script_paths.push(file_path);
101   } else {
102     script_paths = [file_path];
103   }
104 if(script_paths) {
105   script_paths.forEach(function(script_path) {
106     var lstat = fs.lstatSync(script_path, {throwIfNoEntry: false});
107     if(lstat && lstat.isFile()) {
108       var [script, i] = obj.getScriptByPath(script_path);
109       if(script == undefined) {
110         obj.createScript(script_path);
111       } else {
112         script.activate();
113       }
114     }
115   });
116 }
117 if(this.scripts.length > 0) {
118   if(file_path) {
119     this.activateScriptByPath(file_path);
120   } else {
121     var last_active_script = store.get("last_active_script");
122     if(last_active_script) {
123       this.activateScriptByPath(last_active_script);
124     }
125   }
126   this.win.showEditor();
127 } else {
128   // New tab
129   this.createScript();
130 }
131 }
132 }

133 /**
134 * Saves the currently active script.
135 */
136 saveScript() {
137   this.getScriptByTab(this.active_tab)[0].save();
138 }
139

140 /**
141 * Saves the currently active script under a new file name.
142 */
143 saveAsScript() {
144   this.getScriptByTab(this.active_tab)[0].saveAs();
145 }
```

```
146 }
147
148 /**
149 * Opens search dialog for currently active script.
150 */
151 openSearchDialog() {
152     this.getScriptByTab(this.active_tab)[0].openSearchDialog();
153 }
154
155 /**
156 * Compiles arduino code.
157 */
158 compileArduino() {
159     this.getScriptByTab(this.active_tab)[0].compileArduino();
160 }
161
162 /**
163 * Uploads arduino code.
164 */
165 uploadArduino() {
166     this.getScriptByTab(this.active_tab)[0].uploadArduino();
167 }
168
169 /**
170 * Opens a script file from the filesystem into the editor.
171 */
172 openScriptFile() {
173     var obj = this;
174     let options = {
175         title: language.currentString(146),
176         buttonLabel: language.currentString(147),
177         filters: [
178             { name: "All Files", extensions: ["*"] },
179         ],
180     };
181
182     ipcRenderer.invoke("dialog", "showOpenDialog", options)
183         .then(function(result) {
184             if(result.canceled) return obj.win.editor.disp("@editor/openScriptFile"
185                 : "+language.string(132))";
186             var open_path = result.filePaths[0];
187             var [script, i] = obj.getScriptByPath(open_path);
188             if(script == undefined) {
189                 obj.createScript(open_path);
190             } else {
191                 script.activate();
192             }
193         }).catch(function(err) {
194             obj.win.editor.errorInternal(err);
195         });
196 }
197 /**
198 * Opens a script by its path if not already open, or activates it if it is
already open.
```

```

199   * @param {Array<string , number>} data - An array where the first element is
200   *      the script path and the second is the line number.
201   */
202 openScript(data) {
203   var script_path = data[0];
204   var script_lineno = data[1];
205   var script_charpos = data[2];
206   var lstat = fs.lstatSync(script_path, {throwIfNoEntry: false});
207   if(lstat && lstat.isFile()) {
208     var [script, i] = this.getScriptByPath(script_path);
209     if(script == undefined) {
210       this.createScript(script_path, script_lineno, script_charpos);
211     } else {
212       this.win.editor.disp("@editor/openScript: "+language.string(133)+" "+
213         script_path+" "+language.string(134)+"!", false);
214       script.activate();
215       script.setLine(script_lineno, script_charpos);
216     }
217   }
218 /**
219 * Executes the currently active script.
220 */
221 runScript() {
222   this.getScriptByTab(this.active_tab)[0].run();
223 }
224 /**
225 * Creates a new script tab and editor instance, loading the script from the
226 * given path.
227 * @param {string} path - The file path of the script to load.
228 * @param {number} lineno - The line number to navigate to in the newly
229 *      created script.
230 * @param {number} charpos - The char position to navigate to in the newly
231 *      created script.
232 */
233 createScript(path, lineno, charpos) {
234   this.tab = this.tabs.addTab();
235   var script = new PRDC_JSLAB_EDITOR_SCRIPT(this.win, this, path, this.tab);
236   script.setLine(lineno, charpos);
237   this.scripts.push(script);
238 /**
239 * Toggles the comment state of the selected lines in the currently active
240 *      script tab.
241 * This action uses the active tab's associated script editor to apply or
242 *      remove comments.
243 */
244 toggleComment() {
245   this.getScriptByTab(this.active_tab)[0].toggleComment();
246 }
247 /**

```

```
247     * Activates the script associated with the given tab element .
248     * @param {HTMLElement} tab The tab element associated with the script to
249     * activate .
250     */
251     activateScriptByTab(tab) {
252         this.tabs.setCurrentTab(tab);
253     }
254
255     /**
256     * Activates the script associated with the given filesystem path .
257     * @param {string} script_path The path to the script to activate .
258     */
259     activateScriptByPath(script_path) {
260         var [script , i] = this.getScriptByPath(script_path);
261         if(script != undefined) {
262             this.activateScriptByTab(script.tab);
263         }
264     }
265
266     /**
267     * Retrieves the script object and its index associated with the given tab
268     * element .
269     * @param {HTMLElement} tab The tab element associated with the script .
270     * @returns {Array} An array containing the script object and its index in
271     * the scripts array .
272     */
273     getScriptByTab(tab) {
274         var i = this.scripts.findIndex(function(script) {
275             return script.tab == tab;
276         });
277         if(i > -1) {
278             return [this.scripts[i] , i];
279         } else {
280             return [undefined , -1];
281         }
282     }
283
284     /**
285     * Retrieves the script object and its index associated with the given
286     * filesystem path .
287     * @param {string} script_path The path to the script .
288     * @returns {Array} An array containing the script object and its index in
289     * the scripts array .
290     */
291     getScriptByPath(script_path) {
292         var i = this.scripts.findIndex(function(script) {
293             return pathEqual(script.path , script_path);
294         });
295         if(i > -1) {
296             return [this.scripts[i] , i];
297         } else {
298             return [undefined , -1];
299         }
300     }
```

```

297  /**
298   * Updates the name displayed on a script's tab.
299   * @param {HTMLElement} tab The tab element associated with the script.
300   * @param {string} name The new name to display on the tab.
301   */
302 setScriptNameByTab(tab, name) {
303   this.tabs.updateTab(tab, {
304     title: name,
305     favicon: false
306   });
307 }
308
309 /**
310  * Removes the script associated with the given tab element from the manager
311  * and UI.
312  * @param {HTMLElement} tab The tab element associated with the script to
313  * remove.
314  */
315 removeScriptByTab(tab) {
316   var [script, i] = this.getScriptByTab(tab);
317   script.activate();
318   if(script.remove()) {
319     this.scripts[i].removeCodeEditor();
320     this.scripts.splice(i, 1);
321     this.tabs.removeTab(tab);
322   }
323 }
324 /**
325  * Initiates the closure of all open scripts, optionally persisting their
326  * state for future sessions.
327  */
328 closeAllScripts() {
329   var obj = this;
330   this.close_window = true;
331   this.last_script_paths = [];
332   this.last_active_script = this.active_tab.getAttribute('title');
333
334   this.tabs_cont.querySelectorAll('.tabs-content .tab').forEach(function(tab) {
335     var path = tab.getAttribute('title');
336     if(path) {
337       obj.last_script_paths.push(path);
338     }
339   });
340   ipcRenderer.send("MainProcess", "focus-win");
341   $("#close-dialog-cancel").hide();
342   for(var i = this.scripts.length - 1; i >= 0; i--) {
343     var tab = this.scripts[i].tab;
344     if(this.scripts[i].remove()) {
345       this.scripts[i].removeCodeEditor();
346       this.scripts.splice(i, 1);
347       this.tabs.removeTab(tab);
348     }
349   }
350 }
```

```

348     }
349 }
350
351 /**
352 * Checks if a script with the given filesystem path is already open in the
353 * editor.
354 * @param {string} script_path The path to the script to check.
355 * @returns {boolean} True if the script is already open, false otherwise.
356 */
357 checkScriptOpenByPath(script_path) {
358   var [script, i] = this.getScriptByPath(script_path);
359   return !(script === undefined);
360 }
361 /**
362 * Handles user interaction with the script closing dialog, determining
363 * whether to save changes, discard them, or cancel the closure.
364 * @param {number} s - button state
365 */
366 closingDialogButton(s) {
367   var [script, i] = this.getScriptByTab(this.active_tab);
368   var tab = this.scripts[i].tab;
369   var script_path = this.scripts[i].path;
370   if(s == 2 || s == 1) {
371     if(s == 2) {
372       this.scripts[i].save();
373       script_path = this.scripts[i].path;
374     }
375     this.scripts[i].removeCodeEditor();
376     this.scripts.splice(i, 1);
377     this.tabs.removeTab(tab);
378   } else {
379     this.scripts[i].closing = false;
380     $("#close-dialog-cont").hide();
381   }
382   if(this.close_window && script_path !== undefined) {
383     this.last_script_paths.push(script_path);
384   }
385 }
386 /**
387 * Updates body class based on active extension of script
388 */
389 updateActiveExtension(script) {
390   if(script && (typeof script.path === 'string' || script.path instanceof
391     String)) {
392     $(document.body).attr("class", 'file-' + path.extname(script.path).
393       substring(1));
394   } else {
395     $(document.body).attr("class", '');
396   }
397 }
```

Listing 67 - script-manager.js

```
49     theme: "notepadpp",
50     rulers: [{ color: "#aff", column: 75, lineStyle: "solid" }],
51     indentUnit: 2,
52     tabSize: 2,
53     lineNumbers: true,
54     lineWrapping: true,
55     styleActiveLine: true,
56     matchBrackets: true,
57     gutter: true,
58     gutters: [
59       "CodeMirror-linenumbers",
60       "CodeMirror-foldgutter",
61       "CodeMirror-lint-markers",
62     ],
63     foldGutter: true,
64     searchDialog: true,
65     highlightSelectionMatches: { annotateScrollbar: true },
66   });
67
68   CodeMirror.keyMap.default["Shift-Tab"] = "indentLess";
69   CodeMirror.keyMap.default["Tab"] = "indentMore";
70   CodeMirror.keyMap.default["Ctrl-F"] = "showSearchDialog";
71   CodeMirror.keyMap.default['Ctrl-G'] = 'findNext';
72   CodeMirror.keyMap.default['Shift-Ctrl-G'] = 'findPrev';
73   CodeMirror.keyMap.default['Shift-Ctrl-F'] = 'replace';
74   CodeMirror.keyMap.default['Shift-Ctrl-R'] = 'replaceAll';
75   CodeMirror.keyMap.default['Ctrl-/'] = 'toggleComment';
76
77   // Keypress events
78   this.code_editor.on("keypress", function(cm, event) {
79     if(
80       !cm.state.completionActive &&
81       !event.ctrlKey &&
82       event.key != "Enter" &&
83       event.key != ";" &&
84       event.key != " " &&
85       (event.key != "{" & (event.key != "}"))
86     ) {
87       CodeMirror.commands.autocomplete(cm, null, { completeSingle: false });
88     }
89   });
90
91   // Drop events
92   this.code_editor.on("drop", function(data, e) {
93     e.preventDefault();
94   });
95
96   this.code_editor.setValue(this.code);
97   this.code_editor.clearHistory();
98   this.code_editor.on("change", function() {
99     obj.codeChanged();
100   });
101
102   this.show();
103 }
```

```
104
105  /**
106   * Loads the code from the specified script path into the editor.
107   * @param {string} script_path The path to the script file to be loaded.
108   */
109  loadCode(script_path) {
110    try {
111      var data = fs.readFileSync(script_path);
112      this.code = data.toString();
113    } catch (err) {
114      this.win.editor.errorInternal(err.stack);
115    }
116    this.saved_code = this.code;
117  }
118
119  /**
120   * Displays the editor for this script, focusing it and hiding other scripts
121   * editors.
122   */
123  show() {
124    $(".CodeMirror").hide();
125    if(this.closing) {
126      $("#close-file").text(this.name);
127      $("#close-dialog-cont").fadeIn(300, "linear");
128    }
129    $(this.code_editor.display.wrapper).show();
130    this.code_editor.focus();
131    this.updateEditorMode();
132  }
133
134  /**
135   * Updates the editor's stored code value based on the current content in
136   * the editor.
137   */
138  update() {
139    this.code = this.code_editor.getValue();
140  }
141
142  /**
143   * Saves the current script to its associated file path. If the script does
144   * not have a path, prompts for one.
145   * @returns {boolean} True if the save operation was successful, false
146   * otherwise.
147   */
148  save() {
149    if(this.path === undefined) {
150      if(this.isActive()) {
151        this.update();
152      }
153      this.tab.classList.remove("changed");
154      if(this.code !== this.saved_code) {
155        try {
156          fs.writeFileSync(this.path, this.code);
157        } catch (err) {
158          this.win.editor.errorInternal(err.stack);
159        }
160      }
161    }
162  }
```

```

155         return false;
156     }
157     this.saved_code = this.code;
158     this.tab.classList.remove("changed");
159     return true;
160 } else {
161     return true;
162 }
163 } else {
164     return this.saveAs();
165 }
166 }
167
168 /**
169 * Saves the current script to a new file, prompting the user for the file
170 * path.
171 * @returns {boolean} True if the save operation was successful, false
172 * otherwise.
173 */
174 saveAs() {
175     var script_path;
176     if(this.path === undefined) {
177         script_path = "script.jsl";
178     } else {
179         script_path = this.path;
180     }
181     let options = {
182         title: language.currentString(144),
183         defaultPath: script_path,
184         buttonLabel: language.currentString(145),
185         filters: [
186             { name: "All Files", extensions: ["*"] },
187         ],
188     };
189
190     if(this.isActive()) {
191         this.update();
192     }
193     script_path = ipcRenderer.sendSync("dialog", "showSaveDialogSync", options);
194     if(script_path === undefined) {
195         this.win.editor.disp("@editor/saveAs: "+language.string(129));
196         return false;
197     }
198     if(script_path != this.path) {
199         if(!this.script_manager.checkScriptOpenByPath(script_path)) {
200             try {
201                 fs.writeFileSync(script_path, this.code);
202             } catch (err) {
203                 this.win.editor.errorInternal(err.stack);
204                 return false;
205             }
206             this.path = script_path;
207         }
208     }
209 }

```

```

207     this.name = this.path.replace(/^.*/[\\\]/, "");
208     this.tab.setAttribute("title", this.path);
209     this.tab.querySelector(".tab-title").innerHTML = this.name;
210     this.saved_code = this.code;
211     this.tab.classList.remove("changed");
212     this.updateEditorMode();
213     return true;
214   } else {
215     this.win.editor.errorInternal(language.string(130));
216     return false;
217   }
218 } else {
219   return false;
220 }
221 }

222 /**
223 * Prepares the script for removal, checking if changes are unsaved and
224 * potentially prompting the user.
225 * @returns {boolean} True if the script can be safely removed, false if
226 * there are unsaved changes.
227 */
228 remove() {
229   this.closing = true;
230   if(this.isActive()) {
231     this.update();
232   }
233   if(this.code.replace(/\r/g, '') != this.saved_code.replace(/\r/g, '')) {
234     // Fade in window
235     ipcRenderer.send("MainProcess", "fade-in-win");
236     if(this.isActive()) {
237       $("#close-file").text(this.name);
238       $("#close-dialog-cont").fadeIn(300, "linear");
239     }
240     return false;
241   } else {
242     return true;
243   }
244 }

245 /**
246 * Removes the code editor associated with this script from the DOM.
247 */
248 removeCodeEditor() {
249   $(this.code_editor.display.wrapper).remove();
250 }

251 /**
252 * Checks if the script's tab is currently active in the UI.
253 * @returns {boolean} True if this script's tab is active, false otherwise.
254 */
255 isActive() {
256   return this.tab == this.script_manager.active_tab;
257 }
258 }
```

```
259
260  /**
261   * Activates this script's tab in the UI, showing its editor and hiding
262   * others.
263   */
264  activate() {
265    if(!this.isActive()) {
266      this.script_manager.activateScriptByTab(this.tab);
267      this.show();
268    }
269  }
270 /**
271  * Sets the cursor in the code editor to the specified line number.
272  * @param {number} lineno - The line number to navigate to (1-based index).
273  * @param {number} charpos - The char position to navigate to in the newly
274  * created script.
275  */
276 setLine(lineno, charpos = 0) {
277   if(isFinite(lineno)) {
278     this.code_editor.setCursor({ line: lineno - 1, ch: charpos - 1 });
279   }
280 }
281 /**
282  * Saves the script and then runs it , typically in an external execution
283  * environment.
284  * @param {array} lines The specific lines or sections of the script to
285  * execute , if applicable.
286  */
287 run(lines) {
288   if(this.save()) {
289     ipcRenderer.send("MainWindow", "run", [this.path, lines]);
290   } else {
291     this.win.editor.disp("@editor/run: "+language.string(131));
292   }
293 }
294 /**
295  * Marks the script as having unsaved changes , updating the UI accordingly.
296  */
297 codeChanged() {
298   this.tab.classList.add("changed");
299 }
300 /**
301  * Toggles the comment state of the currently selected lines in the code
302  * editor.
303  */
304 toggleComment() {
305   this.code_editor.execCommand('toggleComment');
306 }
307 /**
308  * Update editor mode based on script extension
```

```

309  */
310 updateEditorMode() {
311   var obj = this;
312   this.script_manager.updateActiveExtension(this);
313   if(typeof this.name === 'string' || this.name instanceof String) {
314     var file_extension = path.basename(this.name);
315     if(['.cpp', '.c', '.ino', '.h', '.hpp'].includes(file_extension.toLowerCase())))
316       this.code_editor.setOption("mode", "text/x-csrc");
317       this.code_editor.setOption("lint", {});
318       return;
319   }
320 }
321 this.code_editor.setOption("mode", "javascript");
322 this.code_editor.setOption("lint", {
323   getAnnotations: async function(text, callback) {
324     var results = await obj.script_manager.eslint.lintText(text);
325     callback(results[0].messages.map(message => ({
326       from: CodeMirror.Pos(message.line - 1, message.column - 1),
327       to: CodeMirror.Pos(
328         message.endLine ? message.endLine - 1 : message.line - 1,
329         message.endColumn ? message.endColumn - 1 : message.column
330       ),
331       severity: message.severity === 2 ? "error" : "warning",
332       message: message.message,
333     })));
334   },
335   async: true
336 });
337 }
338 /**
339 * Opens search dialog in the code editor.
340 */
341 openSearchDialog() {
342   this.code_editor.execCommand('showSearchDialog');
343 }
344
345 /**
346 * Compiles arduino code.
347 */
348 compileArduino() {
349   if(typeof this.path === 'string' || this.path instanceof String) {
350     ipcRenderer.send("MainWindow", "eval-command", ['compileArduino(${path.
351       dirname(this.path).replace(/\\\\(?!\\\\)/g, "\\\\"})}']);
352   } else {
353     this.win.editor disp("@editor/compileArduino: "+language.string(229));
354   }
355 }
356 /**
357 * Uploads arduino code.
358 */
359 uploadArduino() {
360   if(typeof this.path === 'string' || this.path instanceof String) {

```

```

362     ipcRenderer.send("MainWindow", "eval-command", [ 'uploadArduino( ${path.
363         dirname(this.path).replace(/\\(?!\\\)/g, "\\\\"})' )']);
364     } else {
365       this.win.editor.disp("@editor/uploadArduino: "+language.string(229));
366     }
367   }
368
369 exports.PRDC_JSLAB_EDITOR_SCRIPT = PRDC_JSLAB_EDITOR_SCRIPT;

```

Listing 68 - script.js

```

1  /**
2   * @file JSLAB Editor
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8 // Modules
9 // -----
10 const { ipcRenderer, webUtils } = require('electron');
11
12 const { PRDC_JSLAB_EDITOR } = require('./editor');
13 const { PRDC_JSLAB_EDITOR_SCRIPT_MANAGER } = require('./script-manager');
14
15 const { PRDC_POPUP } = require('../.../lib/PRDC_POPUP/PRDC_POPUP');
16
17 const fs = require('fs');
18
19 /**
20  * Class for JSLAB editor win.
21  */
22 class PRDC_JSLAB_WIN_EDITOR {
23
24   /**
25    * Create editor win.
26    */
27   constructor() {
28     var obj = this;
29
30     // Classes
31     this.editor = new PRDC_JSLAB_EDITOR(this);
32     this.script_manager = new PRDC_JSLAB_EDITOR_SCRIPT_MANAGER(this);
33     this.editor_more_popup = new PRDC_POPUP(document.getElementById('editor-
34       more-icon'),
35       document.getElementById('editor-more-popup')));
36
37     // Prevent redirects
38     preventRedirect();
39
40     // Events
41     // -----
42     // Toggle DevTools
43     document.addEventListener("keydown", function (e) {
        if (e.key == "F12") {

```

```

44         ipcRenderer.send("MainProcess", "toggle-dev-tools");
45     } else if(e.ctrlKey && e.key.toLowerCase() === 'c') {
46         if(obj.getSelectionText() === "") {
47             // No selected text
48             obj.editor.dispInternal(language.string(89));
49             ipcRenderer.send('SandboxWindow', 'stop-loop', true);
50             e.stopPropagation();
51             e.preventDefault();
52         }
53     });
54 );
55
56 // Error handle
57 // -----
58 window.addEventListener("unhandledrejection", function (e) {
59     obj.editor.errorInternal(e.reason.stack);
60     e.preventDefault();
61 });
62
63 window.addEventListener("error", function (e) {
64     obj.editor.errorInternal(e.error.stack);
65     e.preventDefault();
66 });
67
68 // On IPC message
69 ipcRenderer.on("EditorWindow", function(event, action, data) {
70     switch(action) {
71         case "open-script":
72             // On open script
73             obj.script_manager.openScript(data);
74             break;
75         case "close-all":
76             // On close all
77             obj.script_manager.closeAllScripts();
78             break;
79         case "set-language":
80             language.set(data);
81             break;
82         case "toggle-comment":
83             obj.script_manager.toggleComment();
84             break;
85     }
86 });
87
88 // On file drop
89 document.addEventListener("drop", function(e) {
90     e.stopPropagation();
91     e.preventDefault();
92     console.log(e);
93
94     for(const f of e.dataTransfer.files) {
95         obj.script_manager.openScript([webUtils.getPathForFile(f)]);
96     }
97 }, false);
98 }
```

```

99
100 /**
101 * Retrieves selected text within the application , if any.
102 * @return {string} The currently selected text.
103 */
104 getSelectionText () {
105   var text = "";
106   if(window.getSelection) {
107     text = window.getSelection().toString();
108   } else if(document.selection && document.selection.type != "Control") {
109     text = document.selection.createRange().text;
110   }
111   return text;
112 }
113
114 /**
115 * Method used to execute code when gui is ready.
116 */
117 onReady() {
118   // Fade in window
119   ipcRenderer.send('MainProcess', 'fade-in-win');
120 }
121
122 /**
123 * Show editor
124 */
125 showEditor() {
126   ipcRenderer.send("MainProcess", "show-editor");
127 }
128 }
129 exports.PRDC_JSLAB_WIN_EDITOR = PRDC_JSLAB_WIN_EDITOR;

```

Listing 69 - win-editor.js

6.4 main

```

1 /**
2  * @file JSLAB app module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const os = require('os');
9 const cp = require('child_process');
10
11 /**
12  * Class for JSLAB app.
13 */
14 class PRDC_JSLAB_APP {
15
16 /**
17  * Initializes application properties , fetching system and environment
18  * information relevant to the application .

```

```

18  * @param {object} win The window object representing the current Electron
19  * window.
20  */
21  constructor(win) {
22   var obj = this;
23   this.win = win;
24
25   this.version;
26   this.user;
27   this.documents_path;
28
29   // Platform specific code
30   this.computer_name;
31   switch(process.platform) {
32     case 'win32':
33       this.computer_name = process.env.COMPUTERNAME;
34       break;
35     case 'darwin':
36       this.computer_name = cp.execSync('scutil --get ComputerName')
37         .toString().trim();
38       break;
39     default:
40       this.computer_name = os.hostname();
41       break;
42   }
43   this.version = ipcRenderer.sendSync('sync-message', 'get-app-version');
44   this.documents_path = ipcRenderer.sendSync('sync-message',
45     'get-path', 'documents');
46   this.exe_path = ipcRenderer.sendSync('sync-message',
47     'get-path', 'exe');
48   this.user = os.userInfo().username + '@' + this.computer_name;
49 }
50 /**
51 * Generates a formatted string representing the current date and time,
52 * suitable for timestamps and logging.
53 * @param {number} [t=Date.now()] Optional timestamp to format, defaults to
54 * the current time if not provided.
55 * @returns {string} A string formatted to represent a date and time,
56 * structured as DD_MM_YYYY. HH_MM_SS_mmm.
57 */
58 getDateFullStr(t = Date.now()) {
59   var d = new Date(t);
60   return ('0' + d.getDate()).slice(-2) + "_" +
61     ('0' + (d.getMonth() + 1)).slice(-2) + "_" +
62     d.getFullYear() + '.' + ('0' + d.getHours()).slice(-2) + "_" +
63     ('0' + d.getMinutes()).slice(-2) + "_" +
64     ('0' + d.getSeconds()).slice(-2) + "_" +
65     ('00' + d.getMilliseconds()).slice(-3);
66 }
67
68 exports.PRDC_JSLAB_APP = PRDC_JSLAB_APP;

```

Listing 70 - app.js

```
1  /**
2   * @file JSLAB command history module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  const Store = require('electron-store');
9
10 const store = new Store();
11
12 /**
13  * Class for JSLAB command history.
14  */
15 class PRDC_JSLAB_COMMAND_HISTORY {
16
17 /**
18  * Initializes the command history.
19  * @param {object} win The window object representing the current Electron
20  * window.
21  */
22 constructor(win) {
23     var obj = this;
24     this.win = win;
25     this.history_cont = document.getElementById('command-history');
26     this.full_history = store.get('full_history') || [];
27     this.history = [];
28
29     this.N_history_max = Number(store.get('N_history_max'));
30     if (!isFinite(this.N_history_max)) {
31         this.setMaxSize(20);
32     }
33
34     this.renderHistory();
35
36     // Append initialization command
37     const init_cmd = `// JSLAB ${this.win.app.version}, ${new Date()} [${this.
38         win.app.user}]`;
39     this.updateHistory(init_cmd);
40
41     // History clear button click
42     const clear_button = document.querySelector('#command-history-options .
43         clear');
44     if (clear_button) {
45         clear_button.addEventListener('click', function() {
46             obj.clearHistory()
47         });
48     }
49
50     // Event delegation for command interactions
51     this.history_cont.addEventListener('click', function(e) {
52         if (e.target && !e.target.classList.contains('comment')) {
53             obj.selectCommand(e.target.textContent);
54         }
55     });
56 }
```

```

53
54   this.history_cont.addEventListener('dblclick', function(e) {
55     if(e.target && !e.target.classList.contains('comment')) {
56       obj.evalSelectedCommand(e.target.textContent);
57     }
58   });
59 }
60
61 /**
62 * Renders the entire history efficiently using DocumentFragment.
63 */
64 renderHistory() {
65   const fragment = document.createDocumentFragment();
66
67   this.full_history.forEach((cmd) => {
68     const div = document.createElement('div');
69     if(cmd.startsWith('//')) {
70       div.classList.add('comment');
71     }
72     div.textContent = cmd.replace(/\//g, '\\\\');
73     fragment.appendChild(div);
74   });
75
76   this.history_cont.appendChild(fragment);
77   this.scrollToBottom();
78 }
79
80 /**
81 * Adds a command to the history, updating the UI and internal storage
82 * accordingly.
83 * @param {string} cmd The command to be added to the history.
84 */
84 updateHistory(cmd) {
85   if(this.full_history.length >= this.N_history_max) {
86     this.full_history.shift();
87     if(this.history_cont.firstChild) {
88       this.history_cont.removeChild(this.history_cont.firstChild);
89     }
90   }
91
92   const div = document.createElement('div');
93   if(cmd.startsWith('//')) {
94     div.classList.add('comment');
95   } else {
96     this.history.unshift(cmd);
97   }
98   div.textContent = cmd;
99   this.history_cont.appendChild(div);
100  this.full_history.push(cmd);
101
102  this.scrollToBottom();
103  this.saveHistory();
104 }
105
106 /**

```

```
107     * Scrolls the history container to the bottom.
108     */
109    scrollToBottom() {
110      this.history_cont.scrollTop = this.history_cont.scrollHeight;
111    }
112
113    /**
114     * Selects a command from the history, placing it in the command input for
115     * potential modification and re-execution.
116     * @param {string} cmd The command to select.
117     */
118    selectCommand(cmd) {
119      const code_input = this.win.command_window.code_input;
120      code_input.setValue(cmd);
121      code_input.focus();
122      code_input.setCursor(code_input.lineCount(), 0);
123    }
124
125    /**
126     * Evaluates a selected command from the history, directly executing it
127     * without modification.
128     * @param {string} cmd The command to evaluate.
129     */
130    evalSelectedCommand(cmd) {
131      this.win.eval.evalCommand(cmd);
132    }
133
134    /**
135     * Clears the command history.
136     */
137    clearHistory() {
138      this.full_history = [];
139      this.history_cont.innerHTML = '';
140      this.saveHistory();
141    }
142
143    /**
144     * Saves the current history to storage asynchronously.
145     */
146    saveHistory() {
147      store.set('full_history', this.full_history);
148    }
149
150    /**
151     * Sets the maximum number of commands to retain in the history.
152     * @param {number} N - The desired maximum number of history entries
153     */
154    setMaxSize(N) {
155      if(N < 5) {
156        N = 5;
157      }
158      this.N_history_max = N;
159      $('#settings-container .N-history-max').val(N);
160      while(this.full_history.length >= this.N_history_max) {
161        this.full_history.shift();
162      }
163    }
164
165    /**
166     * Adds a command to the history.
167     * @param {string} cmd The command to add.
168     */
169    addCommand(cmd) {
170      this.full_history.push(cmd);
171      this.history_cont.innerHTML += cmd + '\n';
172    }
173
174    /**
175     * Deletes a command from the history.
176     * @param {number} index The index of the command to delete.
177     */
178    deleteCommand(index) {
179      this.full_history.splice(index, 1);
180      this.history_cont.innerHTML = this.full_history.join('\n');
181    }
182
183    /**
184     * Deletes all commands from the history.
185     */
186    clearAllCommands() {
187      this.full_history = [];
188      this.history_cont.innerHTML = '';
189    }
190
191    /**
192     * Gets the full history of commands.
193     */
194    getFullHistory() {
195      return this.full_history;
196    }
197
198    /**
199     * Gets the current history entry at the top of the stack.
200     */
201    getCurrentHistoryEntry() {
202      return this.full_history[0];
203    }
204
205    /**
206     * Gets the maximum number of history entries stored.
207     */
208    getMaxSize() {
209      return this.N_history_max;
210    }
211
212    /**
213     * Gets the total number of history entries stored.
214     */
215    getTotalHistoryEntries() {
216      return this.full_history.length;
217    }
218
219    /**
220     * Gets the history entry at a specific index.
221     * @param {number} index The index of the history entry to get.
222     */
223    getHistoryEntry(index) {
224      return this.full_history[index];
225    }
226
227    /**
228     * Checks if a specific command is in the history.
229     * @param {string} cmd The command to search for.
230     */
231    containsCommand(cmd) {
232      return this.full_history.includes(cmd);
233    }
234
235    /**
236     * Checks if the history is empty.
237     */
238    isEmpty() {
239      return this.full_history.length === 0;
240    }
241
242    /**
243     * Checks if the history has reached its maximum size.
244     */
245    isAtMaxSize() {
246      return this.full_history.length === this.N_history_max;
247    }
248
249    /**
250     * Checks if the history has any entries.
251     */
252    hasEntries() {
253      return this.full_history.length > 0;
254    }
255
256    /**
257     * Checks if the history has reached its minimum size.
258     */
259    isAtMinSize() {
260      return this.full_history.length === 0;
261    }
262
263    /**
264     * Checks if the history has reached its maximum length.
265     */
266    isAtMaxLength() {
267      return this.full_history.length === this.N_history_max;
268    }
269
270    /**
271     * Checks if the history has reached its minimum length.
272     */
273    isAtMinLength() {
274      return this.full_history.length === 0;
275    }
276
277    /**
278     * Checks if the history has reached its maximum capacity.
279     */
280    isAtCapacity() {
281      return this.full_history.length === this.N_history_max;
282    }
283
284    /**
285     * Checks if the history has reached its minimum capacity.
286     */
287    isAtCapacity() {
288      return this.full_history.length === 0;
289    }
290
291    /**
292     * Checks if the history has reached its maximum capacity.
293     */
294    isAtCapacity() {
295      return this.full_history.length === this.N_history_max;
296    }
297
298    /**
299     * Checks if the history has reached its minimum capacity.
300     */
301    isAtCapacity() {
302      return this.full_history.length === 0;
303    }
304
305    /**
306     * Checks if the history has reached its maximum capacity.
307     */
308    isAtCapacity() {
309      return this.full_history.length === this.N_history_max;
310    }
311
312    /**
313     * Checks if the history has reached its minimum capacity.
314     */
315    isAtCapacity() {
316      return this.full_history.length === 0;
317    }
318
319    /**
320     * Checks if the history has reached its maximum capacity.
321     */
322    isAtCapacity() {
323      return this.full_history.length === this.N_history_max;
324    }
325
326    /**
327     * Checks if the history has reached its minimum capacity.
328     */
329    isAtCapacity() {
330      return this.full_history.length === 0;
331    }
332
333    /**
334     * Checks if the history has reached its maximum capacity.
335     */
336    isAtCapacity() {
337      return this.full_history.length === this.N_history_max;
338    }
339
340    /**
341     * Checks if the history has reached its minimum capacity.
342     */
343    isAtCapacity() {
344      return this.full_history.length === 0;
345    }
346
347    /**
348     * Checks if the history has reached its maximum capacity.
349     */
350    isAtCapacity() {
351      return this.full_history.length === this.N_history_max;
352    }
353
354    /**
355     * Checks if the history has reached its minimum capacity.
356     */
357    isAtCapacity() {
358      return this.full_history.length === 0;
359    }
360
361    /**
362     * Checks if the history has reached its maximum capacity.
363     */
364    isAtCapacity() {
365      return this.full_history.length === this.N_history_max;
366    }
367
368    /**
369     * Checks if the history has reached its minimum capacity.
370     */
371    isAtCapacity() {
372      return this.full_history.length === 0;
373    }
374
375    /**
376     * Checks if the history has reached its maximum capacity.
377     */
378    isAtCapacity() {
379      return this.full_history.length === this.N_history_max;
380    }
381
382    /**
383     * Checks if the history has reached its minimum capacity.
384     */
385    isAtCapacity() {
386      return this.full_history.length === 0;
387    }
388
389    /**
390     * Checks if the history has reached its maximum capacity.
391     */
392    isAtCapacity() {
393      return this.full_history.length === this.N_history_max;
394    }
395
396    /**
397     * Checks if the history has reached its minimum capacity.
398     */
399    isAtCapacity() {
400      return this.full_history.length === 0;
401    }
402
403    /**
404     * Checks if the history has reached its maximum capacity.
405     */
406    isAtCapacity() {
407      return this.full_history.length === this.N_history_max;
408    }
409
410    /**
411     * Checks if the history has reached its minimum capacity.
412     */
413    isAtCapacity() {
414      return this.full_history.length === 0;
415    }
416
417    /**
418     * Checks if the history has reached its maximum capacity.
419     */
420    isAtCapacity() {
421      return this.full_history.length === this.N_history_max;
422    }
423
424    /**
425     * Checks if the history has reached its minimum capacity.
426     */
427    isAtCapacity() {
428      return this.full_history.length === 0;
429    }
430
431    /**
432     * Checks if the history has reached its maximum capacity.
433     */
434    isAtCapacity() {
435      return this.full_history.length === this.N_history_max;
436    }
437
438    /**
439     * Checks if the history has reached its minimum capacity.
440     */
441    isAtCapacity() {
442      return this.full_history.length === 0;
443    }
444
445    /**
446     * Checks if the history has reached its maximum capacity.
447     */
448    isAtCapacity() {
449      return this.full_history.length === this.N_history_max;
450    }
451
452    /**
453     * Checks if the history has reached its minimum capacity.
454     */
455    isAtCapacity() {
456      return this.full_history.length === 0;
457    }
458
459    /**
460     * Checks if the history has reached its maximum capacity.
461     */
462    isAtCapacity() {
463      return this.full_history.length === this.N_history_max;
464    }
465
466    /**
467     * Checks if the history has reached its minimum capacity.
468     */
469    isAtCapacity() {
470      return this.full_history.length === 0;
471    }
472
473    /**
474     * Checks if the history has reached its maximum capacity.
475     */
476    isAtCapacity() {
477      return this.full_history.length === this.N_history_max;
478    }
479
480    /**
481     * Checks if the history has reached its minimum capacity.
482     */
483    isAtCapacity() {
484      return this.full_history.length === 0;
485    }
486
487    /**
488     * Checks if the history has reached its maximum capacity.
489     */
490    isAtCapacity() {
491      return this.full_history.length === this.N_history_max;
492    }
493
494    /**
495     * Checks if the history has reached its minimum capacity.
496     */
497    isAtCapacity() {
498      return this.full_history.length === 0;
499    }
500
501    /**
502     * Checks if the history has reached its maximum capacity.
503     */
504    isAtCapacity() {
505      return this.full_history.length === this.N_history_max;
506    }
507
508    /**
509     * Checks if the history has reached its minimum capacity.
510     */
511    isAtCapacity() {
512      return this.full_history.length === 0;
513    }
514
515    /**
516     * Checks if the history has reached its maximum capacity.
517     */
518    isAtCapacity() {
519      return this.full_history.length === this.N_history_max;
520    }
521
522    /**
523     * Checks if the history has reached its minimum capacity.
524     */
525    isAtCapacity() {
526      return this.full_history.length === 0;
527    }
528
529    /**
530     * Checks if the history has reached its maximum capacity.
531     */
532    isAtCapacity() {
533      return this.full_history.length === this.N_history_max;
534    }
535
536    /**
537     * Checks if the history has reached its minimum capacity.
538     */
539    isAtCapacity() {
540      return this.full_history.length === 0;
541    }
542
543    /**
544     * Checks if the history has reached its maximum capacity.
545     */
546    isAtCapacity() {
547      return this.full_history.length === this.N_history_max;
548    }
549
550    /**
551     * Checks if the history has reached its minimum capacity.
552     */
553    isAtCapacity() {
554      return this.full_history.length === 0;
555    }
556
557    /**
558     * Checks if the history has reached its maximum capacity.
559     */
560    isAtCapacity() {
561      return this.full_history.length === this.N_history_max;
562    }
563
564    /**
565     * Checks if the history has reached its minimum capacity.
566     */
567    isAtCapacity() {
568      return this.full_history.length === 0;
569    }
570
571    /**
572     * Checks if the history has reached its maximum capacity.
573     */
574    isAtCapacity() {
575      return this.full_history.length === this.N_history_max;
576    }
577
578    /**
579     * Checks if the history has reached its minimum capacity.
580     */
581    isAtCapacity() {
582      return this.full_history.length === 0;
583    }
584
585    /**
586     * Checks if the history has reached its maximum capacity.
587     */
588    isAtCapacity() {
589      return this.full_history.length === this.N_history_max;
590    }
591
592    /**
593     * Checks if the history has reached its minimum capacity.
594     */
595    isAtCapacity() {
596      return this.full_history.length === 0;
597    }
598
599    /**
600     * Checks if the history has reached its maximum capacity.
601     */
602    isAtCapacity() {
603      return this.full_history.length === this.N_history_max;
604    }
605
606    /**
607     * Checks if the history has reached its minimum capacity.
608     */
609    isAtCapacity() {
610      return this.full_history.length === 0;
611    }
612
613    /**
614     * Checks if the history has reached its maximum capacity.
615     */
616    isAtCapacity() {
617      return this.full_history.length === this.N_history_max;
618    }
619
620    /**
621     * Checks if the history has reached its minimum capacity.
622     */
623    isAtCapacity() {
624      return this.full_history.length === 0;
625    }
626
627    /**
628     * Checks if the history has reached its maximum capacity.
629     */
630    isAtCapacity() {
631      return this.full_history.length === this.N_history_max;
632    }
633
634    /**
635     * Checks if the history has reached its minimum capacity.
636     */
637    isAtCapacity() {
638      return this.full_history.length === 0;
639    }
640
641    /**
642     * Checks if the history has reached its maximum capacity.
643     */
644    isAtCapacity() {
645      return this.full_history.length === this.N_history_max;
646    }
647
648    /**
649     * Checks if the history has reached its minimum capacity.
650     */
651    isAtCapacity() {
652      return this.full_history.length === 0;
653    }
654
655    /**
656     * Checks if the history has reached its maximum capacity.
657     */
658    isAtCapacity() {
659      return this.full_history.length === this.N_history_max;
660    }
661
662    /**
663     * Checks if the history has reached its minimum capacity.
664     */
665    isAtCapacity() {
666      return this.full_history.length === 0;
667    }
668
669    /**
670     * Checks if the history has reached its maximum capacity.
671     */
672    isAtCapacity() {
673      return this.full_history.length === this.N_history_max;
674    }
675
676    /**
677     * Checks if the history has reached its minimum capacity.
678     */
679    isAtCapacity() {
680      return this.full_history.length === 0;
681    }
682
683    /**
684     * Checks if the history has reached its maximum capacity.
685     */
686    isAtCapacity() {
687      return this.full_history.length === this.N_history_max;
688    }
689
690    /**
691     * Checks if the history has reached its minimum capacity.
692     */
693    isAtCapacity() {
694      return this.full_history.length === 0;
695    }
696
697    /**
698     * Checks if the history has reached its maximum capacity.
699     */
700    isAtCapacity() {
701      return this.full_history.length === this.N_history_max;
702    }
703
704    /**
705     * Checks if the history has reached its minimum capacity.
706     */
707    isAtCapacity() {
708      return this.full_history.length === 0;
709    }
710
711    /**
712     * Checks if the history has reached its maximum capacity.
713     */
714    isAtCapacity() {
715      return this.full_history.length === this.N_history_max;
716    }
717
718    /**
719     * Checks if the history has reached its minimum capacity.
720     */
721    isAtCapacity() {
722      return this.full_history.length === 0;
723    }
724
725    /**
726     * Checks if the history has reached its maximum capacity.
727     */
728    isAtCapacity() {
729      return this.full_history.length === this.N_history_max;
730    }
731
732    /**
733     * Checks if the history has reached its minimum capacity.
734     */
735    isAtCapacity() {
736      return this.full_history.length === 0;
737    }
738
739    /**
740     * Checks if the history has reached its maximum capacity.
741     */
742    isAtCapacity() {
743      return this.full_history.length === this.N_history_max;
744    }
745
746    /**
747     * Checks if the history has reached its minimum capacity.
748     */
749    isAtCapacity() {
750      return this.full_history.length === 0;
751    }
752
753    /**
754     * Checks if the history has reached its maximum capacity.
755     */
756    isAtCapacity() {
757      return this.full_history.length === this.N_history_max;
758    }
759
760    /**
761     * Checks if the history has reached its minimum capacity.
762     */
763    isAtCapacity() {
764      return this.full_history.length === 0;
765    }
766
767    /**
768     * Checks if the history has reached its maximum capacity.
769     */
770    isAtCapacity() {
771      return this.full_history.length === this.N_history_max;
772    }
773
774    /**
775     * Checks if the history has reached its minimum capacity.
776     */
777    isAtCapacity() {
778      return this.full_history.length === 0;
779    }
780
781    /**
782     * Checks if the history has reached its maximum capacity.
783     */
784    isAtCapacity() {
785      return this.full_history.length === this.N_history_max;
786    }
787
788    /**
789     * Checks if the history has reached its minimum capacity.
790     */
791    isAtCapacity() {
792      return this.full_history.length === 0;
793    }
794
795    /**
796     * Checks if the history has reached its maximum capacity.
797     */
798    isAtCapacity() {
799      return this.full_history.length === this.N_history_max;
800    }
801
802    /**
803     * Checks if the history has reached its minimum capacity.
804     */
805    isAtCapacity() {
806      return this.full_history.length === 0;
807    }
808
809    /**
810     * Checks if the history has reached its maximum capacity.
811     */
812    isAtCapacity() {
813      return this.full_history.length === this.N_history_max;
814    }
815
816    /**
817     * Checks if the history has reached its minimum capacity.
818     */
819    isAtCapacity() {
820      return this.full_history.length === 0;
821    }
822
823    /**
824     * Checks if the history has reached its maximum capacity.
825     */
826    isAtCapacity() {
827      return this.full_history.length === this.N_history_max;
828    }
829
830    /**
831     * Checks if the history has reached its minimum capacity.
832     */
833    isAtCapacity() {
834      return this.full_history.length === 0;
835    }
836
837    /**
838     * Checks if the history has reached its maximum capacity.
839     */
840    isAtCapacity() {
841      return this.full_history.length === this.N_history_max;
842    }
843
844    /**
845     * Checks if the history has reached its minimum capacity.
846     */
847    isAtCapacity() {
848      return this.full_history.length === 0;
849    }
850
851    /**
852     * Checks if the history has reached its maximum capacity.
853     */
854    isAtCapacity() {
855      return this.full_history.length === this.N_history_max;
856    }
857
858    /**
859     * Checks if the history has reached its minimum capacity.
860     */
861    isAtCapacity() {
862      return this.full_history.length === 0;
863    }
864
865    /**
866     * Checks if the history has reached its maximum capacity.
867     */
868    isAtCapacity() {
869      return this.full_history.length === this.N_history_max;
870    }
871
872    /**
873     * Checks if the history has reached its minimum capacity.
874     */
875    isAtCapacity() {
876      return this.full_history.length === 0;
877    }
878
879    /**
880     * Checks if the history has reached its maximum capacity.
881     */
882    isAtCapacity() {
883      return this.full_history.length === this.N_history_max;
884    }
885
886    /**
887     * Checks if the history has reached its minimum capacity.
888     */
889    isAtCapacity() {
890      return this.full_history.length === 0;
891    }
892
893    /**
894     * Checks if the history has reached its maximum capacity.
895     */
896    isAtCapacity() {
897      return this.full_history.length === this.N_history_max;
898    }
899
900    /**
901     * Checks if the history has reached its minimum capacity.
902     */
903    isAtCapacity() {
904      return this.full_history.length === 0;
905    }
906
907    /**
908     * Checks if the history has reached its maximum capacity.
909     */
910    isAtCapacity() {
911      return this.full_history.length === this.N_history_max;
912    }
913
914    /**
915     * Checks if the history has reached its minimum capacity.
916     */
917    isAtCapacity() {
918      return this.full_history.length === 0;
919    }
920
921    /**
922     * Checks if the history has reached its maximum capacity.
923     */
924    isAtCapacity() {
925      return this.full_history.length === this.N_history_max;
926    }
927
928    /**
929     * Checks if the history has reached its minimum capacity.
930     */
931    isAtCapacity() {
932      return this.full_history.length === 0;
933    }
934
935    /**
936     * Checks if the history has reached its maximum capacity.
937     */
938    isAtCapacity() {
939      return this.full_history.length === this.N_history_max;
940    }
941
942    /**
943     * Checks if the history has reached its minimum capacity.
944     */
945    isAtCapacity() {
946      return this.full_history.length === 0;
947    }
948
949    /**
950     * Checks if the history has reached its maximum capacity.
951     */
952    isAtCapacity() {
953      return this.full_history.length === this.N_history_max;
954    }
955
956    /**
957     * Checks if the history has reached its minimum capacity.
958     */
959    isAtCapacity() {
960      return this.full_history.length === 0;
961    }
962
963    /**
964     * Checks if the history has reached its maximum capacity.
965     */
966    isAtCapacity() {
967      return this.full_history.length === this.N_history_max;
968    }
969
970    /**
971     * Checks if the history has reached its minimum capacity.
972     */
973    isAtCapacity() {
974      return this.full_history.length === 0;
975    }
976
977    /**
978     * Checks if the history has reached its maximum capacity.
979     */
980    isAtCapacity() {
981      return this.full_history.length === this.N_history_max;
982    }
983
984    /**
985     * Checks if the history has reached its minimum capacity.
986     */
987    isAtCapacity() {
988      return this.full_history.length === 0;
989    }
990
991    /**
992     * Checks if the history has reached its maximum capacity.
993     */
994    isAtCapacity() {
995      return this.full_history.length === this.N_history_max;
996    }
997
998    /**
999     * Checks if the history has reached its minimum capacity.
1000    */
1001   isAtCapacity() {
1002     return this.full_history.length === 0;
1003   }
1004 }
```

```

160     if(this.history_cont.firstChild) {
161         this.history_cont.removeChild(this.history_cont.firstChild);
162     }
163 }
164 store.set('N_history_max', this.N_history_max);
165 this.saveHistory();
166 }
167 }
168
169 exports.PRDC_JSLAB_COMMAND_HISTORY = PRDC_JSLAB_COMMAND_HISTORY;

```

Listing 71 - command-history.js

```

1  /**
2   * @file JSLAB command window module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8 const { shell } = require('electron');
9 const { BigJsonViewerDom } = require('big-json-viewer');
10 const Store = require('electron-store');
11 const { ESLint } = require("eslint");
12
13 const store = new Store();
14
15 /**
16  * Class for JSLAB command window.
17 */
18 class PRDC_JSLAB_COMMAND_WINDOW {
19
20 /**
21  * Initializes the command window, setting up the UI components, event
22  * listeners, and loading settings from storage.
23  * @param {object} win The window object representing the current Electron
24  * window.
25 */
26 constructor(win) {
27   var obj = this;
28   this.win = win;
29
30   this.terminal_cont = document.getElementById('right-panel');
31
32   this.terminal_history_cont;
33   this.terminal_options_cont;
34   this.messages;
35   this.autoscroll = true;
36   this.show_timestamp = false;
37   this.write_timestamps = true;
38   this.log = [];
39   this.log_dialog;
40   this.settings_dialog;
41   this.N_messages = 1;
42   this.N_messages_max = Infinity;
43   this.code_input;

```

```

42   this.textarea;
43   this.ignore_output = false;
44   this.no_ans = false;
45   this.i_history = -1;
46
47   this.last_class;
48   this.last_tic;
49
50   this.eslint = new ESLint(config.LINT_OPTIONS);
51
52   // Load settings
53   this.autoscroll = store.get('autoscroll');
54   if(!this.autoscroll) {
55     this.autoscroll = true;
56   }
57   this.show_timestamp = store.get('show_timestamp');
58   if(!this.show_timestamp) {
59     this.show_timestamp = false;
60   }
61   this.N_messages_max = Number(store.get('N_messages_max'));
62   if(!isFinite(this.N_messages_max) || this.N_messages_max == 0) {
63     this.N_messages_max = Infinity;
64   }
65   this.write_timestamps = store.get('write_timestamps');
66   if(!this.write_timestamps) {
67     this.write_timestamps = true;
68   }
69
70   // Create terminal DOM
71   this.messages = $('#right-panel .messages');
72
73   // Welcome message
74   var ts = this.getTimestamp();
75   $(this.messages).html('<div class="welcome-message system-in message"><
76     span class="timestamp">' + ts + '</span><
78     div class="clear"></div><p><span>JSLAB</span>, ' + language.string(8) +
79     ' + this.win.app.version + '</p><p>' + language.string(136) + ' ' + new
80     Date().getFullYear() + ' © <span>PR-DC</span> info@pr-dc.com</p><p>' +
81     language.string(137) + '</p><p>' + language.string(138) + '</p><p>' + language
82     .string(139) + '<span>cmd help</span></p><p>' + language.string(135) + ' <a
83     href="https://pr-dc.com/jslab">pr-dc.com/jslab </a></p></div>');
84
85   // Command history
86   this.terminal_history_cont =
87     $('#right-panel .history-cont');
88   $('#right-panel .history-close').click(function() {
89     obj.closeTerminalDialog(obj.terminal_history_cont);
90   });
91   this.terminal_history_cont.on('keydown', function(e) {
92     // https:// keycode.info/
93     function activateCommand() {
94       var el_a = $('#right-panel .history-cont .history-panel li.active');
95       var el = $('#right-panel .history-cont .history-panel li[i="' + obj.
96         i_history + '"]');

```

```
88     el_a.removeClass('active');
89     el.addClass('active');
90     el[0].scrollIntoView({block: 'center', inline: 'center'});
91     obj.code_input.setValue(obj.win.command_history.history[obj.i_history
92     ]);
93     obj.code_input.setCursor(obj.code_input.lineCount(), 0);
94   }
95
96   if(e.key === 'Enter' && !e.shiftKey) {
97     // Enter
98     var el_a = $('#right-panel .history-cont .history-panel li.active');
99     var cmd = el_a.html();
100    obj.win.eval.evalCommand(cmd);
101    obj.closeTerminalDialog(obj.terminal_history_cont);
102    e.stopPropagation();
103    e.preventDefault();
104  } else if(e.key === 'Escape') {
105    // ESC
106    obj.closeTerminalDialog(obj.terminal_history_cont);
107    e.stopPropagation();
108    e.preventDefault();
109  } else if(e.key === 'ArrowUp') {
110    // Arrow up
111    if(obj.i_history > 0) {
112      obj.i_history -= 1;
113      activateCommand();
114    }
115    e.stopPropagation();
116    e.preventDefault();
117  } else if(e.key === 'ArrowDown') {
118    // Arrow down
119    if(obj.i_history < (obj.win.command_history.history.length - 1)) {
120      obj.i_history += 1;
121      activateCommand();
122    }
123    e.stopPropagation();
124    e.preventDefault();
125  } else if(e.key === 'PageUp') {
126    // Page up
127    if(obj.win.command_history.history.length) {
128      obj.i_history = 0;
129      activateCommand();
130    }
131    e.stopPropagation();
132    e.preventDefault();
133  } else if(e.key === 'PageDown') {
134    // Page down
135    if(obj.win.command_history.history.length) {
136      obj.i_history = obj.win.command_history.history.length - 1;
137      activateCommand();
138    }
139    e.stopPropagation();
140    e.preventDefault();
141  });
142});
```

```

142
143 // Create settings dialog
144 this.settings_dialog =
145 $('#right-panel .terminal-settings');
146 this.settings_dialog.on('keydown', function(e) {
147     if(e.key === 'Escape') {
148         // ESC
149         obj.closeTerminalDialog(obj.settings_dialog);
150         e.stopPropagation();
151         e.preventDefault();
152     }
153 });
154 $('#right-panel .terminal-settings .options-close').click(function() {
155     obj.closeTerminalDialog(obj.settings_dialog);
156 });
157 this.setNMessagesMax();
158 $('#right-panel .terminal-settings .change-settings').click(function() {
159     obj.closeTerminalDialog(obj.settings_dialog);
160     obj.N_messages_max =
161         Number($('#right-panel .terminal-settings .N-messages-max').val());
162     obj.setNMessagesMax();
163     if(obj.N_messages > obj.N_messages_max) {
164         while(obj.N_messages > obj.N_messages_max) {
165             obj.messages[0].firstChild.remove();
166             obj.N_messages -= 1;
167         }
168     }
169 });
170
171 // Create log save dialog
172 this.log_dialog = $('#right-panel .terminal-log');
173 this.log_dialog.on('keydown', function(e) {
174     if(e.key === 'Escape') {
175         // ESC
176         obj.closeTerminalDialog(obj.log_dialog);
177         e.stopPropagation();
178         e.preventDefault();
179     }
180 });
181 $('#right-panel .terminal-log .options-close').click(function() {
182     obj.closeTerminalDialog(obj.log_dialog);
183 });
184 $('#right-panel .terminal-log .write-timestamps').click(function() {
185     obj.write_timestamps = this.checked;
186     obj.setWriteTimestamps();
187 });
188 this.setWriteTimestamps();
189 $('#right-panel .terminal-log .save-log').click(function() {
190     obj.closeTerminalDialog(obj.log_dialog);
191     obj.saveLog();
192 });
193
194 // Document keydown callbacks
195 $(document).on('keydown', function(e) {
196     if(e.key.toLowerCase() === 'f' && e.ctrlKey) {

```

```

197   // Ctrl + F
198   obj.scrollToBottom();
199   obj.code_input.focus();
200   obj.code_input.setCursor(obj.code_input.lineCount(), 0);
201   e.stopPropagation();
202   e.preventDefault();
203 }
204 });
205
206 // Create code input
207 this.code_input = CodeMirror.fromTextArea(
208   document.getElementById('command-window-input'), {
209     mode: 'javascript',
210     theme: 'notepadpp',
211     indentUnit: 2,
212     tabSize: 2,
213     lineWrapping: true,
214     matchBrackets: true,
215     gutter: true,
216     gutters: ['CodeMirror-lint-markers'],
217     lint: {
218       getAnnotations: async function(text, callback) {
219         var results = await obj.eslint.lintText(text);
220         callback(results[0].messages.map(message => ({
221           from: CodeMirror.Pos(message.line - 1, message.column - 1),
222           to: CodeMirror.Pos(
223             message.endLine ? message.endLine - 1 : message.line - 1,
224             message.endColumn ? message.endColumn - 1 : message.column
225           ),
226           severity: message.severity === 2 ? "error" : "warning",
227           message: message.message,
228         })));
229       },
230       async: true
231     },
232     highlightSelectionMatches: {annotateScrollbar: true},
233     viewportMargin: Infinity,
234     extraKeys: { Enter: function() {
235       sendCommand();
236     } }
237   });
238
239 $('#command-window-input-submit-cont').click(function() {
240   sendCommand();
241 });
242
243 CodeMirror.keyMap.default['Shift-Tab'] = 'indentLess';
244 CodeMirror.keyMap.default['Tab'] = 'indentMore';
245
246 this.code_input.on('keypress', function(cm, event) {
247   if(!cm.state.completionActive && !event.ctrlKey &&
248     event.key != 'Enter' &&
249     event.key != ';' && event.key != ',' &&
250     event.key != '{' & event.key != '}') {
251     CodeMirror.commands.autocomplete(cm, null,

```

```

252           { completeSingle: false });
253     }
254   });
255
256   // Code input keydown callbacks
257   function historyUp() {
258     if (obj.i_history < (obj.win.command_history.history.length - 1)) {
259       obj.i_history += 1;
260       obj.code_input.setValue(obj.win.command_history.history[obj.i_history]);
261       obj.code_input.setCursor(obj.code_input.lineCount(), 0);
262       obj.scrollToBottom();
263     }
264   }
265
266   function historyDown() {
267     if (obj.i_history > 0) {
268       obj.i_history -= 1;
269       obj.code_input.setValue(obj.win.command_history.history[obj.i_history]);
270       obj.code_input.setCursor(obj.code_input.lineCount(), 0);
271     }
272   }
273
274   function sendCommand() {
275     var cmd = obj.code_input.getValue();
276     obj.win.eval.evalCommand(cmd);
277   }
278
279   this.code_input.on('keydown', function(cm, e) {
280     // https://keycode.info/
281     if (e.key === 'Escape' &&
282         !obj.code_input.state.completionActive) {
283       // ESC
284       obj.code_input.setValue('');
285       obj.resetHistoryIndex();
286       obj.scrollToBottom();
287       e.stopPropagation();
288       e.preventDefault();
289     } else if (e.key === 'ArrowUp' &&
290               !obj.code_input.state.completionActive) {
291       // Arrow up
292
293       var cursor = obj.code_input.getCursor();
294       var line = obj.code_input.lineCount() - 1;
295       var position = obj.code_input.getLine(line).length;
296       if (cursor.line === 0 && (cursor.ch === position || cursor.ch === 0)) {
297         historyUp(e);
298         e.stopPropagation();
299         e.preventDefault();
300       }
301     } else if (e.key === 'ArrowDown' &&
302               !obj.code_input.state.completionActive) {
303       // Arrow down
304

```

```

305     var cursor = obj.code_input.setCursor();
306     var line = obj.code_input.lineCount() -1;
307     var position = obj.code_input.getLine(line).length;
308     if(cursor.line == line && (cursor.ch == position || cursor.ch == 0)) {
309         historyDown(e);
310         e.stopPropagation();
311         e.preventDefault();
312     }
313 } else if(e.key == 'PageUp') {
314 // Page up
315 if(obj.win.command_history.history.length) {
316     obj.i_history = obj.win.command_history.history.length -1;
317     obj.code_input.setValue(obj.win.command_history.history[obj.win.command_history.history.length -1]);
318     obj.code_input.setCursor(obj.code_input.lineCount(), 0);
319     obj.scrollToBottom();
320 }
321 e.stopPropagation();
322 e.preventDefault();
323 } else if(e.key == 'PageDown') {
324 // Page down
325 if(obj.win.command_history.history.length) {
326     obj.i_history = 0;
327     obj.code_input.setValue(obj.win.command_history.history[0]);
328     obj.code_input.setCursor(obj.code_input.lineCount(), 0);
329     obj.scrollToBottom();
330 }
331 e.stopPropagation();
332 e.preventDefault();
333 } else if(e.key == 'F3') {
334 // F3
335 if(obj.win.command_history.history.length) {
336     var new_cmd = obj.win.command_history.history[0];
337     obj.win.evalCommand(new_cmd);
338 }
339 e.stopPropagation();
340 e.preventDefault();
341 } else if(e.key == 'F7' && e.altKey) {
342 // Alt + F7
343     obj.resetHistoryIndex();
344     obj.win.command_history.history = [];
345     e.stopPropagation();
346     e.preventDefault();
347 } else if(e.key == 'F7') {
348 // F7
349     obj.openTerminalDialog(obj.terminal_history_cont);
350     var cmds_cont = $('#right-panel .history-cont .history-panel');
351     cmds_cont.html('');
352     if(obj.win.command_history.history.length) {
353         obj.win.command_history.history.forEach(function(e, i) {
354             if(i == obj.i_history) {
355                 cmds_cont
356                     .append('<li i="' + i + '" class="active">' + e + '</li>');
357             } else {
358                 cmds_cont.append('<li i="' + i + '">' + e + '</li>');
359             }
360         });
361     }
362 }
363 }
364 
```

```

359 }
360 });
361 var el_a = $('#right-panel .history-cont .history-panel li.active');
362 if(el_a.length > 0) {
363   el_a[0].scrollIntoView({block: 'center', inline: 'center'});
364 }
365 var cmdss = cmdss_cont.find('li');
366 cmdss.on('click', function() {
367   cmdss.removeClass('active');
368   $(this).addClass('active');
369   obj.i_history = Number($(this).attr('i'));
370 });
371 } else {
372   cmdss_cont
373     .append('<div class="history-empty">History is empty!</div>');
374 }
375 e.stopPropagation();
376 e.preventDefault();
377 } else if(e.key == 'F8') {
378 // F8
379 var cursor = obj.code_input.getCursor();
380 var line = cursor.line;
381 var position = cursor.ch;
382 var cmd = obj.code_input.getLine(line).substring(0, position);
383 var j = -1;
384 for(var i = 0; i < obj.win.command_history.history.length; i++) {
385   if(obj.win.command_history.history[i].startsWith(cmd)) {
386     if(j > -1) {
387       if(i > obj.i_history) {
388         j = i;
389         break;
390       }
391     } else {
392       j = i;
393       if(obj.i_history == -1) {
394         break;
395       }
396     }
397   }
398 }
399 if(j > -1) {
400   obj.i_history = j;
401   obj.code_input.setValue(obj.win.command_history.history[obj.i_history]);
402   if(obj.win.command_history.history[obj.i_history].length > position)
403     {
404       obj.code_input.focus();
405       obj.code_input.setCursor(cursor);
406     }
407   e.stopPropagation();
408   e.preventDefault();
409 } else if(e.key.toLowerCase() == 's' && e.ctrlKey) {
410 // Ctrl + S
411   obj.openTerminalDialog(obj.settings_dialog);

```

```

412         e.stopPropagation();
413         e.preventDefault();
414     } else if(e.key.toLowerCase() == 'l' && e.ctrlKey) {
415         // Ctrl + L
416         obj.openTerminalDialog(obj.log_dialog);
417         e.stopPropagation();
418         e.preventDefault();
419     }
420 });
421
422 // Focus code input
423 $('#right-panel .terminal-panel').click(function(e){
424     if(e.target != this) return;
425     obj.code_input.focus();
426     obj.code_input.setCursor(obj.code_input.lineCount(), 0);
427 });
428
429 $('#command-window-input-container').click(function(e){
430     if(e.target != this) return;
431     obj.code_input.focus();
432     obj.code_input.setCursor(obj.code_input.lineCount(), 0);
433 });
434
435 // Terminal options cont
436 this.terminal_options_cont = $('#right-panel .options');
437
438 // Terminal settings button click
439 $('#right-panel .options .settings').click(function(){
440     obj.openTerminalDialog(obj.settings_dialog);
441 });
442
443 // Terminal timestamp button click
444 $('#right-panel .options .timestamp').click(function(){
445     if(obj.show_timestamp) {
446         obj.show_timestamp = false;
447     } else {
448         obj.show_timestamp = true;
449     }
450     obj.setTimestamp();
451 });
452 this.setTimestamp();
453
454 // Terminal auto scroll button click
455 $('#right-panel .options .autoscroll').click(function(){
456     if(obj.autoscroll) {
457         obj.autoscroll = false;
458     } else {
459         obj.autoscroll = true;
460     }
461     obj.setAutoscroll();
462 });
463 this.setAutoscroll();
464
465 // Terminal clear button click
466 $('#right-panel .options .clear').click(function(){

```

```

467     obj.clear();
468 });
469
470 // Terminal save log button click
471 $('#right-panel .options .log').click(function(){
472   obj.openTerminalDialog(obj.log_dialog);
473 });
474
475 // Terminal scroll to bottom button click
476 $('#right-panel .options .to-bottom').click(function(){
477   obj.scrollToBottom();
478   obj.code_input.focus();
479   obj.code_input.setCursor(obj.code_input.lineCount(), 0);
480 });
481
482 // Prevent all redirects
483 $(document.body).on('click', '#command-window-messages a', function(e) {
484   if(e.target.href) {
485     e.preventDefault();
486     shell.openExternal(e.target.href);
487     return false;
488   }
489   return true;
490 });
491
492 // Commands for evaluation
493 $(document.body).on('click', '#command-window-messages span.eval-code',
494   function() {
495     obj.win.eval.evalCommand(this.innerText);
496   });
497
498 // Open script in editor
499 $(document.body).on('click', '#command-window-messages span.open-editor',
500   function() {
501     ipcRenderer.send('EditorWindow', 'open-script', [
502       $(this).attr('file_path'),
503       $(this).attr('line_number'),
504       $(this).attr('char_pos')
505     ]);
506   });
507
508 /**
509 * Resets the index used for navigating through command history to its
510 * default state.
511 */
512 resetHistoryIndex() {
513   this.i_history = -1;
514 }
515
516 /**
517 * Applies syntax highlighting to a given snippet of code, returning HTML
518 * markup with syntax highlighting styles applied.
519 * @param {string} data The code snippet to which syntax highlighting should
520 * be applied.
521 * @returns {string} HTML string representing the highlighted code. Special
522 * HTML characters like '<' and '>' are properly escaped.

```

```
514  */
515 highlightCode(data) {
516     return hljs.highlight(data,
517         {language: 'javascript'}).value
518         .replaceAll('&lt;', '<').replaceAll('&gt;', '>'); // Return < and >
519 }
520
521 /**
522 * Clears all messages from the command window.
523 */
524 clear() {
525     this.N_messages = 0;
526     $(this.messages).html('');
527 }
528
529 /**
530 * Displays an error message in the command window.
531 * @param {string} msg The error message to display.
532 */
533 error(msg) {
534     this.win.workspace.updateWorkspace();
535     return this.addMessage('data-in', '<span class="error">' +
536         this.prettyPrint(msg) + '</span>');
537 }
538
539 /**
540 * Displays a warning message in the command window.
541 * @param {string} msg The warning message to display.
542 */
543 warn(msg) {
544     this.win.workspace.updateWorkspace();
545     return this.addMessage('data-in', '<span class="warn">' +
546         this.prettyPrint(msg) + '</span>');
547 }
548
549 /**
550 * Highlights and displays a response message in the command window,
551 * particularly used for 'ans' variable responses.
552 * @param {Array|string} data The data to be highlighted and displayed.
553 */
554 highlightAnsMessage(data) {
555     if(data[1]) {
556         var el = this.message('ans = ', 'ans = '+data[0]);
557         try {
558             var json_data = this.truncateStrings(JSON.parse(data[0]));
559             BigJsonViewerDom.fromObject(json_data).then(function(viewer) {
560                 const node = viewer.getRootElement();
561                 el[0].appendChild(node);
562                 node.openAll(1);
563             }).catch(function(err) {
564                 console.log(err);
565             });
566         } catch(err) {
567             console.log(err);
568         }
569     }
570 }
```

```
568     } else {
569         this.message(this.highlightCode('ans = ' + data[0]), 'ans = ' + data[0])
570         ;
571     }
572 }
573 /**
574 * Displays a general message in the command window.
575 * @param {string} msg The message to display.
576 * @param {string} raw Raw message to log.
577 */
578 message(msg, raw) {
579     this.win.workspace.updateWorkspace();
580     return this.addMessage('data-in', this.prettyPrint(msg), raw);
581 }
582 /**
583 * Displays a general message in the command window with monospaced font.
584 * @param {string} msg The message to display.
585 * @param {string} raw Raw message to log.
586 */
587 messageMonospaced(msg, raw) {
588     this.win.workspace.updateWorkspace();
589     return this.addMessage('data-in', '<div class="monospaced">' + this.
590         prettyPrint(msg) + '</div>', raw);
591 }
592 /**
593 * Displays a general message in the command window.
594 * @param {string} msg The message to display.
595 */
596 messageLatex(msg) {
597     this.win.workspace.updateWorkspace();
598     var el = this.addMessage('data-in', '\\\\(' + msg + '\\\\)');
599     MathJax.typeset(el);
600     return el;
601 }
602 }
603 /**
604 * Displays a message from internal operations in the command window.
605 * @param {string} msg The internal message to display.
606 */
607 messageInternal(msg) {
608     this.addMessage('system-in', this.prettyPrint(msg));
609 }
610 /**
611 * Displays an error message originating from internal operations or system
612 * errors.
613 * @param {string} msg The error message to display.
614 */
615 errorInternal(msg) {
616     this.addMessage('system-in', '<span class="error">' +
617         this.prettyPrint(msg) + '</span>');
618 }
619 }
```

```

620
621  /**
622   * Displays a message related to editor operations in the command window.
623   * @param {string} msg The editor message to display.
624   */
625 messageEditor(msg) {
626   this.addMessage('system-in', '<span class="log">Editor: ' +
627     this.prettyPrint(msg) + '</span>');
628 }
629
630 /**
631  * Opens a specified dialog related to the terminal, like settings or
632  * history.
633  * @param {jQuery} e The jQuery object representing the dialog to open.
634  */
635 openTerminalDialog(e) {
636   if(!e.is(':visible')) {
637     $('.terminal-dialog').fadeOut(300, 'linear');
638     e.fadeIn(300, 'linear', function() {
639       e.focus();
640     });
641   }
642
643 /**
644  * Closes a specified dialog related to the terminal.
645  * @param {jQuery} e The jQuery object representing the dialog to close.
646  */
647 closeTerminalDialog(e) {
648   e.fadeOut(300, 'linear');
649   this.code_input.focus();
650   this.code_input.setCursor(this.code_input.lineCount(), 0);
651 }
652
653 /**
654  * Adds a command as a message to the terminal, applying syntax highlighting
655  * and pretty printing.
656  * @param {string} cmd The command to add to the terminal output.
657  */
658 addMessageCmd(cmd) {
659   var txt = this.prettyPrint(cmd);
660   this.addMessage('data-out', this.highlightCode(txt), txt);
661 }
662
663 /**
664  * Adds a message to the terminal output. Supports merging messages for
665  * continuous output scenarios.
666  * @param {string} msg_class The CSS class to apply to the message, defining
667  *   its type (e.g., 'system-in', 'data-out').
668  * @param {string} data The message content, which can include HTML markup.
669  * @param {boolean} [merge_messages=false] Whether to merge this message
670  *   with the previous one if they are of the same class.
671  * @returns {Object} A jQuery object representing the created message
672  *   element.
673  */

```

```

669 addMessage(msg_class, data, raw, merge_messages = false) {
670   if(typeof raw == 'undefined') {
671     raw = data;
672   }
673   var t = performance.now();
674   var el;
675   if(msg_class != this.last_class) {
676     this.last_class = msg_class;
677     this.last_tic = t;
678     var ts = this.getTimestamp();
679     if(this.N_messages < this.N_messages_max) {
680       this.N_messages += 1;
681     } else {
682       this.messages[0].firstChild.remove();
683     }
684     el = $( '<div class="' + msg_class +
685       '"><span class="timestamp">' +
686       ts + '</span>' + data + '</div>' );
687     $(this).messages.append(el);
688     this.log.push({ 'class': msg_class, 'timestamp': ts, 'data': raw});
689   } else {
690     if(!merge_messages || t - this.last_tic > 1) {
691       this.last_tic = t;
692       var ts = this.getTimestamp();
693       if(this.N_messages < this.N_messages_max) {
694         this.N_messages += 1;
695       } else {
696         this.messages[0].firstChild.remove();
697       }
698       el = $( '<div class="' + msg_class +
699         '"><span class="timestamp">' +
700         ts + '</span>' + data + '</div>' );
701       $(this).messages.append(el);
702       this.log.push({ 'class': msg_class, 'timestamp': ts, 'data': raw});
703     } else {
704       el = $(this.messages).find('div').last();
705       el.append(data);
706       this.log[this.log.length-1].data += raw;
707     }
708   }
709   if(this.autoscroll) {
710     this.scrollToBottom();
711   }
712   return el;
713 }
714
715 /**
716 * Toggles the visibility of timestamps in the command window.
717 */
718 setTimestamp() {
719   var timestamp_button =
720     $( '#right-panel .options .timestamp' );
721   if(this.show_timestamp) {
722     if($(this.messages).hasClass('no-timestamp')) {
723       $(this.messages).removeClass('no-timestamp');

```

```

724     $(timestamp_button).addClass('active');
725     $(timestamp_button).attr('title', language.currentString(41));
726     $(timestamp_button).attr('title-str', 41);
727   }
728 } else {
729   if (!$(this.messages).hasClass('no-timestamp')) {
730     $(this.messages).addClass('no-timestamp');
731     $(timestamp_button).removeClass('active');
732     $(timestamp_button).attr('title', language.currentString(166));
733     $(timestamp_button).attr('title-str', 166);
734   }
735 }
736 store.set('show_timestamp', this.show_timestamp);
737 }

739 /**
740 * Toggles the autoscroll feature of the command window, ensuring the latest
741 * messages are always in view.
742 */
743 setAutoscroll() {
744   var autoscroll_button =
745     $('#right-panel.options.autoscroll');
746   if (this.autoscroll) {
747     if (!$(autoscroll_button).hasClass('active')) {
748       $(autoscroll_button).addClass('active');
749       $(autoscroll_button).attr('title', language.currentString(42));
750       $(autoscroll_button).attr('title-str', 42);
751     }
752   } else {
753     if ($(autoscroll_button).hasClass('active')) {
754       $(autoscroll_button).removeClass('active');
755       $(autoscroll_button).attr('title', language.currentString(167));
756       $(autoscroll_button).attr('title-str', 167);
757     }
758   }
759   store.set('autoscroll', this.autoscroll);
760 }

761 /**
762 * Sets the maximum number of messages to display in the command window
763 * before older messages are removed.
764 */
765 setNMessagesMax() {
766   var N_messages_max_input =
767     $('#right-panel.terminal-settings.N-messages-max');
768   if (this.N_messages_max < 5) {
769     this.N_messages_max = 5;
770   }
771   if ($(N_messages_max_input).val() != this.N_messages_max) {
772     $(N_messages_max_input).val(this.N_messages_max);
773   }
774   store.set('N_messages_max', this.N_messages_max);
775 }

776 /**

```

```

777  * Toggles whether timestamps are written to the log file.
778  */
779 setWriteTimestamps() {
780   var write_timestamps_input =
781     $('#right-panel .terminal-log .write-timestamps')[0];
782   if(this.write_timestamps) {
783     if(!write_timestamps_input.checked) {
784       write_timestamps_input.checked = true;
785     }
786   } else {
787     if(write_timestamps_input.checked) {
788       write_timestamps_input.checked = false;
789     }
790   }
791   store.set('write_timestamps', this.write_timestamps);
792 }
793
794 /**
795  * Saves the current log of the command window to a file.
796  */
797 saveLog() {
798   let options = {
799     title: language.currentString(150),
800     defaultPath: 'jslab_' + this.win.app.getDateTimeFullStr() + '.log',
801     buttonLabel: language.currentString(151),
802     filters: [
803       {name: 'Log', extensions: ['log', 'txt']},
804       {name: 'All Files', extensions: ['*']}
805     ]
806   };
807
808   var obj = this;
809   ipcRenderer.invoke('dialog', 'showSaveDialog', options).then(function(
810     result) {
811     if(!result.canceled) {
812       var data = '';
813       obj.log.forEach(function(x) {
814         data += x.class + ': ';
815         if(obj.write_timestamps) {
816           data += '[' + x.timestamp + '] ';
817         }
818         data += x.data + '\r\n';
819       });
820
821       const fs = require('fs');
822       fs.writeFile(result.filePath, data, function(err) {
823         if(err) {
824           obj.errorInternal(err);
825         }
826       });
827     }).catch(function(err) {
828       obj.errorInternal(err);
829     });
830   }
}

```

```

831
832  /**
833   * Scrolls the command window to the bottom, ensuring the latest messages
834   * are visible.
835  */
836  scrollToBottom() {
837   var bcr = this.messages[0].getBoundingClientRect();
838   $(this.messages).parent()[0].scrollTop = bcr.height;
839 }
840
841 /**
842  * Generates a timestamp for use in the command window.
843  * @returns {string} A string representing the current timestamp.
844  */
845  getTimestamp() {
846   var date = new Date();
847   var pad = function(num, size) {
848     return('000' + num).slice(size * -1);
849   };
850   var time = parseFloat(date.getTime() / 1000).toFixed(3);
851   var hours = date.getHours();
852   var minutes = Math.floor(time / 60) % 60;
853   var seconds = Math.floor(time - minutes * 60);
854   var milliseconds = time.slice(-3);
855
856   return pad(hours, 2) + ':' + pad(minutes, 2) + ':' +
857   pad(seconds, 2) + '.' + pad(milliseconds, 3);
858 }
859 /**
860  * Provides a function to replace circular references while stringifying an
861  * object. Useful for logging objects with circular references.
862  * @returns {Function} A replacer function for JSON.stringify.
863  */
864  getCircularReplacer() {
865   const seen = new WeakSet();
866   return function(key, value) {
867     if(typeof value === 'object' && value !== null) {
868       if(seen.has(value)) {
869         return;
870       }
871       seen.add(value);
872     }
873     return value;
874   };
875 }
876 /**
877  * Formats and pretty-prints data for display in the command window.
878  * @param {object|string} data The data to format.
879  * @returns {string} The formatted data as a string.
880  */
881  prettyPrint(data) {
882   if(typeof data === 'string') {
883     return data.replace(/\n/g, '<br>');

```

```

884     } else if(typeof data === 'object') {
885       if(!Object.keys(data).length){
886         if(data.constructor.name === 'Error') {
887           return data.stack.toString();
888         } else {
889           return data.toString();
890         }
891       } else {
892         return JSON.stringify(data, this.getCircularReplacer(), 2);
893       }
894     } else {
895       return String(data);
896     }
897   }
898
899 /**
900 * Truncates strings in data to config.MAX_JSON_STRING_LENGTH.
901 * @param {any} data - Data to be processed.
902 * @returns {any} Truncated data.
903 */
904 truncateStrings(data) {
905   if(typeof data === 'string') {
906     if(data.length <= config.MAX_JSON_STRING_LENGTH) return data;
907
908     const suffix_prefix = "... [truncated | full size: ";
909     const suffix_suffix = "]";
910
911     let prefix_length = config.MAX_JSON_STRING_LENGTH;
912     while(true) {
913       const full_size_digits = String(data.length).length;
914       const suffix_length = suffix_prefix.length + full_size_digits +
915         suffix_suffix.length;
916       const new_prefix_length = config.MAX_JSON_STRING_LENGTH -
917         suffix_length;
918       if(new_prefix_length >= prefix_length) {
919         prefix_length = new_prefix_length;
920         break;
921       }
922       if(new_prefix_length === prefix_length) break;
923       prefix_length = new_prefix_length;
924     }
925     const suffix = suffix_prefix + data.length + suffix_suffix;
926     return data.slice(0, prefix_length) + suffix;
927   } else if(Array.isArray(data)) {
928     return data.map(item => this.truncateStrings(item));
929   } else if(data !== null && typeof data === 'object') {
930     const new_obj = {};
931     for(const key in data) {
932       if(Object.prototype.hasOwnProperty.call(data, key)) {
933         new_obj[key] = this.truncateStrings(data[key]);
934       }
935     }
936     return new_obj;
937   }
938   return data;
939 }
```

```
937     }
938 }
939
940 exports.PRDC_JSLAB_COMMAND_WINDOW = PRDC_JSLAB_COMMAND_WINDOW;
```

Listing 72 - command-window.js

```
1  /**
2   * @file JSLAB eval module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  const path = require('path');
9
10 /**
11  * Class for JSLAB eval.
12  */
13 class PRDC_JSLAB_EVAL {
14
15 /**
16  * Initializes the script evaluation functionality, setting up the necessary
17  * environment and variables.
18  * @param {object} win The window object representing the current Electron
19  * window.
20  */
21 constructor(win) {
22     var obj = this;
23     this.win = win;
24
25     this.last_script_path;
26     this.jslfilename = 'jslcmdwindow';
27 }
28
29 /**
30  * Evaluates a script from the specified path, optionally focusing on
31  * specific lines. Manages the evaluation process to prevent overlap with
32  * ongoing evaluations.
33  * @param {string} script_path The path to the script file to be evaluated.
34  * @param {Array<number>} [lines] Optional. Specifies the lines of the
35  * script to focus the evaluation on.
36  */
37 evalScript(script_path, lines) {
38     if(!this.win.evaluating) {
39         this.last_script_path = script_path;
40         script_path = script_path.replace(/\//g, '\\\\');
41         var cmd;
42         if(lines !== undefined) {
43             cmd = 'run("' + script_path + ', ' + lines.toString() + ')';
44         } else {
45             cmd = 'run("' + script_path + ')';
46         }
47         this.evalCommand(cmd);
48     } else {
49         this.win.command_window.message('@evalScript: Sandbox is busy...');
```

```

45      }
46  }
47
48 /**
49  * Evaluates a given command, optionally displaying the output in the
50  * application's command window. Manages the command evaluation process to
51  * prevent overlap with ongoing evaluations.
52  * @param {string} cmd The command to be evaluated.
53  * @param {boolean} [show_output=true] Specifies whether the output of the
54  * command should be displayed in the command window.
55  * @param {string} [jsl_file_name='jslcmdwindow'] Specifies the file name
56  * context for the command evaluation.
57  */
58 evalCommand(cmd, show_output = true, jsl_file_name = 'jslcmdwindow') {
59   if(!this.win.evaluating) {
60     if(cmd.length) {
61       this.win.command_window.addMessageCmd(cmd);
62       this.win.command_history.updateHistory(cmd);
63       this.win.command_window.code_input.setValue('');
64       this.win.command_window.scrollToBottom();
65       this.win.command_window.resetHistoryIndex();
66       this.evalCode(cmd, show_output, jsl_file_name);
67     }
68   } else {
69     this.win.command_window.message('@evalCommand: Sandbox is busy... ');
70   }
71 }
72
73 /**
74  * Directly evaluates code, interacting with the sandbox environment for
75  * execution. Ensures that only one evaluation is happening at any time.
76  * @param {string} code The code snippet to be evaluated.
77  * @param {boolean} [show_output=true] Specifies whether the output of the
78  * code evaluation should be shown.
79  * @param {string} [jsl_file_name='jslcmdwindow'] The context file name for
80  * the code evaluation.
81  */
82 evalCode(code, show_output = true, jsl_file_name = 'jslcmdwindow') {
83   if(!this.win.evaluating) {
84     this.win.evaluating = true;
85     ipcRenderer.send('SandboxWindow', 'eval-code',
86       [code, show_output, jsl_file_name]);
87   } else {
88     this.win.command_window.message('@evalCode: Sandbox is busy... ');
89   }
90 }
91
92 /**
93  * Handles actions for the script directory dialog buttons, including
94  * changing the active directory, saving the directory to paths, and
95  * running the last script.
96  * @param {number} s The button state indicating the action to be performed.
97  */
98 scriptDirDialogButton(s) {
99   this.win.gui.closeDialog($('#script-path-container'));

```

```

91   var script_dir = path.dirname(this.last_script_path);
92   if(s == 2) {
93     // Change active directory
94     this.win.folder_navigation.setPath(script_dir);
95   } else if(s == 1) {
96     // Add directory to saved paths
97     this.win.folder_navigation.savePath(script_dir);
98   }
99
100  ipcRenderer.send('SandboxWindow', 'run-last-script');
101 }
102
103 }
104
105 exports.PRDC_JSLAB_EVAL = PRDC_JSLAB_EVAL;

```

Listing 73 - eval.js

```

1 /**
2  * @file JSLAB file browser module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const fs = require('fs');
9 const path = require('path');
10
11 /**
12  * Class for JSLAB file browser.
13  */
14 class PRDC_JSLAB_FILE_BROWSER {
15
16 /**
17  * Initializes the file browser, setting up the UI component and event
18  * listeners for file browser interactions.
19  * @param {object} win The window object representing the current Electron
20  * window.
21  */
22 constructor(win) {
23   var obj = this;
24   this.win = win;
25
26   this.file_browser_cont = document.getElementById('file-browser-cont');
27
28   // File browser refresh button click
29   $('#file-browser-options .refresh').click(function(e){
30     obj.updateFileBrowser();
31   });
32
33 /**
34  * Displays the contents of a specified folder within the file browser UI,
35  * optionally clearing the current display and replacing it with the new
36  * content.
37  * @param {string} folder_path The path to the folder whose contents should

```

```

35      be displayed .
36  * @param {HTMLElement} [element=this.file_browser_cont] The HTML element
37      where the folder contents should be displayed. Defaults to the file
38      browser container .
39  * @param {boolean} [root=false] Indicates whether the folder is the root
40      folder being displayed. If true , the browser will clear its current
41      content .
42  */
43 showFolderContent(folder_path , element = this.file_browser_cont , root =
44     false) {
45     var obj = this ;
46     var ul = document.createElement('ul') ;
47     ul.setAttribute('path' , folder_path.replace(/\\/g , '/')) ;
48     if(root) {
49         $(element).html('') ;
50     } else {
51         $(element).find('ul').remove() ;
52     }
53     fs.readdir(folder_path , {withFileTypes: true} , function(err , dirents) {
54         if(err) {
55             obj.win.command_window.errorInternal(language.string(92) + ': ' + err)
56             ;
57             return ;
58         }
59         dirents.sort((a, b) => {
60             if(a.isDirectory() && !b.isDirectory()) return -1;
61             if(!a.isDirectory() && b.isDirectory()) return 1;
62             return a.name.localeCompare(b.name) ;
63         });
64         dirents.forEach(function(dirent) {
65             var absolute_path = path.join(folder_path , dirent.name) ;
66             obj.addFileBrowserItem(absolute_path , dirent , ul) ;
67         });
68         $(ul).appendTo(element).hide().slideDown(300 , 'linear') ;
69     });
70 }
71 /**
72  * Adds an item to the file browser UI, such as a file or folder , including
73  * its name and an icon indicating its type .
74  * @param {string} absolute_path The absolute path to the file or folder to
75  * add .
76  * @param {HTMLElement} ul The unordered list (UL) HTML element to which the
77  * item should be added .
78  */
79 addFileBrowserItem(absolute_path , dirent , ul) {
80     var obj = this ;
81     var type_folder = false ;
82     var li = document.createElement('li') ;
83     li.setAttribute('path' , absolute_path.replace(/\\/g , '/')) ;
84     li.innerHTML = '<span>' + path.basename(absolute_path) + '</span>' ;

```

```

80      if(dirent.isDirectory()) {
81          li.className = 'folder';
82          type_folder = true;
83      } else if(dirent.isSymbolicLink()) {
84          li.className = 'link';
85          absolute_path = fs.readlinkSync(absolute_path);
86          type_folder = true;
87      } else {
88          li.className = 'file';
89          li.onclick = function(e) {
90              e.stopPropagation();
91              e.preventDefault();
92              ipcRenderer.send('EditorWindow', 'open-script', [absolute_path]);
93          };
94          var ext = absolute_path.split('.').pop();
95          if(ext === 'jsl') {
96              li.classList.add('jsl');
97          } else if(ext === 'js') {
98              li.classList.add('js');
99          } else if(ext === 'json') {
100              li.classList.add('json');
101          }
102      }
103      if(type_folder) {
104          li.onclick = function(e) {
105              e.stopPropagation();
106              e.preventDefault();
107              obj.win.folder_navigation.setPath(absolute_path);
108          };
109          var expand = document.createElement('i');
110          expand.className = 'expand';
111          expand.onclick = function(e) {
112              e.stopPropagation();
113              e.preventDefault();
114              if($(this).hasClass('expanded')) {
115                  $(this).removeClass('expanded');
116                  var parent_ul = $(this).parent().find('ul');
117                  $(parent_ul).slideUp(300, 'linear', function() {
118                      $(parent_ul).remove();
119                  });
120              } else {
121                  $(this).addClass('expanded');
122                  obj.showFolderContent(absolute_path, $(this).parent());
123              }
124          };
125          li.appendChild(expand);
126      }
127      ul.appendChild(li);
128  }
129
130 /**
131 * Refreshes the file browser to reflect the current state of the filesystem
132 * or the contents of the current directory.
133 */
updateFileBrowser() {

```

```

134     this.win.folder_navigation.setPath(this.win.folder_navigation.current_path
135         );
136     }
137 }
138
139 exports.PRDC_JSLAB_FILE_BROWSER = PRDC_JSLAB_FILE_BROWSER;

```

Listing 74 - file-browser.js

```

1  /**
2   * @file JSLAB folder navigation module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8 const fs = require('fs');
9 const path = require('path');
10 const { pathEqual } = require('path-equal');
11 const Store = require('electron-store');
12
13 const store = new Store();
14
15 /**
16  * Class for JSLAB folder navigation.
17  */
18 class PRDC_JSLAB_FOLDER_NAVIGATION {
19
20 /**
21  * Initializes folder navigation, setting up UI components and event
22  * listeners for folder navigation actions.
23  * @param {object} win The window object representing the current Electron
24  * window.
25  */
26 constructor(win) {
27     var obj = this;
28     this.win = win;
29
30     this.current_path = undefined;
31     this.saved_paths = [];
32     this.path_history = [];
33     this.i_path_history = 0;
34
35     // Folder navigation
36     $('#folder-navigation-container .address-line').blur(function() {
37         obj.onPathInput(this);
38     });
39     $('#folder-navigation-container .address-line').on('keydown', function(e)
40         {
41             if(e.key === 'Enter' && !e.shiftKey) {
42                 // Enter
43                 e.stopPropagation();

```

```

43         e.preventDefault();
44         obj.onPathInput(this);
45     }
46 });
47 $('#folder-navigation-container .address-line').focus(function() {
48     this.setSelectionRange(0, this.value.length);
49 });
50 $('#folder-navigation-container .open-folder').click(function() {
51     let options = {
52         title: language.currentString(148),
53         defaultPath: obj.current_path,
54         buttonLabel: language.currentString(149),
55         properties: [ 'openDirectory' ]
56     };
57     ipcRenderer.invoke('dialog', 'showOpenDialog', options).then(function(
58         result) {
59         if (!result.canceled) {
60             obj.setPath(result.filePaths[0]);
61         }
62     }).catch(function(err) {
63         obj.win.command_window.errorInternal(err);
64     });
65 });
66 $('#folder-navigation-container .up-folder').click(function() {
67     var folders = obj.current_path.split(path.sep);
68     folders = folders.filter(function(el) { return el != ''; });
69     folders.pop();
70     if(folders.length == 1) {
71         obj.setPath(folders);
72     } else {
73         obj.setPath(path.join(...folders));
74     }
75 });
76 $('#folder-navigation-container .previous-folder').click(function() {
77     obj.setPath(obj.current_path, obj.i_path_history+1);
78 });
79 $('#folder-navigation-container .next-folder').click(function() {
80     obj.setPath(obj.current_path, obj.i_path_history-1);
81 });
82 // Paths logic
83 $('#paths-menu').click(function() {
84     obj.updatePathsList();
85     obj.win.gui.openPathsMenu();
86 });
87 $('#paths-close').click(function() {
88     obj.win.gui.closePathsMenu();
89 });
90 $('#paths-container').on('keydown', function(e) {
91     if(e.key == 'Escape') {
92         // ESC
93         obj.win.gui.closePathsMenu();
94         e.stopPropagation();
95         e.preventDefault();
96     }

```

```

97     });
98     $('#save-path').click(function() {
99       obj.toggleSavePath();
100    });
101
102   // Script path logic
103   $('#script-path-close').click(function() {
104     obj.win.gui.closeDialog($('#script-path-container'));
105   });
106   $('#script-path-container').on('keydown', function(e) {
107     if(e.key === 'Escape') {
108       // ESC
109       obj.win.gui.closeDialog($('#script-path-container'));
110       e.stopPropagation();
111       e.preventDefault();
112     }
113   });
114
115  // Saved paths
116  this.saved_paths = store.get('saved_paths');
117  if(!this.saved_paths) {
118    this.saved_paths = [];
119  }
120  ipcRenderer.send('SandboxWindow', 'set-saved-paths', this.saved_paths);
121
122  // Set path
123  var current_path = store.get('current_path');
124  if(!current_path || !(fs.existsSync(current_path) &&
125    fs.lstatSync(current_path, {throwIfNoEntry: false}).isDirectory())) {
126    current_path = this.win.app.documents_path;
127  }
128  this.setPath(current_path);
129}
130
131 /**
132  * Processes the input from the address line, navigating to the specified
133  * path if it is different from the current path.
134  * @param {HTMLElement} e The HTML input element containing the path.
135  */
136 onPathInput(e) {
137   var new_path = $(e).val();
138   new_path = this.addPathSep(new_path);
139   if(!pathEqual(new_path, this.current_path)) {
140     $(e).val(new_path);
141     this.setPath(new_path);
142   }
143 }
144 /**
145  * Sets the current path for navigation, updating the UI and internal state
146  * accordingly. Supports navigation through history via index.
147  * @param {string} new_path The new path to set as the current directory.
148  * @param {number} [i=undefined] Optional index for navigation through the
149  * path history.
150  * @param {boolean} [inform_sandbox=true] Whether to inform the sandbox

```

```

    process of the path change.

149  */
150 setPath(new_path, i = undefined, inform_sandbox = true) {
151   new_path = this.addPathSep(new_path);
152   if(this.checkDirectory(new_path)) {
153     if (!pathEqual(new_path, this.current_path) && i === undefined) {
154       this.path_history.unshift(new_path);
155       if(this.path_history.length > 1) {
156         \$('#folder-navigation-container .previous-folder').removeClass('disabled');
157         \$('#folder-navigation-container .next-folder').addClass('disabled');
158         this.i_path_history = 0;
159       }
160     } else if(i !== undefined && i >= 0 &&
161           i < this.path_history.length) {
162       this.i_path_history = i;
163       new_path = this.path_history[i];
164       if(i == 0) {
165         \$('#folder-navigation-container .next-folder').addClass('disabled');
166       } else {
167         \$('#folder-navigation-container .next-folder').removeClass('disabled');
168       }
169     if(i == (this.path_history.length - 1)) {
170       \$('#folder-navigation-container .previous-folder').addClass('disabled');
171     } else {
172       \$('#folder-navigation-container .previous-folder').removeClass('disabled');
173     }
174   }
175
176   this.current_path = new_path;
177   if(inform_sandbox) {
178     ipcRenderer.send('SandboxWindow', 'set-current-path', new_path);
179   }
180   if(this.saved_paths.indexOf(this.current_path) >= 0) {
181     \$('#save-path').addClass('saved');
182   } else {
183     \$('#save-path').removeClass('saved');
184   }
185   this.win.file_browser.showFolderContent(this.current_path, undefined,
186                                         true);
186 } else {
187   this.setPath(this.win.app.documents_path);
188 }
189 this.showCurrentPath();
190 }

191 /**
192 * Saves the current path to the list of saved paths for quick access.
193 * @param {string} new_path The path to save.
194 * @param {boolean} [inform_sandbox=true] Whether to inform the sandbox
195   process of the update.
196 */

```

```

197 savePath(new_path, inform_sandbox = true) {
198     new_path = this.addPathSep(new_path);
199     var i = this.saved_paths.indexOf(new_path);
200     if(i < 0) {
201         this.saved_paths.push(new_path);
202     }
203     if(inform_sandbox) {
204         ipcRenderer.send('SandboxWindow', 'set-saved-paths', this.saved_paths);
205     }
206 }
207 /**
208 * Removes a path from the list of saved paths.
209 * @param {string} saved_path The path to remove.
210 * @param {boolean} [inform_sandbox=true] Whether to inform the sandbox
211   process of the update.
212 */
213 removePath(saved_path, inform_sandbox = true) {
214     saved_path = this.addPathSep(saved_path);
215     var i = this.saved_paths.indexOf(saved_path);
216     if(i >= 0) {
217         this.saved_paths.splice(i, 1);
218     }
219     if(inform_sandbox) {
220         ipcRenderer.send('SandboxWindow', 'set-saved-paths', this.saved_paths);
221     }
222 }
223 /**
224 * Toggles the current path between being saved and not saved, updating the
225   UI and stored paths accordingly.
226 */
227 toggleSavePath() {
228     var i = this.saved_paths.indexOf(this.current_path);
229     if(i >= 0) {
230         this.removePath(this.current_path);
231         $('#save-path').removeClass('saved');
232     } else {
233         this.savePath(this.current_path);
234         $('#save-path').addClass('saved');
235     }
236     this.updatePathsList();
237 }
238 /**
239 * Updates the UI to display the current path in the address line and
240   navigation breadcrumbs.
241 */
242 showCurrentPath() {
243     var obj = this;
244     $('#folder-navigation-container .address-line').val(this.current_path);
245     var folders = this.current_path.split(path.sep);
246     folders = folders.filter(function(el) { return el != ''; });
247     var address = $('#folder-navigation-container .current-address')[0];
248     address.innerHTML = '';

```

```

249  var full_path = folders[0];
250  if(folders.length == 1) {
251    $('#folder-navigation-container .up-folder').addClass('disabled');
252  } else {
253    $('#folder-navigation-container .up-folder').removeClass('disabled');
254  }
255  for(var i = 0; i < folders.length; i++) {
256    if(i > 0) {
257      full_path += path.sep;
258      full_path += folders[i];
259    }
260    if(folders[i] != '') {
261      var span = document.createElement('span');
262      span.className = 'folder';
263      span.title = full_path;
264      if(i == 0) {
265        span.title += path.sep;
266      }
267      span.textContent = folders[i];
268      span.onclick = function() {
269        obj.setPath(this.title);
270      };
271      address.appendChild(span);
272      if(i != (folders.length-1)) {
273        $(address).append('<i class="i-next-folder"></i>');
274      }
275    }
276  }
277 }
278 /**
279 * Updates the list of saved paths in the UI, allowing users to quickly
280 * navigate to frequently accessed directories.
281 */
282 updatePathsList() {
283  var obj = this;
284  var cont = $('#paths-container .page-panel ul');
285  if(this.saved_paths.length > 0) {
286    $(cont).html('');
287    this.saved_paths.forEach(function(saved_path) {
288      var row = document.createElement('li');
289      row.textContent = saved_path;
290      row.onclick = function() {
291        obj.win.gui.closePathsMenu();
292        obj.setPath(saved_path);
293      };
294      if(!obj.checkDirectory(saved_path)) {
295        row.className = 'inactive';
296      }
297      var btn = document.createElement('img');
298      btn.src = '../img/close.svg';
299      btn.className = 'remove-path';
300      btn.onclick = function(event) {
301        event.preventDefault();
302        event.stopPropagation();

```

```

303     if(pathEqual(saved_path, obj.current_path)) {
304         $('#save-path').removeClass('saved');
305     }
306     obj.removePath(saved_path);
307     $(this).parent().remove();
308     if(obj.saved_paths.length == 0) {
309         $(cont).html('<li class="no-paths">' + language.string(140) + '</li>');
310         ;
311     }
312     row.appendChild(btn);
313     $(cont).append(row);
314 });
315 } else {
316     $(cont).html('<li class="no-paths">' + language.string(140) + '</li>');
317 }
318 }

319 /**
320 * Handles UI and state updates when a script directory is unknown,
321 * prompting the user for action.
322 */
323 unknownScriptDir() {
324     var script_dir = this.addPathSep(path.dirname(this.win.eval(
325         last_script_path));
326     $('#script-path').text(script_dir);
327     this.win.gui.openDialog($('#script-path-container'));
328     return true;
329 }

330 /**
331 * Appends a path separator to the end of a path string if it is not already
332 * present.
333 * @param {string} path_str The path string to modify.
334 * @return {string} The modified path string with a trailing separator.
335 */
336 addPathSep(path_str) {
337     if(path_str && path_str[path_str.length - 1] != path.sep) {
338         path_str += path.sep;
339     }
340     return path_str;
341 }

342 /**
343 * Checks if the specified directory exists and is a directory.
344 * @param {string} directory The path to check.
345 * @return {boolean} True if the directory exists and is a directory, false
346 * otherwise.
347 */
348 checkDirectory(directory) {
349     var lstat = fs.lstatSync(directory, {throwIfNoEntry: false});
350     if(lstat != undefined && lstat.isDirectory()) {
351         return true;
352     } else {
353         return false;
354     }
355 }

```

```
353     }
354 }
355
356 /**
357 * Checks if the specified file exists and is a file.
358 * @param {string} file_path The path to the file to check.
359 * @return {boolean} True if the file exists and is a file, false otherwise.
360 */
361 checkFile(file_path) {
362     var lstat = fs.lstatSync(file_path, {throwIfNoEntry: false});
363     if(lstat != undefined && lstat.isFile()) {
364         return true;
365     } else {
366         return false;
367     }
368 }
369 }
370
371 exports.PRDC_JSLAB_FOLDER_NAVIGATION = PRDC_JSLAB_FOLDER_NAVIGATION;
```

Listing 75 - folder-navigation.js

```
1  /**
2   * @file JSLAB GUI module
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB GUI.
10  */
11 class PRDC_JSLAB_GUI {
12
13  /**
14   * Initializes the GUI, setting up event listeners for window controls, menu
15   * actions, and dialog interactions.
16   * @param {object} win The window object representing the current Electron
17   * window.
18   */
19  constructor(win) {
20    var obj = this;
21    this.win = win;
22
23    this.state = 'ready';
24    this.stats = {};
25    this.stats_num = 0;
26    this.fullscreen = false;
27
28    this.last_focus = document.activeElement;
29    this.status_cont = document.getElementById('status');
30    this.sandbox_stats_icon = document.getElementById('sandbox-stats-icon');
31
32    document.addEventListener("keydown", function(e) {
33      if(e.key === 'F11') {
34        obj.toggleFullscreen();
35      }
36    });
37  }
38}
```

```

33     }
34 );
35
36 // On devtools-menu click
37 $('#devtools-menu').click(function() {
38   ipcRenderer.send('MainProcess', 'show-dev-tools');
39 });
40
41 // On editor-menu click
42 $('#editor-menu').click(function() {
43   ipcRenderer.send('MainProcess', 'show-editor');
44 });
45
46 $('#script-path-dialog-change-dir').click(function() { obj.win.eval(
47   scriptDirDialogButton(2); });
48 $('#script-path-dialog-save').click(function() { obj.win.eval(
49   scriptDirDialogButton(1); });
50 $('#script-path-dialog-run').click(function() { obj.win.eval(
51   scriptDirDialogButton(0); });

52 // Window controls
53 window.addEventListener('resize', function() {
54   // Detect change of maximize
55   obj.maximized = ipcRenderer.sendSync('sync-message', 'is-maximized-win')
56   ;
57   if(obj.maximized) {
58     $('#win-restore img').attr('src', '../img/win-restore.svg');
59   } else {
60     $('#win-restore img').attr('src', '../img/win-maximize.svg');
61   }
62   , true);
63
64 $('#win-close').click(function() {
65   obj.win.close();
66 });
67
68 $('#win-restore').click(function() {
69   obj.toggleFullscreen(false);
70   obj.maximized = !obj.maximized;
71   if(obj.maximized) {
72     ipcRenderer.send('MainProcess', 'maximize-win');
73   } else {
74     ipcRenderer.send('MainProcess', 'restore-win');
75   }
76 });
77
78 $('#win-minimize').click(function() {
79   obj.toggleFullscreen(false);
80   ipcRenderer.send('MainProcess', 'minimize-win');
81 });
82
83   window.dispatchEvent(new Event('resize'));
84 }

85 /**
86 * Invoked when the GUI is ready, performing initial UI setup tasks such as

```

```

      fading in the window.

84  /*
85   onReady() {
86     // Fade in window
87     ipcRenderer.send('MainProcess', 'fade-in-win');
88   }

89
90 /**
91 * Toggles the fullscreen state of the application window.
92 * @param {boolean} [fullscreen] Optional. Specifies the fullscreen state.
93 * If not provided, the state is toggled based on the current state.
94 */
95 toggleFullscreen(fullscreen) {
96   if(fullscreen == null) {
97     fullscreen = !this.fullscreen;
98   }
99   if(fullscreen) {
100     ipcRenderer.send('MainProcess', 'set-fullscreen', true);
101   } else {
102     ipcRenderer.send('MainProcess', 'set-fullscreen', false);
103   }
104   this.fullscreen = fullscreen;
105 }

106 /**
107 * Updates the status displayed in the status bar of the application.
108 * @param {string} state The current state to display.
109 * @param {string} txt The text to display in the status bar.
110 */
111 setStatus(state, txt) {
112   this.state = state;
113   $(this.status_cont).html(txt);
114   this.setStatsIcon();
115 }

116 /**
117 * Resets the stats data to initial values.
118 */
119 resetStats() {
120   this.setStatus('ready', language.string(87));
121   var stats = {
122     'required_modules': 0,
123     'promises': 0,
124     'timeouts': 0,
125     'immediates': 0,
126     'intervals': 0,
127     'animation_frames': 0,
128     'idle_callbacks': 0
129   };
130   this.setStats(stats);
131 }
132

133 /**
134 * Updates the statistics displayed in the GUI, such as the number of active
135 * promises, timeouts, and intervals.

```

```

136     * @param {object} stats An object containing statistical information to
137     * display.
138     */
139     setStats(stats) {
140         this.stats = stats;
141         $('#sandbox-required-modules').text(stats['required_modules']);
142         $('#sandbox-promises-num').text(stats['promises']);
143         $('#sandbox-timeouts-num').text(stats['timeouts']);
144         $('#sandbox-immediates-num').text(stats['immediates']);
145         $('#sandbox-intervals-num').text(stats['intervals']);
146         $('#sandbox-animation-frames-num').text(stats['animation_frames']);
147         $('#sandbox-idle-callbacks-num').text(stats['idle_callbacks']);
148         this.stats_num = stats['promises']+stats['timeouts']+stats['immediates']+  

149             stats['intervals']+stats['animation_frames']+stats['idle_callbacks'];
150         this.setStatsIcon();
151     }
152
153     /**
154      * Updates the visibility and appearance of the statistics icon based on the
155      * current application state and stats.
156      */
157     setStatsIcon() {
158         this.sandbox_stats_icon.className = '';
159         if(this.state == 'ready') {
160             if(this.stats_num > 0) {
161                 this.sandbox_stats_icon.classList.add('async-busy');
162             } else {
163                 this.sandbox_stats_icon.classList.add('ready');
164             }
165         } else {
166             this.sandbox_stats_icon.classList.add('busy');
167         }
168     }
169
170     /**
171      * Opens a specified dialog within the application.
172      * @param {jQuery} e The jQuery object representing the dialog to open.
173      */
174     openDialog(e) {
175         if(!e.is(':visible')) {
176             this.last_focus = document.activeElement;
177             $('.main-dialog').fadeOut(300, 'linear');
178             e.fadeIn(300, 'linear', function() {
179                 e.focus();
180             });
181         }
182     }
183
184     /**
185      * Closes a specified dialog within the application.
186      * @param {jQuery} e The jQuery object representing the dialog to close.
187      */
188     closeDialog(e) {
189         e.fadeOut(300, 'linear');
190         $(this.last_focus).focus();

```

```
189 }
190
191 /**
192 * Opens the paths menu dialog, providing quick access to saved and
193 * frequently used paths.
194 */
195 openPathsMenu() {
196     this.openDialog($('#paths-container'));
197 }
198 /**
199 * Closes the paths menu dialog.
200 */
201 closePathsMenu() {
202     this.closeDialog($('#paths-container'));
203 }
204
205 /**
206 * Opens the help dialog, providing access to application documentation or
207 * assistance.
208 */
209 help() {
210     this.openDialog($('#help-container'));
211 }
212 /**
213 * Closes the help dialog.
214 */
215 closeHelp() {
216     this.closeDialog($('#help-container'));
217 }
218
219 /**
220 * Opens the application info dialog, displaying information about the
221 * application.
222 */
223 info() {
224     this.openDialog($('#info-container'));
225 }
226 /**
227 * Closes the application info dialog.
228 */
229 closeInfo() {
230     this.closeDialog($('#info-container'));
231 }
232
233 /**
234 * Opens the settings dialog, allowing the user to configure application
235 * settings.
236 */
237 settings() {
238     this.openDialog($('#settings-container'));
239 }
```

```

240  /**
241   * Closes the settings dialog.
242   */
243  closeSettings() {
244   this.closeDialog($('#settings-container'));
245 }
246
247 /**
248  * Changes the application language to the specified language ID.
249  * @param {number} id The ID of the language to switch to.
250  */
251 changeLangauge(id) {
252   language.set(id);
253   ipcRenderer.send('EditorWindow', 'set-language', id);
254   ipcRenderer.send('SandboxWindow', 'set-language', id);
255 }
256 }
257
258 exports.PRDC_JSLAB_GUI = PRDC_JSLAB_GUI;

```

Listing 76 - gui.js

```

1 /**
2  * @file JSLAB help module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB help.
10 */
11 class PRDC_JSLAB_HELP {
12
13 /**
14  * Initializes the help functionality, setting up event listeners for help
15  * menu actions and dialog interactions.
16  * @param {object} win The window object representing the current Electron
17  * window.
18  */
19 constructor(win) {
20   var obj = this;
21   this.win = win;
22
23   // Help logic
24   $('#help-menu').click(function() { obj.win.gui.help(); });
25
26   $('#help-close').click(function() { obj.win.gui.closeHelp(); });
27
28   $('#help-container').on('keydown', function(e) {
29     if(e.key === 'Escape') {
30       // ESC
31       obj.win.gui.closeHelp();
32       e.stopPropagation();
33       e.preventDefault();
34     }
35   });

```

```

33     });
34 }
35
36 }
37
38 exports.PRDC_JSLAB_HELP = PRDC_JSLAB_HELP

```

Listing 77 - help.js

```

1 /**
2  * @file JSLAB info module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB info .
10 */
11 class PRDC_JSLAB_INFO {
12
13 /**
14  * Initializes the information display functionality , setting up event
15  * listeners for information menu actions and dialog interactions .
16  * @param {object} win The window object representing the current Electron
17  * window .
18 */
19 constructor(win) {
20   var obj = this ;
21   this.win = win ;
22
23   // Info logic
24   $('#info-container .app-version').text('version ' + this.win.app.version);
25   $('#info-menu').click(function() { obj.win.gui.info(); });
26   $('#info-close').click(function() { obj.win.gui.closeInfo(); });
27   $('#info-container').on('keydown', function(e) {
28     if(e.key == 'Escape') {
29       // ESC
30       obj.win.gui.closeInfo();
31       e.stopPropagation();
32       e.preventDefault();
33     }
34   });
35 }
36
37 exports.PRDC_JSLAB_INFO = PRDC_JSLAB_INFO

```

Listing 78 - info.js

```

1 /**
2  * @file JSLAB main window init file
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com

```

```

6   */
7
8 // Modules
9 // -----
10 const { ipcRenderer } = require('electron');
11
12
13 const helper = require("../js/helper.js");
14 require("../js/init-config.js");
15 const { PRDC_APP_LOGGER } = require('../lib/PRDC_APP_LOGGER/PRDC_APP_LOGGER');
16 const { PRDC_JSLAB_LANGUAGE } = require('../js/language');
17
18 global.app_path = process.argv.find(e => e.startsWith('--app-path=')).split('=
  ')[1].replace(/\js\?$/,'');
19
20 const { PRDC_JSLAB_WIN_MAIN } = require('../js/main/win-main');
21
22 // Start log
23 const log_file = ipcRenderer.sendSync('sync-message', 'get-log-file');
24 const app_logger = new PRDC_APP_LOGGER(log_file);
25
26 // Global variables
27 var language = new PRDC_JSLAB_LANGUAGE();
28 var win_main = new PRDC_JSLAB_WIN_MAIN();
29
30 // When document is ready
31 // -----
32 ready(function() {
33   // Jquery ready
34   $(document).ready(function() {
35     win_main.onReady();
36   });
37 });

```

Listing 79 - init-main.js

```

1 /**
2  * @file JSLAB panels module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const { PRDC_PANEL } = require('../lib/PRDC_PANEL/PRDC_PANEL');
9
10 /**
11  * Class for JSLAB panels.
12  */
13 class PRDC_JSLAB_PANELS {
14
15 /**
16  * Initializes main application panels and configures their default sizes
17  * and orientations.
18  * @param {object} win The window object representing the current Electron
19  * window.
20 */

```

```

19  constructor(win) {
20    var obj = this;
21    this.win = win;
22
23    this.columns = new PRDC_PANEL('columns', 'vertical', document.
24      getElementById('panels-container'), [document.getElementById('left-
25      panel'), document.getElementById('right-panel')], config.
26      PANEL_DEFAULT_COLUMNS);
27
28    this.left_rows = new PRDC_PANEL('left-rows', 'horizontal', document.
29      getElementById('left-panel'), [document.getElementById('left-top-panel')
30      ], document.getElementById('left-middle-panel'), document.
31      getElementById('left-bottom-panel')], config.PANEL_DEFAULT_LEFT_ROWS);
32
33    this.workspace_columns = new PRDC_PANEL('workspace-columns', 'vertical',
34      document.getElementById('workspace'), ['#left-middle-panel.col-1',
35      '#left-middle-panel.col-2', '#left-middle-panel.col-3'], config.
36      PANEL_DEFAULT_WORKSPACE_COLUMNS);
37
38    // Initialize panels
39    window.addEventListener('resize', function() {
40      obj.columns.onResize();
41    });
42
43
44 /**
45 * Invoked when the application window is ready, triggering an initial
46 * resize event to ensure panels are correctly laid out.
47 */
48 onReady() {
49   this.columns.onResize();
50 }
51
52
53 exports.PRDC_JSLAB_PANELS = PRDC_JSLAB_PANELS;

```

Listing 80 - panels.js

```

1 /**
2  * @file JSLAB settings module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB settings.
10 */
11 class PRDC_JSLAB_SETTINGS {
12
13 /**
14  * Initializes the settings management functionality, setting up event
15  * listeners for settings menu actions and dialog interactions.

```

```

15  * @param {object} win The window object representing the current Electron
16  * window.
17  */
18  constructor(win) {
19   var obj = this;
20   this.win = win;
21
22   // Settings logic
23   $('#settings-menu').click(function() { obj.win.gui.settings(); });
24   $('#settings-close').click(function() { obj.win.gui.closeSettings(); });
25   $('#settings-container').on('keydown', function(e) {
26     if(e.key === 'Escape') {
27       // ESC
28       obj.win.gui.closeSettings();
29       e.stopPropagation();
30       e.preventDefault();
31     }
32   });
33   $('#settings-container .set-langauge').on('change', function() {
34     obj.win.gui.changeLangauge($('#this').val());
35   });
36   $('#settings-container .set-langauge').val(language.lang);
37
38   $('#settings-container .change-settings').on('click', function() {
39     obj.win.command_history.setMaxSize($('#settings-container .N-history-max')
40       .val());
41     obj.win.gui.closeSettings();
42   });
43 }
44
45 exports.PRDC_JSLAB_SETTINGS = PRDC_JSLAB_SETTINGS

```

Listing 81 - settings.js

```

1 /**
2  * @file JSLAB GUI script
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 // Modules
9 // -----
10 const { ipcRenderer } = require('electron');
11
12 const { PRDC_JSLAB_HELP } = require('../help');
13 const { PRDC_JSLAB_INFO } = require('../info');
14 const { PRDC_JSLAB_SETTINGS } = require('../settings');
15 const { PRDC_JSLAB_EVAL } = require('../eval');
16 const { PRDC_JSLAB_COMMAND_WINDOW } = require('../command-window');
17 const { PRDC_JSLAB_COMMAND_HISTORY } = require('../command-history');
18 const { PRDC_JSLAB_WORKSPACE } = require('../workspace');
19 const { PRDC_JSLAB_FOLDER_NAVIGATION } = require('../folder-navigation');
20 const { PRDC_JSLAB_FILE_BROWSER } = require('../file-browser');

```

```

21 const { PRDC_JSLAB_PANELS } = require('./panels');
22 const { PRDC_JSLAB_GUI } = require('./gui');
23 const { PRDC_JSLAB_APP } = require('./app');
24
25 const { PRDC_POPUP } = require('../lib/PRDC_POPUP/PRDC_POPUP');
26
27 const fs = require('fs');
28 const path = require('path');
29 const Store = require('electron-store');
30
31 const store = new Store();
32
33 /**
34 * Class for JSLAB main win.
35 */
36 class PRDC_JSLAB_WIN_MAIN {
37
38 /**
39 * Create main win.
40 */
41 constructor() {
42   var obj = this;
43
44 // Classes
45 this.eval = new PRDC_JSLAB_EVAL(this);
46 this.workspace = new PRDC_JSLAB_WORKSPACE(this);
47 this.file_browser = new PRDC_JSLAB_FILE_BROWSER(this);
48 this.panels = new PRDC_JSLAB_PANELS(this);
49 this.gui = new PRDC_JSLAB_GUI(this);
50 this.help = new PRDC_JSLAB_HELP(this);
51 this.app = new PRDC_JSLAB_APP(this);
52 this.command_history = new PRDC_JSLAB_COMMAND_HISTORY(this);
53 this.command_window = new PRDC_JSLAB_COMMAND_WINDOW(this);
54 this.info = new PRDC_JSLAB_INFO(this);
55 this.settings = new PRDC_JSLAB_SETTINGS(this);
56 this.folder_navigation = new PRDC_JSLAB_FOLDER_NAVIGATION(this);
57
58 this.flight_commands_popup = new PRDC_POPUP(document.getElementById(
      'sandbox-stats-icon'),
      document.getElementById('sandbox-stats-popup'));
59
60 // Prevent redirects
61 preventRedirect();
62
63 // Events
64 // -----
65 // Catch errors
66 window.addEventListener('unhandledrejection', function(e) {
67   obj.command_window.errorInternal(e.reason.stack);
68   e.preventDefault();
69 });
70
71 window.addEventListener('error', function(e) {
72   obj.command_window.errorInternal(e.error.stack);
73   e.preventDefault();
74 });

```

```
75 });
76
77 // On IPC message
78 ipcRenderer.on('MainWindow', function(event, action, data) {
79   switch(action) {
80     case 'editor-disp':
81       var msg = data[0];
82       var focus = data[1];
83       obj.command_window.messageEditor(msg);
84       if(focus) {
85         ipcRenderer.send('MainProcess', 'focus-win');
86       }
87       break;
88     case 'disp':
89       obj.command_window.message(data);
90       break;
91     case 'disp-monospaced':
92       obj.command_window.messageMonospaced(data);
93       break;
94     case 'disp-latex':
95       obj.command_window.messageLatex(data);
96       break;
97     case 'error':
98       obj.command_window.error(data);
99       ipcRenderer.send('MainProcess', 'focus-win');
100      break;
101    case 'warn':
102      obj.command_window.warn(data);
103      ipcRenderer.send('MainProcess', 'focus-win');
104      break;
105    case 'internal-error':
106      obj.command_window.errorInternal(data);
107      ipcRenderer.send('MainProcess', 'focus-win');
108      break;
109    case 'run':
110      var script_path = data[0];
111      var lines = data[1];
112      obj.eval.evalScript(script_path, lines);
113      ipcRenderer.send('MainProcess', 'focus-win');
114      break;
115    case 'eval-command':
116      obj.eval.evalCommand(data[0]);
117      ipcRenderer.send('MainProcess', 'focus-win');
118      break;
119    case 'help':
120      obj.gui.help();
121      break;
122    case 'info':
123      obj.gui.info();
124      break;
125    case 'settings':
126      obj.gui.settings();
127      break;
128    case 'clear':
129      obj.command_window.clear();
```

```

130      break;
131  case 'save-path':
132    var new_path = data;
133    obj.folder_navigation.savePath(new_path, false);
134    break;
135  case 'remove-path':
136    var saved_path = data;
137    obj.folder_navigation.removePath(saved_path, false);
138    break;
139  case 'set-workspace':
140    obj.workspace.setWorkspace(data);
141    break;
142  case 'update-workspace':
143    obj.workspace.updateWorkspace();
144    break;
145  case 'update-file-browser':
146    obj.file_browser.updateFileBrowser();
147    break;
148  case 'code-evaluating':
149    obj.evaluating = true;
150    break;
151  case 'code-evaluated':
152    obj.evaluating = false;
153    break;
154  case 'show-ans':
155    obj.command_window.highlightAnsMessage(data);
156    break;
157  case 'set-status':
158    var state = data[0];
159    var txt = data[1];
160    obj.gui.setStatus(state, txt);
161    break;
162  case 'set-stats':
163    var stats = data;
164    obj.gui.setStats(stats);
165    break;
166  case 'clear-storage':
167    store.clear();
168    break;
169  case 'unknown-script-dir':
170    obj.folder_navigation.unknownScriptDir();
171    break;
172  case 'set-current-path':
173    obj.folder_navigation.setPath(data);
174    break;
175  }
176 });
177
178 // Keyboard events
179 document.addEventListener('keydown', function(e) {
180   // Show Dev Tools
181   if(e.key === 'F12') {
182     ipcRenderer.send('MainProcess', 'show-dev-tools');
183     ipcRenderer.send('MainProcess', 'show-sandbox-dev-tools');
184     e.stopPropagation();

```

```

185     e.preventDefault();
186 } else if(e.ctrlKey && e.key.toLowerCase() == 'c') {
187   if(obj.getSelectionText() == "") {
188     // No selected text
189     obj.command_window.messageInternal(language.string(89));
190     ipcRenderer.send('SandboxWindow', 'stop-loop', true);
191     e.stopPropagation();
192     e.preventDefault();
193   }
194 } else if(e.key.toLowerCase() == 'h' && e.ctrlKey) {
195   // Ctrl + H
196   obj.gui.help();
197   e.stopPropagation();
198   e.preventDefault();
199 } else if(e.key.toLowerCase() == 'i' && e.ctrlKey) {
200   // Ctrl + I
201   obj.gui.info();
202   e.stopPropagation();
203   e.preventDefault();
204 } else if(e.key.toLowerCase() == 's' && e.ctrlKey) {
205   // Ctrl + S
206   obj.gui.settings();
207   e.stopPropagation();
208   e.preventDefault();
209 } else if(e.key.toLowerCase() == 'd' && e.ctrlKey) {
210   // Ctrl + D
211   obj.eval.evalCommand('openDoc()');
212   e.stopPropagation();
213   e.preventDefault();
214 }
215 );
216 }
217
218 /**
219 * Method called when the program is ready to start.
220 */
221 onReady() {
222   var obj = this;
223   obj.processArguments();
224   obj.panels.onReady();
225
226   // Fade in window
227   ipcRenderer.send('MainProcess', 'fade-in-win');
228
229   // Focus code input
230   obj.command_window.code_input.focus();
231   obj.command_window.code_input.setCursor(
232     obj.command_window.code_input.lineCount(), 0);
233 }
234
235 /**
236 * Processes startup arguments, opening files or setting the workspace as
237 * needed.
238 */
processArguments() {

```

```

239 // Process arguments
240 if(process.argv.length > 0) {
241   var arg = process.argv[1];
242
243   // Check if there is scripts in argument
244   if(arg && this.folder_navigation.checkFile(arg)) {
245     // Open script in editor
246     ipcRenderer.send('EditorWindow', 'open-script', [arg]);
247     var dir = path.dirname(arg);
248     if(dir !== undefined) {
249       this.folder_navigation.setPath(dir);
250     }
251   }
252 }
253
254 /**
255 * Retrieves selected text within the application, if any.
256 * @return {string} The currently selected text.
257 */
258 getSelectionText() {
259   var text = "";
260   if(window.getSelection) {
261     text = window.getSelection().toString();
262   } else if(document.selection && document.selection.type != "Control") {
263     text = document.selection.createRange().text;
264   }
265   return text;
266 }
267
268 /**
269 * Resets the sandbox and updates paths and settings.
270 */
271 onSandboxReset() {
272   ipcRenderer.send('SandboxWindow', 'set-language', language.lang);
273   ipcRenderer.send('SandboxWindow', 'set-saved-paths', this.folder_navigation.saved_paths);
274   ipcRenderer.send('SandboxWindow', 'set-current-path', this.folder_navigation.current_path);
275   this.command_window.clear();
276   this.workspace.updateWorkspace();
277   this.gui.resetStats();
278   this.evaluating = false;
279 }
280
281 /**
282 * Gracefully closes the application after performing necessary cleanup
283 * operations.
284 */
285 close() {
286   store.set('full_history', this.command_history.full_history);
287   store.set('current_path', this.folder_navigation.current_path);
288   store.set('saved_paths', this.folder_navigation.saved_paths);
289   ipcRenderer.send('MainProcess', 'close-app');
290 }

```

```

291 }
292
293 exports.PRDC_JSLAB_WIN_MAIN = PRDC_JSLAB_WIN_MAIN;

```

Listing 82 - win-main.js

```

1 /**
2  * @file JSLAB workspace module
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB workspace.
10 */
11 class PRDC_JSLAB_WORKSPACE {
12
13 /**
14  * Initializes the workspace, setting up the UI component and event
15  * listeners for workspace interactions.
16  * @param {object} win The window object representing the current Electron
17  * window.
18 */
19 constructor(win) {
20   var obj = this;
21   this.win = win;
22
23   this.data = [];
24   this.workspace_cont = document.getElementById('workspace');
25
26   // Workspace clear button click
27   $('#workspace-options .clear').click(function(){
28     obj.win.eval.evalCommand('clear');
29   });
30 }
31 /**
32  * Updates the workspace with new data, refreshing the display of variables
33  * and their values.
34  * @param {Array} data The data to be displayed in the workspace, typically
35  * an array of variable information.
36 */
37 setWorkspace(data) {
38   this.data = data;
39   this.updateWorkspace();
40 }
41 /**
42  * Refreshes the workspace display based on the current data, updating the
43  * layout and content of the workspace area.
44 */
45 updateWorkspace() {
46   var obj = this;
47
48   var workspace_table = $('#workspace .table')[0];

```

```

46   workspace_table.innerHTML = '';
47   var cells_size = this.win.panels.workspace_columns.cells_size;
48   this.data.forEach(function(v) {
49     var row = document.createElement('div');
50     row.onclick = function() {
51       obj.win.command_window.code_input.setValue(this.getAttribute('variable'));
52     };
53     obj.win.command_window.code_input.focus();
54     obj.win.command_window.code_input.setCursor(obj.win.command_window.
55       code_input.lineCount(), 0);
56   };
57   row.ondblclick = function() {
58     obj.win.eval.evalCommand(this.getAttribute('variable'));
59   };
60   row.setAttribute('variable', v[0]);
61   row.className = 'row';
62   row.innerHTML = '<div class="col col-1" style="width:' + cells_size[0] +
63     '%">' + v[0] +
64     '</div><div class="col col-2" style="width:' + cells_size[1] + '%">' + v
65     [1] +
66     '</div><div class="col col-3" style="width:' + cells_size[2] + '%">' + v
67     [2] + '</div>';
68   workspace_table.appendChild(row);
69 });
70 }
71 }
72
73 exports.PRDC_JSLAB_WORKSPACE = PRDC_JSLAB_WORKSPACE;

```

Listing 83 - workspace.js

6.5 sandbox

```

1 /**
2  * @file JSLAB library array submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB array submodule.
10 * Column-major order for matrix operation
11 */
12 class PRDC_JSLAB_LIB_ARRAY {
13
14 /**
15  * Constructs an array submodule object for JSLAB.
16  * @param {Object} js1 - Reference to the main JSLAB object.
17 */
18 constructor(js1) {
19   var obj = this;
20   this.js1 = js1;
21 }

```

```
22
23  /**
24   * Converts an iterable or array-like object into a standard array.
25   * @param {Iterable|ArrayLike} A - The iterable or array-like object to
26   * convert.
27   * @returns {Array} A new array containing the elements from the input.
28   */
29 array(A) {
30   return Array.from(A);
31 }
32 /**
33  * Retrieves the last element from the provided array.
34  * @param {Array} A - The array from which the last element is to be
35  * retrieved.
36  * @returns {*} The last element of the array. If the array is empty,
37  * returns undefined.
38 */
39 end(A) {
40   return A[A.length - 1];
41 }
42 /**
43  * Returns the index of the last element in the array.
44  * @param {Array} A - The array to evaluate.
45  * @returns {number} The index of the last element.
46 */
47 endi(A) {
48   return A.length - 1;
49 }
50 /**
51  * Retrieves a specific column from the provided matrix.
52  * @param {Array} A - The matrix array.
53  * @param {number} col - The column index to retrieve.
54  * @returns {Array} The specified column as an array.
55 */
56 column(A, col) {
57   return A.map(function(x) { return x[col]; });
58 }
59 /**
60  * Retrieves a specific row from the provided matrix.
61  * @param {Array} A - The matrix array.
62  * @param {number} row - The row index to retrieve.
63  * @returns {Array} The specified row as an array.
64 */
65 row(A, row) {
66   return A[row];
67 }
68 /**
69  * Returns the first N elements from an array.
70  * @param {Array} A - The input array.
71  * @param {number} [N=1] - The number of elements to return from the start.
```

```

74   * @returns {Array} - An array containing the first N elements.
75   */
76 first(A, N = 1) {
77   return A.slice(0, N);
78 }
79
80 /**
81  * Returns the last N elements from an array.
82  * @param {Array} A - The input array.
83  * @param {number} [N=1] - The number of elements to return from the end.
84  * @returns {Array} - An array containing the last N elements.
85 */
86 last(A, N = 1) {
87   return A.slice(-N);
88 }
89
90 /**
91  * Generates an array of indices based on rows and columns.
92  * @param {number|number[]} rows - The row index or array of row indices.
93  * @param {number|number[]} cols - The column index or array of column
94  *     indices.
95  * @param {number} rows_max - The maximum number of rows.
96  * @returns {number[]} An array of calculated indices.
97 */
98 index(rows, cols, rows_max) {
99   if (!Array.isArray(rows)) {
100     rows = [rows];
101   }
102   if (!Array.isArray(cols)) {
103     cols = [cols];
104   }
105   var A = zeros(rows.length * cols.length);
106
107   var k = 0;
108   for (var j = 0; j < cols.length; j++) {
109     for (var i = 0; i < rows.length; i++) {
110       A[k++] = rows[i] + cols[j] * rows_max;
111     }
112   }
113   return A;
114 }
115
116 /**
117  * Finds all indices of a specified value in the array.
118  * @param {Array} A - The array to search.
119  * @param {*} value - The value to find.
120  * @returns {number[]} An array of indices where the value is found.
121 */
122 indexOfAll(A, value) {
123   return A.reduce(function(a, e, i) {
124     if (e === value) {
125       a.push(i);
126     }
127   return a;
128 }

```

```
128     }, []));
129 }
130
131 /**
132 * Returns the index of a value in an array.
133 * @param {Array} A - Array to search.
134 * @param {*} value - Value to locate.
135 * @returns {number} Index of value or -1.
136 */
137 indexOf(A, value) {
138     return A.indexOf(value);
139 }
140
141 /**
142 * Finds the index of a sequence of elements in the array.
143 * @param {Array} A - The array to search.
144 * @param {Array} search_elements - The sequence of elements to find.
145 * @param {number} [from_index=0] - The index to start the search from.
146 * @returns {number} The starting index of the found sequence, or -1 if not
147 * found.
148 */
149 indexOfMulti(A, search_elements, from_index) {
150     from_index = from_index || 0;
151
152     var index = Array.prototype.indexOf.call(A, search_elements[0],
153         from_index);
154     if(search_elements.length === 1 || index === -1) {
155         // Not found or no other elements to check
156         return index;
157     }
158
159     for(var i = index, j = 0;
160         j < search_elements.length && i < A.length; i++, j++) {
161         if(A[i] !== search_elements[j]) {
162             return indexOfMulti(A, search_elements, index + 1);
163         }
164     }
165
166     return (i === index + search_elements.length) ? index : -1;
167 }
168
169 /**
170 * Shuffles indices of an array using the Fisher-Yates algorithm.
171 * @param {Array} array Array whose indices are to be shuffled.
172 * @return {Array} Shuffled array of indices.
173 */
174 shuffleIndices(array) {
175     const idx = Array.from(array.keys());
176     for(let i = idx.length - 1; i > 0; i--) {
177         const j = Math.floor(Math.random() * (i + 1));
178         [idx[i], idx[j]] = [idx[j], idx[i]];
179     }
180     return idx;
181 }
```

```

182 /**
183 * Sets a subset of array elements based on provided indices.
184 * @param {Array} A - The target array.
185 * @param {number[]} indices - The indices at which to set values.
186 * @param {Array} B - The array of values to set.
187 */
188 setSub(A, indices, B) {
189     var j = 0;
190     if(!Array.isArray(B)) {
191         B = createFilledArray(indices.length, B);
192     }
193     for(var i = 0; i < indices.length; i++) {
194         A[indices[i]] = B[j++];
195     }
196 }
197 /**
198 * Sets a subset of array elements based on provided boolean indices.
199 * @param {Array} A - The target array to modify.
200 * @param {boolean[]} b - A boolean array indicating which elements to set (true = set, false = skip).
201 * @param {Array} B - The array of values to set at the indices determined by 'b'.
202 */
203 setSubB(A, b, B) {
204     this.setSub(A, b2i(b), B);
205 }
206 /**
207 * Retrieves a subset of array elements based on provided indices.
208 * @param {Array} A - The source array.
209 * @param {number[]} indices - The indices of elements to retrieve.
210 * @returns {Array} An array containing the retrieved elements.
211 */
212 getSub(A, indices) {
213     var B = zeros(indices.length);
214     var j = 0;
215     for(var i = 0; i < indices.length; i++) {
216         B[j++] = A[indices[i]];
217     }
218     return B;
219 }
220 /**
221 * Retrieves a subset of array elements based on provided indices.
222 * @param {Array} A - The source array.
223 * @param {boolean[]} b - A boolean array indicating which elements to retrieve.
224 * @returns {Array} An array containing the retrieved elements.
225 */
226 getSubB(A, b) {
227     return this.getSub(A, b2i(b));
228 }
229 /**
230 * @param {Array} A - The target array.
231 * @param {number[]} indices - The indices at which to set values.
232 * @param {Array} B - The array of values to set.
233 */

```

```

234  * Moves an element within the array from one index to another.
235  * @param {Array} A - The array to modify.
236  * @param {number} from_index - The index of the element to move.
237  * @param {number} to_index - The target index where the element should be
238  * moved.
239  */
240 moveElement(A, from_index, to_index) {
241   const start_index = from_index < 0 ? A.length + from_index : from_index;
242
243   if(start_index >= 0 && start_index < A.length) {
244     const end_index = to_index < 0 ? A.length + to_index : to_index;
245
246     const [item] = A.splice(from_index, 1);
247     A.splice(end_index, 0, item);
248   }
249   return A;
250 }
251 /**
252  * Removes an element from the array at the specified index.
253  * @param {Array} A - The array to modify.
254  * @param {number} index - The index of the element to remove.
255  * @returns {Array} The array after the element has been removed.
256  */
257 removeElement(A, index) {
258   return A.splice(index, 1);
259 }
260 /**
261  * Removes the first occurrence of a specified value from the array.
262  * @param {Array} A - The array to modify.
263  * @param {*} value - The value to remove.
264  * @returns {Array} The array after the value has been removed.
265  */
266 removeElementByValue(A, value) {
267   var index = A.indexOf(value);
268   if(index !== -1) {
269     A.splice(index, 1);
270   }
271   return A;
272 }
273 /**
274  * Removes elements from array A that have properties listed in array B.
275  * @param {Array<Object>} A - The array to filter.
276  * @param {Array} B - The array of properties or values to remove.
277  * @param {string} [prop] - The property name to check in objects within A.
278  * @returns {Array} The filtered array.
279  */
280 removeElementProp(A, B, prop) {
281   if(prop) {
282     return A.filter(function(e) { return !B.includes(e[prop]); });
283   } else {
284     return A.filter(function(e) { return !B.includes(e); });
285   }
286 }
287

```

```
288     }
289 }
290
291 /**
292 * Finds the index of the first object in the array where the specified
293 * property matches the given value.
294 * @param {Array<Object>} A - The array to search.
295 * @param {string} property - The property name to compare.
296 * @param {*} value - The value to match.
297 * @returns {number} The index of the matching object, or -1 if not found.
298 */
299 findIndexProp(A, property, value) {
300     return A.findIndex(function(object) {
301         return object[property] === value;
302     });
303
304 /**
305 * Sets a value at the specified multi-dimensional indices in the array.
306 * @param {Array} A - The target array.
307 * @param {number[]} indices - An array of indices representing the position
308 * @param {*} value - The value to set.
309 * @returns {*} The value that was set.
310 */
311 setValueAt(A, indices, value) {
312     var reference = A;
313     for(var i = 0; i < indices.length - 1; i++) {
314         reference = reference[indices[i]];
315     }
316     reference[indices[indices.length - 1]] = value;
317     return reference;
318 }
319
320 /**
321 * Retrieves a value from the array at the specified multi-dimensional
322 * indices.
323 * @param {Array} A - The source array.
324 * @param {number[]} indices - An array of indices representing the position
325 * @returns {*} The value at the specified indices, or undefined if out of
326 * bounds.
327 */
328 getValueAt(A, indices) {
329     if(Array.isArray(A)) {
330         var reference = A;
331         for(var i = 0; i < indices.length; i++) {
332             if(indices[i] < reference.length) {
333                 reference = reference[indices[i]];
334             } else {
335                 return undefined;
336             }
337         }
338         return reference;
339     } else {
340 
```



```
338     return undefined;
339 }
340 }
341 /**
342 * Alias for setValueAt.
343 * @param {Array} A - The target array.
344 * @param {number[]} indices - An array of indices representing the position
345 *
346 * @param {*} value - The value to set.
347 * @returns {*} The value that was set.
348 */
349 setVal(A, indices, value) {
350     return this.setValueAt(A, indices, value);
351 }
352 /**
353 * Alias for getValueAt.
354 * @param {Array} A - The source array.
355 * @param {number[]} indices - An array of indices representing the position
356 *
357 * @returns {*} The value at the specified indices, or undefined if out of
358 * bounds.
359 */
360 getVal(A, indices) {
361     return this.getValueAt(A, indices);
362 }
363 /**
364 * Returns the intersection of two arrays.
365 * @param {Array} A - First array.
366 * @param {Array} B - Second array.
367 * @returns {Array} Common elements.
368 */
369 arrayIntersect(A, B) {
370     if (A.length > B.length) [A, B] = [B, A];
371     const set_B = new Set(B);
372     return [...A.filter(item => set_B.has(item))];
373 }
374 /**
375 * Creates a shallow copy of the provided array.
376 * @param {Array} A - The array to clone.
377 * @returns {Array} A new array containing all elements from A.
378 */
379 cloneArray(A) {
380     return structuredClone(A);
381 }
382 /**
383 * Creates an n-dimensional array.
384 * @param {number} length - The size of the first dimension.
385 * @param {...number} [dimensions] - Sizes of subsequent dimensions.
386 * @returns {Array} The newly created n-dimensional array.
387 */
388
```

```
390  createArray(length) {
391      var A = new Array(length || 0);
392      var i = length;
393
394      if(arguments.length > 1) {
395          var args = Array.prototype.slice.call(arguments, 1);
396          while(i--) A[(length-1)-i] = createArray.apply(this, args);
397      }
398
399      return A;
400  }
401
402  /**
403   * Creates an n-dimensional array filled with a specific value.
404   * @param {number} length - The size of the first dimension.
405   * @param {*} val - The value to fill the array with.
406   * @param {...number} [dimensions] - Sizes of subsequent dimensions.
407   * @returns {Array} The filled n-dimensional array.
408   */
409  createFilledArray(length, val) {
410      var A = new Array(length || 0);
411      var i = length;
412
413      if(arguments.length > 2) {
414          var args = Array.prototype.slice.call(arguments, 1);
415          while(i--) A[(length-1)-i] = this.createFilledArray.apply(this, args);
416      } else {
417          A.fill(val);
418      }
419
420      return A;
421  }
422
423  /**
424   * Fills the array with a specific value.
425   * @param {*} val - The value to fill the array with.
426   * @param {Array} A - The array to fill.
427   * @param {number} length - The number of elements to fill.
428   */
429  fill(val, A, length) {
430      for(var i = 0; i < length; i++) {
431          A[i] = val;
432      }
433  }
434
435  /**
436   * Creates an n-dimensional array filled with NaN.
437   * @param {...number} size - The size of each dimension.
438   * @returns {Array} The NaN-filled n-dimensional array.
439   */
440  NaNs(...size) {
441      return this.createFilledArray(...size, NaN);
442  }
443
444  /**
```

```

445  * Creates an n-dimensional array filled with zeros.
446  * @param {...number} size - The size of each dimension.
447  * @returns {Array} The zero-filled n-dimensional array.
448  */
449 zeros(...size) {
450   return this.createFilledArray(...size, 0);
451 }
452
453 /**
454  * Creates an n-dimensional array filled with ones.
455  * @param {...number} size - The size of each dimension.
456  * @returns {Array} The one-filled n-dimensional array.
457  */
458 ones(...size) {
459   return this.createFilledArray(...size, 1);
460 }
461
462 /**
463  * Creates a 2-dimensional identity matrix.
464  * @param {number} size - The size of the identity matrix.
465  * @returns {Array} The identity matrix as a 2D array.
466  */
467 eye(size) {
468   return this.diag(this.ones(size), size);
469 }
470
471 /**
472  * Scales an array by a scalar.
473  * @param {Array<number>} A - The array to scale.
474  * @param {number} s - The scalar value.
475  * @returns {Array<number>} The scaled array.
476  */
477 scale(A, s) {
478   var obj = this;
479   return A.map(function(x) {
480     if(Array.isArray(x)) {
481       return obj.scale(x, s);
482     } else {
483       return x * s;
484     }
485   });
486 }
487
488 /**
489  * Creates a linearly spaced vector.
490  * @param {number} x1 - The start value.
491  * @param {number} x2 - The end value.
492  * @param {number} [N=100] - The number of points.
493  * @returns {Array<number>} The linearly spaced vector.
494  */
495 linspace(x1, x2, rows = 100) {
496   var A = new Array(rows);
497   var dx = (x2-x1)/(rows-1);
498   for(var i = 0; i < rows; i++) {
499     A[i] = x1+dx*i;

```

```

500     }
501     return A;
502   }
503
504 /**
505 * Generates an array of numbers within a specified range.
506 * @param {...*} args - The start, end, and optional step for the range.
507 * @returns {Array<number>} An array containing numbers within the specified
508   range.
509 */
510 range (... args) {
511   return [... this.jsl.env.math.range (... args, true).toArray ()];
512 }
513 /**
514 * Generates a sequence of numbers from 'x1' to 'x2' with increments of 'dx'
515   .
516 * @param {number} x1 - The starting value of the sequence.
517 * @param {number} x2 - The ending value of the sequence.
518 * @param {number} dx - The increment between values in the sequence.
519 * @returns {number[]} An array of numbers from 'x1' to 'x2' with step size
520   'dx'.
521 */
522 colon (x1, x2, dx) {
523   if (dx === 0) {
524     this.jsl.env.error ('@colon: '+language.string (189));
525   }
526   const x = [];
527   const tolerance = 1e-14; // Tolerance for floating-point comparisons
528   let n = 0;
529   let xi = x1;
530   const increasing = dx > 0;
531
532   while (
533     (increasing && xi <= x2 + tolerance) ||
534     (!increasing && xi >= x2 - tolerance)
535   ) {
536     x.push (xi);
537     xi = x1 + dx * ++n;
538   }
539
540   return x;
541 }
542 /**
543 * Applies a function to corresponding elements of one or more arrays.
544 * @param {Function} func - The function to apply to the elements.
545 * @param {...Array} arrays - One or more arrays to process.
546 * @returns {Array} A new array with the function applied to each
547   corresponding element.
548 */
549 elementWise (func, ... arrays) {
550   // Ensure all arrays have the same length
551   const length = arrays [0].length;

```

```

551     // Initialize the result array
552     const result = [];
553
554     // Apply the function element-wise
555     for(let i = 0; i < length; i++) {
556         // Get the corresponding elements from all arrays
557         const values = arrays.map(array => array[i]);
558
559         // Apply the function to the values and store in the result array
560         result.push(func(...values));
561     }
562
563     return result;
564 }
565
566 /**
567 * Applies a function to each element of an array or matrix along a
568 * specified dimension.
569 * @param {Array|Matrix} A - The array or matrix to process.
570 * @param {number} dim - The dimension along which to apply the function.
571 * @param {Function} fun - The function to apply to each element.
572 * @returns {Array|Matrix} The result of applying the function to A.
573 */
574 arrayfun(A, dim, fun) {
575     return this.jsl.env.math.apply(A, dim, fun);
576 }
577 /**
578 * Performs element-wise division on two arrays or matrices.
579 * @param {Array|Matrix} x - The numerator array or matrix.
580 * @param {Array|Matrix} y - The denominator array or matrix.
581 * @returns {Array|Matrix} The result of the element-wise division.
582 */
583 divideEl(x, y) {
584     return this.jsl.env.math.dotDivide(x, y);
585 }
586 /**
587 * Performs element-wise multiplication on two arrays or matrices.
588 * @param {Array|Matrix} x - The first array or matrix.
589 * @param {Array|Matrix} y - The second array or matrix.
590 * @returns {Array|Matrix} The result of the element-wise multiplication.
591 */
592 multiplyEl(x, y) {
593     return this.jsl.env.math.dotMultiply(x, y);
594 }
595
596 /**
597 * Raises elements of an array or matrix to the power of elements in another
598 * array or matrix, element-wise.
599 * @param {Array|Matrix} x - The base array or matrix.
600 * @param {Array|Matrix|number} y - The exponent array, matrix, or scalar.
601 * @returns {Array|Matrix} The result of the element-wise exponentiation.
602 */
603 powEl(x, y) {

```

```

604     return this.jsl.env.math.dotPow(x, y);
605   }
606
607   /**
608    * Calculates the dot product of two vectors or matrices.
609    * @param {number[]} x - The first input vector or flat array.
610    * @param {number[]} y - The second input vector or flat array.
611    * @param {number} [cols] - Optional number of columns to reshape the inputs
612    * into matrices.
613    * @returns {number | number[][]} The resulting dot product, either as a
614    * scalar or a matrix.
615    */
616   dot(x, y, cols) {
617     if(cols) {
618       var rows = x.length/cols;
619       var x_in = reshape(transpose(x, cols, rows), cols, rows);
620       var y_in = reshape(transpose(y, cols, rows), cols, rows);
621       return this.elementWise((a, b) => this.jsl.env.math.dot(a, b), x_in,
622                               y_in);
623     } else {
624       return this.jsl.env.math.dot(x, y);
625     }
626   }
627
628   /**
629    * Determines if the element is greater than zero.
630    * @param {number} e - The element to check.
631    * @returns {boolean} True if greater than zero, otherwise false.
632    */
633   lZero(e) {
634     return e > 0;
635   }
636
637   /**
638    * Converts a boolean array into an array of indices where the values are ‘
639    * true’.
640    * @param {boolean[]} arr - The input array of boolean values.
641    * @returns {number[]} An array of indices corresponding to ‘true’ values in
642    * the input array.
643    */
644   b2i(arr) {
645     return arr.map((value, index) => value ? index : -1).filter(index => index
646                   !== -1);
647   }
648
649   /**
650    * Calculates the average value of an array.
651    * @param {Array<number>} arr - The array to average.
652    * @returns {number} The average value.
653    */
654   average(arr) {
655     return arr.reduce(function(p, c) {
656       return p + c;
657     }, 0) / arr.length;
658   }

```

```

653
654  /**
655   * Calculates the Exponential Moving Average of the data.
656   * @param {number[]} data - The data array.
657   * @param {number} alpha - The smoothing factor between 0 and 1.
658   * @returns {number[]} The Exponential Moving Average of the data.
659   */
660 averageEM(data, alpha) {
661   var result = [];
662   var ema = data[0];
663   result.push(ema);
664   for(var i = 1; i < data.length; i++) {
665     ema = alpha * data[i] + (1 - alpha) * ema;
666     result.push(ema);
667   }
668   return result;
669 }
670
671 /**
672  * Applies a moving average filter to an input array while keeping the
673  * output array the same size.
674  * @param {number[]} inputArray - The array of numbers to filter.
675  * @param {number} windowSize - The size of the moving window (number of
676  * elements to average).
677  * @returns {number[]} The filtered array with the same length as the input
678  * array.
679  */
680 averageMoving(inputArray, windowSize) {
681   var result = [];
682   var len = inputArray.length;
683   var halfWindow = Math.floor(windowSize / 2);
684
685   for(var i = 0; i < len; i++) {
686     var start = i - halfWindow;
687     var end = i + halfWindow;
688
689     // Adjust the window if it goes beyond the array bounds
690     if(start < 0) start = 0;
691     if(end >= len) end = len - 1;
692
693     var sum = 0;
694     var count = 0;
695     for(var j = start; j <= end; j++) {
696       sum += inputArray[j];
697       count++;
698     }
699     result.push(sum / count);
700   }
701   return result;
702 }
703 /**
704  * Applies a moving average filter to an input array while keeping the
705  * output array the same size.
706  * @param {number[]} inputArray - The array of numbers to filter.

```

```
704     * @param {number} windowSize - The size of the moving window (number of
705     * elements to average).
706     * @returns {number[]} The filtered array with the same length as the input
707     * array.
708     */
709     movmean(inputArray, windowSize) {
710       return this.averageMoving(inputArray, windowSize);
711     }
712   /**
713    * Determines if two arrays are equal.
714    * @param {Array} A1 - The first array.
715    * @param {Array} A2 - The second array.
716    * @returns {boolean} True if arrays are equal, otherwise false.
717    */
718   isEqual(A1, A2) {
719     let n;
720     if((n = A1.length) != A2.length) return false;
721     for(let i = 0; i < n; i++) if(A1[i] !== A2[i]) return false;
722     return true;
723   }
724   /**
725    * Negates the boolean values in an array.
726    * @param {Array<boolean>} A - The array to negate.
727    * @returns {Array<boolean>} The negated array.
728    */
729   neg(A) {
730     var B = new Array(A.length);
731     for(var i = 0; i < A.length; i++) {
732       B[i] = !A[i];
733     }
734     return B;
735   }
736   /**
737    * Determines if all elements in an array evaluate to true.
738    * @param {Array<boolean>} A - The array to check.
739    * @returns {boolean} True if all elements are true, otherwise false.
740    */
741   all(A) {
742     let n = A.length;
743     for(let i = 0; i < n; i++) {
744       if(!A[i]) {
745         return false;
746       }
747     }
748   }
749   return true;
750 }
751 /**
752  * Determines if any element in an array evaluates to true.
753  * @param {Array<boolean>} A - The array to check.
754  * @returns {boolean} True if any element is true, otherwise false.
755  */
756
```

```

757     any(A) {
758         let n = A.length;
759         for(let i = 0; i < n; i++) {
760             if(A[i]) {
761                 return true;
762             }
763         }
764         return false;
765     }
766
767     /**
768      * Checks if the array contains a specified item.
769      * @param {Array} arr - The array to search.
770      * @param {*} item - The item to search for.
771      * @returns {boolean} True if the item is found, otherwise false.
772      */
773     arrayContains(arr, item) {
774         if(!Array.prototype.indexOf) {
775             var i = arr.length;
776             while(i--) {
777                 if(arr[i] === item) {
778                     return true;
779                 }
780             }
781             return false;
782         }
783         return arr.indexOf(item) !== -1;
784     }
785
786     /**
787      * Checks if an array contains any duplicate elements.
788      * @param {Array} array - The array to check for duplicates.
789      * @returns {boolean} True if duplicates are found, false otherwise.
790      */
791     hasDuplicates(array) {
792         return new Set(array).size !== array.length;
793     }
794
795     /**
796      * Removes duplicate values from an array.
797      * @param {Array} arr - The array from which duplicates are to be removed.
798      * @returns {Array} An array containing only unique elements from the
799      * original array.
800      */
801     removeDuplicates(arr) {
802         return arr.filter(function(item, index) { return arr.indexOf(item) ===
803             index; });
804     }
805
806     /**
807      * Reverses the order of elements in each row of a matrix.
808      * @param {Array} array - The matrix array to flip horizontally.
809      * @returns {Array} The horizontally flipped matrix.
810      */
811     fliplr(array) {

```

```

810     return array.reverse();
811 }
812
813 /**
814 * Moves the first 'n' elements of the array to the end.
815 * @param {Array} array - The array to be modified.
816 * @param {number} n - The number of elements to move from the start to the
817 * end.
818 */
819 movelr(array, n) {
820   var A = [...array];
821   if(A.length === 0 || n <= 0) return array;
822   n = n % A.length;
823   const elements_to_move = A.splice(0, n);
824   A.push(...elements_to_move);
825   return A;
826 }
827
828 /**
829 * Adds multiple operands, which can be either scalars or arrays.
830 * If multiple operands are arrays, they must be of the same length.
831 * @param {...number|Array<number>} args - The operands, scalar or arrays.
832 * @returns {number|Array<number>} The result of adding all operands.
833 * @throws {Error} If input types are invalid or arrays have different
834 * lengths.
835 */
836 plus(...args) {
837   if(args.length === 0) {
838     this.jsl.env.error('@plus: No arguments provided');
839   }
840
841   // Helper function to add two operands
842   const addTwo = (a, b) => {
843     if(Array.isArray(a) && Array.isArray(b)) {
844       if(a.length !== b.length) {
845         this.jsl.env.error('@plus: ' + language.string(176)); // Arrays have
846         different lengths
847       }
848       return a.map((val, idx) => this.plus(val, b[idx]));
849     } else if(Array.isArray(a) && typeof b === 'number') {
850       return a.map(val => this.plus(val, b));
851     } else if(typeof a === 'number' && Array.isArray(b)) {
852       return b.map(val => this.plus(a, val));
853     } else if(typeof a === 'number' && typeof b === 'number') {
854       return a + b;
855     } else {
856       this.jsl.env.error('@plus: ' + language.string(177)); // Invalid type
857     }
858     return false;
859   };
860
861   // Iterate through all arguments and accumulate the result
862   return args.reduce((acc, current) => addTwo(acc, current));

```

```

861 }
862
863 /**
864 * Adds multiple operands , which can be either scalars or arrays .
865 * If multiple operands are arrays , they must be of the same length .
866 * @param { ... number | Array<number> } args - The operands , scalar or arrays .
867 * @returns { number | Array<number> } The result of adding all operands .
868 * @throws { Error } If input types are invalid or arrays have different
869 lengths .
870 */
871 add ( ... args ) {
872   return this . plus ( ... args );
873 }
874 /**
875 * Subtracts multiple operands from the first one , which can be either
876 scalars or arrays .
877 * If multiple operands are arrays , they must be of the same length .
878 * @param { ... number | Array<number> } args - The operands , scalar or arrays .
879 * @returns { number | Array<number> } The result of subtracting all subsequent
880 operands from the first one .
881 * @throws { Error } If input types are invalid or arrays have different
882 lengths .
883 */
884 minus ( ... args ) {
885   if ( args . length === 0 ) {
886     this . jsl . env . error ( '@minus: No arguments provided ' );
887   }
888
889   // Helper function to subtract two operands
890   const subtractTwo = ( a , b ) => {
891     if ( Array . isArray ( a ) && Array . isArray ( b ) ) {
892       if ( a . length !== b . length ) {
893         this . jsl . env . error ( '@minus: ' + language . string ( 176 ) ); // Arrays
894         have different lengths
895       }
896       return a . map ( ( val , idx ) => this . minus ( val , b [ idx ] ) );
897     } else if ( Array . isArray ( a ) && typeof b === ' number ' ) {
898       return a . map ( val => this . minus ( val , b ) );
899     } else if ( typeof a === ' number ' && Array . isArray ( b ) ) {
900       // Subtracting an array from a scalar: scalar - array
901       return b . map ( val => this . minus ( a , val ) );
902     } else if ( typeof a === ' number ' && typeof b === ' number ' ) {
903       return a - b ;
904     } else {
905       this . jsl . env . error ( '@minus: ' + language . string ( 177 ) ); // Invalid type
906     }
907   }
908   return false ;
909 }
910
911 // The first argument is the initial value
912 let result = args [ 0 ];
913
914 // Iterate through the rest of the arguments and subtract each from the
915 result

```

```

910     for(let i = 1; i < args.length; i++) {
911         result = subtractTwo(result, args[i]);
912     }
913
914     return result;
915 }
916
917 /**
918 * Subtracts multiple operands from the first one, which can be either
919 scalars or arrays.
920 * If multiple operands are arrays, they must be of the same length.
921 * @param {...number|Array<number>} args - The operands, scalar or arrays.
922 * @returns {number|Array<number>} The result of subtracting all subsequent
923   operands from the first one.
924 * @throws {Error} If input types are invalid or arrays have different
925   lengths.
926 */
927 subtract(...args) {
928     return this.minus(...args);
929 }
930
931 /**
932 * Computes the cross product of two 3D vectors.
933 * @param {number[]} A - The first 3D vector.
934 * @param {number[]} B - The second 3D vector.
935 * @param {number} cols - The number of columns (typically 1 for single
936   vectors).
937 * @returns {number[]} The cross product vector.
938 */
939 cross3D(A, B, cols = 1) {
940     var C = new Array(3 * cols).fill(0);
941     for(var j = 0; j < cols; j++) {
942         C[j * 3] = A[j * 3 + 1] * B[j * 3 + 2] - A[j * 3 + 2] * B[j * 3 + 1];
943         C[j * 3 + 1] = A[j * 3 + 2] * B[j * 3] - A[j * 3] * B[j * 3 + 2];
944         C[j * 3 + 2] = A[j * 3] * B[j * 3 + 1] - A[j * 3 + 1] * B[j * 3];
945     }
946     return C;
947 }
948
949 /**
950 * Computes the inverse of a matrix represented as a flat array.
951 * @param {number[]} A - The input matrix in a flat array format.
952 * @param {number} size - The number of rows and columns in the matrix.
953 * @returns {number[]|false} The inverted matrix as a flat array, or 'false'
954   if the matrix is non-invertible.
955 */
956 inv(A, size) {
957     var A_mat = reshape(A, size, size);
958     try {
959         return reshape(this.jsl.env.math.inv(A_mat), size*size, 1);
960     } catch {
961         return false;
962     }
963 }

```

```

960  /**
961   * Concatenates multiple matrices row-wise.
962   * @param {number} cols_C - The number of columns in the concatenated matrix
963   *
964   * @param {...Array} args - The matrices to concatenate.
965   * @returns {Array} The concatenated matrix.
966   */
967   concatRow(cols_C, ...args) {
968     var N = args.length;
969     var rows_C = args.reduce((a, e) => a += e.length, 0) / cols_C;
970
971     var C = new Array(rows_C * cols_C).fill(0);
972
973     var p = 0;
974     for(var j = 0; j < cols_C; j++) {
975       for(var k = 0; k < N; k++) {
976         var P = args[k].length / cols_C;
977         for(var i = 0; i < P; i++) {
978           C[p++] = args[k][j * P + i];
979         }
980       }
981     }
982     return C;
983   }
984 /**
985  * Concatenates multiple matrices column-wise.
986  * @param {number} rows_C - The number of rows in the concatenated matrix.
987  * @param {...Array} args - The matrices to concatenate.
988  * @returns {Array} The concatenated matrix.
989  */
990 concatCol(rows_C, ...args) {
991   return args.flat();
992 }
993
994 /**
995  * Concatenates multiple vectors.
996  * @param {...Array} args - The vectors to concatenate.
997  * @returns {Array} The concatenated vector.
998  */
999 concat(...args) {
1000   return args.flat();
1001 }
1002
1003 /**
1004  * Repeats a row vector multiple times.
1005  * @param {Array} A - The row vector to repeat.
1006  * @param {number} rows - The number of times to repeat the row.
1007  * @returns {Array} The repeated row matrix.
1008  */
1009 repRow(A, rows) {
1010   var cols = A.length;
1011   var C = new Array(cols * rows).fill(0);
1012   for(var i = 0; i < cols; i++) {
1013     for(var j = 0; j < rows; j++) {

```

```

1014         C[ i *rows+j ] = A[ i ];
1015     }
1016   }
1017   return C;
1018 }
1019
1020 /**
1021 * Repeats a column vector multiple times.
1022 * @param {Array} A - The column vector to repeat.
1023 * @param {number} cols - The number of times to repeat the column.
1024 * @returns {Array} The repeated column matrix.
1025 */
1026 repCol(A, cols) {
1027   var rows = A.length;
1028   var C = new Array(cols * rows).fill(0);
1029   for(var i = 0; i < cols; i++) {
1030     for(var j = 0; j < rows; j++) {
1031       C[ i*rows+j ] = A[ j ];
1032     }
1033   }
1034   return C;
1035 }
1036
1037 /**
1038 * Sums the elements of each row in a matrix.
1039 * @param {Array<number>} A - The matrix array.
1040 * @param {number} rows - The number of rows in the matrix.
1041 * @param {number} cols - The number of columns in the matrix.
1042 * @returns {number[]} An array containing the sum of each row.
1043 */
1044 sumRow(A, rows, cols) {
1045   var C = new Array(rows).fill(0);
1046   for(var i = 0; i < rows; i++) {
1047     for(var j = 0; j < cols; j++) {
1048       C[ i ] += A[ i*cols+j ];
1049     }
1050   }
1051   return C;
1052 }
1053
1054 /**
1055 * Sums the elements of each column in a matrix.
1056 * @param {Array<number>} A - The matrix array.
1057 * @param {number} rows - The number of rows in the matrix.
1058 * @param {number} cols - The number of columns in the matrix.
1059 * @returns {number[]} An array containing the sum of each column.
1060 */
1061 sumCol(A, rows, cols) {
1062   var C = new Array(cols).fill(0);
1063   for(var j = 0; j < cols; j++) {
1064     for(var i = 0; i < rows; i++) {
1065       C[ j ] += A[ j*rows+i ];
1066     }
1067   }
1068   return C;

```

```
1069 }
1070
1071 /**
1072 * Calculates the Euclidean norm of each row in a matrix.
1073 * @param {Array<number>} A - The matrix array.
1074 * @param {number} rows - The number of rows in the matrix.
1075 * @param {number} cols - The number of columns in the matrix.
1076 * @returns {number[]} An array containing the norm of each row.
1077 */
1078 normRow(A, rows, cols) {
1079     var C = new Array(rows).fill(0);
1080     for(var i = 0; i < rows; i++) {
1081         for(var j = 0; j < cols; j++) {
1082             C[i] += Math.pow(A[i*cols+j], 2);
1083         }
1084         C[i] = Math.sqrt(C[i]);
1085     }
1086     return C;
1087 }
1088
1089 /**
1090 * Calculates the Euclidean norm of each column in a matrix.
1091 * @param {Array<number>} A - The matrix array.
1092 * @param {number} rows - The number of rows in the matrix.
1093 * @param {number} cols - The number of columns in the matrix.
1094 * @returns {number[]} An array containing the norm of each column.
1095 */
1096 normCol(A, rows, cols) {
1097     var C = new Array(cols).fill(0);
1098     for(var j = 0; j < cols; j++) {
1099         for(var i = 0; i < rows; i++) {
1100             C[j] += Math.pow(A[j*rows+i], 2);
1101         }
1102         C[j] = Math.sqrt(C[j]);
1103     }
1104     return C;
1105 }
1106
1107 /**
1108 * Transposes a matrix.
1109 * @param {Array<number>} A - The matrix array to transpose.
1110 * @param {number} rows - The number of rows in the original matrix.
1111 * @param {number} cols - The number of columns in the original matrix.
1112 * @returns {Array<number>} The transposed matrix array.
1113 */
1114 transpose(A, rows, cols) {
1115     var C = new Array(rows * cols).fill(0);
1116     for(var i = 0; i < rows; i++) {
1117         for(var j = 0; j < cols; j++) {
1118             C[j * rows + i] = A[i * cols + j];
1119         }
1120     }
1121     return C;
1122 }
1123
```

```

1124 /**
1125 * Multiplies two matrices.
1126 * @param {Array<number>} A - The first matrix array.
1127 * @param {Array<number>} B - The second matrix array.
1128 * @param {number} rows_A - The number of rows in matrix A.
1129 * @param {number} cols_A - The number of columns in matrix A.
1130 * @param {number} cols_B - The number of columns in matrix B.
1131 * @returns {Array<number>} The resulting matrix array after multiplication.
1132 */
1133 multiply(A, B, rows_A, cols_A, cols_B) {
1134   var C = new Array(rows_A * cols_B).fill(0);
1135   for(var i = 0; i < rows_A; i++) {
1136     for(var j = 0; j < cols_A; j++) {
1137       for(var k = 0; k < cols_B; k++) {
1138         C[k * rows_A + i] += A[j * rows_A + i] * B[k * cols_A + j];
1139       }
1140     }
1141   }
1142   return C;
1143 }
1144
1145 /**
1146 * Performs element-wise dot product on columns of two matrices.
1147 * @param {Array<number>} A - The first matrix array.
1148 * @param {Array<number>} B - The second matrix array.
1149 * @param {number} rows - The number of rows in each matrix.
1150 * @param {number} cols - The number of columns in each matrix.
1151 * @returns {Array<number>} An array containing the dot product of each
1152 * column.
1153 */
1154 dotColumn(A, B, rows, cols) {
1155   var C = new Array(cols).fill(0);
1156   for(var j = 0; j < cols; j++) {
1157     for(var i = 0; i < rows; i++) {
1158       C[j] += A[j * rows + i] * B[j * rows + i];
1159     }
1160   }
1161   return C;
1162 }
1163 /**
1164 * Creates a diagonal matrix from a given array.
1165 * @param {number[]} A - The array to form the diagonal.
1166 * @param {number} length - The size of the square matrix.
1167 * @returns {Array<number>} The diagonal matrix as a 1D array.
1168 */
1169 diag(A, length) {
1170   var B = new Array(length*length).fill(0);
1171   for(var i = 0; i < length; i++) {
1172     B[i * length + i] = A[i];
1173   }
1174   return B;
1175 }
1176 /**
1177 */

```

```

1178 * Solves a linear system of equations using LU decomposition.
1179 * @param {Array<number>} A - The coefficient matrix.
1180 * @param {Array<number>} B - The constant terms.
1181 * @param {number} N - The size of the matrix (NxN).
1182 * @returns {Array<number>} The solution vector.
1183 */
1184 linsolve(A, B, N) {
1185   return this.reshape(this.jsl.env.math.lusolve(this.reshape(A, N, N), B),
1186                      1, N);
1187 }
1188 /**
1189 * Computes the reciprocal of each element in the array.
1190 * @param {Array<number>} A - The array of numbers.
1191 * @param {number} length - The number of elements in the array.
1192 * @returns {Array<number>} An array containing the reciprocals of the
1193   original elements.
1194 */
1195 reciprocal(A, length) {
1196   var B = new Array(length).fill(0);
1197   for(var i = 0; i < length; i++) {
1198     B[i] = 1 / A[i];
1199   }
1200   return B;
1201 }
1202 /**
1203 * Reshapes an array into a new dimension.
1204 * @param {Array} A - The array to reshape.
1205 * @param {number} rows - The number of rows in the new shape.
1206 * @param {number} cols - The number of columns in the new shape.
1207 * @returns {Array} The reshaped array.
1208 * @throws {Error} If the total number of elements does not match.
1209 */
1210 reshape(A, rows, cols) {
1211   if(rows === _) {
1212     rows = A.length / cols;
1213   }
1214   if(cols === _) {
1215     cols = A.length / rows;
1216   }
1217   let flat_arr = [];
1218
1219   if(Array.isArray(A[0])) {
1220     var numRows = A.length;
1221     var numCols = A[0].length;
1222
1223     for(let j = 0; j < numCols; j++) {
1224       for(let i = 0; i < numRows; i++) {
1225         flat_arr.push(A[i][j]);
1226       }
1227     }
1228   } else {
1229     flat_arr = A.slice();
1230   }

```

```

1231
1232   if(flat_arr.length !== rows * cols) {
1233     this.jsl.env.error('@reshape: '+language.string(178));
1234   }
1235
1236   var reshaped = Array.from({ length: rows }, () => Array(cols));
1237
1238   for(let j = 0; j < cols; j++) {
1239     for(let i = 0; i < rows; i++) {
1240       var index = j * rows + i;
1241       reshaped[i][j] = flat_arr[index];
1242     }
1243   }
1244
1245   if(rows === 1 || cols === 1) {
1246     reshaped = reshaped.flat();
1247   }
1248   return reshaped;
1249 }
1250
1251 /**
1252 * Finds the maximum element in the array and its index, excluding NaN
1253 * values.
1254 * @param {number[]} A - The array to search.
1255 * @returns {Array} An array containing the max value and its index.
1256 * @throws {TypeError} If the input is not an array.
1257 * @throws {Error} If the array is empty or only contains NaN values.
1258 */
1259 maxi(A) {
1260   if(!Array.isArray(A)) {
1261     this.jsl.env.error('@maxi: '+language.string(190));
1262   }
1263
1264   const filtered_A = A.filter(num => !isNaN(num));
1265   if(filtered_A.length === 0) {
1266     this.jsl.env.error('@maxi: '+language.string(191));
1267   }
1268
1269   let max = filtered_A[0];
1270   let index = A.indexOf(max);
1271
1272   for(let i = 1; i < A.length; i++) {
1273     const current = A[i];
1274     if(!isNaN(current) && current > max) {
1275       max = current;
1276       index = i;
1277     }
1278   }
1279
1280   return [max, index];
1281 }
1282
1283 /**
1284 * Finds the minimum element in the array and its index, excluding NaN
1285 * values.

```

```

1284 * @param {number[]} A - The array to search .
1285 * @returns {Array} An array containing the min value and its index .
1286 * @throws {TypeError} If the input is not an array .
1287 * @throws {Error} If the array is empty or only contains NaN values .
1288 */
1289 mini(A) {
1290   if (!Array.isArray(A)) {
1291     this.jsl.env.error('@mini: '+language.string(190));
1292   }
1293
1294   const filtered_A = A.filter(num => !isNaN(num));
1295   if(filtered_A.length === 0) {
1296     this.jsl.env.error('@mini: '+language.string(191));
1297   }
1298
1299   let min = filtered_A[0];
1300   let index = A.indexOf(min);
1301
1302   for(let i = 1; i < A.length; i++) {
1303     const current = A[i];
1304     if(!isNaN(current) && current < min) {
1305       min = current;
1306       index = i;
1307     }
1308   }
1309
1310   return [min, index];
1311 }
1312
1313 /**
1314 * Sorts the array in ascending order with indices , excluding NaN values .
1315 * @param {number[]} A - The array to sort .
1316 * @returns {Array} An array containing the sorted values and their original
1317 *                 indices .
1318 * @throws {TypeError} If the input is not an array .
1319 */
1320 sorti(A) {
1321   if (!Array.isArray(A)) {
1322     this.jsl.env.error('@sorti: '+language.string(190));
1323   }
1324
1325   // Create an array of indices and sort based on values in A,
1326   // with NaNs sorted to the end .
1327   const indices = A.map(_ , idx) => idx;
1328   indices.sort((i, j) => {
1329     if(isNaN(A[i]) && isNaN(A[j])) return 0;
1330     if(isNaN(A[i])) return 1;
1331     if(isNaN(A[j])) return -1;
1332     return A[i] - A[j];
1333   });
1334
1335   // Map the sorted indices to the sorted values
1336   const sorted_scores = indices.map(i => A[i]);
1337   return [sorted_scores, indices];
}

```

```

1338
1339 /**
1340 * Computes the weighted sum of two vectors and stores the result in the 'ret' array.
1341 * @param {number[]} ret - The array to store the result.
1342 * @param {number} w1 - Weight for the first vector.
1343 * @param {number[]} v1 - The first vector.
1344 * @param {number} w2 - Weight for the second vector.
1345 * @param {number[]} v2 - The second vector.
1346 */
1347 weightedSum(ret, w1, v1, w2, v2) {
1348   for(let j = 0; j < ret.length; ++j) {
1349     ret[j] = w1 * v1[j] + w2 * v2[j];
1350   }
1351 }
1352
1353 /**
1354 * Computes the consecutive differences of elements in an array.
1355 * @param {number[]} A - The input array of numbers.
1356 * @returns {number[]} An array containing the differences between consecutive elements of the input array.
1357 */
1358 condiff(A) {
1359   return A.slice(1).map((num, i) => num - A[i]);
1360 }
1361
1362 /**
1363 * Generates an array with random floating-point numbers within a specified range.
1364 * @param {number} l - The lower bound of the range.
1365 * @param {number} u - The upper bound of the range.
1366 * @param {number} rows - The number of rows.
1367 * @param {number} cols - The number of columns.
1368 * @param {Function} [randFun=Math.random] - The random function to use.
1369 * @returns {Array<number>} An array filled with random numbers within the specified range.
1370 */
1371 arrayRand(l, u, rows, cols, randFun) {
1372   if(!this.jsl.formatisFunction(randFun)) {
1373     randFun = Math.random;
1374   }
1375   return this.plus(this.repCol(l, cols), this.multiplyEl(repCol(minus(u, 1), cols), Array.from({length: rows * cols}, () => randFun())));
1376 }
1377
1378 /**
1379 * Generates an array with random integer numbers within a specified range.
1380 * @param {number[]} lu - An array containing the lower and upper bounds [lower, upper].
1381 * @param {number} rows - The number of rows.
1382 * @param {number} cols - The number of columns.
1383 * @param {Function} [randFun=Math.random] - The random function to use.
1384 * @returns {Array<number>} An array filled with random integers within the specified range.
1385 */

```

```

1386 arrayRandi(lu, rows, cols, randFun) {
1387   if(!this.jsl.format.isFunction(randFun)) {
1388     randFun = Math.random;
1389   }
1390   return Array.from({length: rows * cols}, () => Math.floor(randFun() * (
1391     lu[1] - lu[0] + 1) + lu[0]));
1392 }
1393 /**
1394 * Normalizes a 3D vector.
1395 * @param {number[]} v - The vector to normalize.
1396 * @returns {number[]} The normalized vector.
1397 */
1398 normalizeVector(v) {
1399   var len = this.jsl.env.math.norm(v);
1400   if(len === 0) return [0, 0, 0];
1401   return [v[0]/len, v[1]/len, v[2]/len];
1402 }
1403 /**
1404 * Computes the dot product of two 3D vectors.
1405 * @param {number[]} a - The first vector.
1406 * @param {number[]} b - The second vector.
1407 * @returns {number} The dot product of the vectors.
1408 */
1409 dotVector(a, b) {
1410   return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
1411 }
1412 /**
1413 * Calculates the angle between two vectors.
1414 * @param {number[]} a - The first vector.
1415 * @param {number[]} b - The second vector.
1416 * @returns {number} The angle in radians between vectors a and b.
1417 */
1418 angleVectors(a, b) {
1419   return acos(dotVector(a, b)/(norm(a)*norm(b)));
1420 }
1421 /**
1422 * Creates a skew-symmetric matrix from a 3D vector.
1423 * @param {number[]} v - The vector to skew.
1424 * @returns {number[]} The skew-symmetric matrix as a 1D array.
1425 */
1426 skewVector(v) {
1427   return [
1428     0, v[2], -v[1],
1429     -v[2], 0, v[0],
1430     v[1], -v[0], 0
1431   ];
1432 }
1433 /**
1434 * Generates coordinate matrices from coordinate vectors for N dimensions.
1435 * @param {...number[]} args - The coordinate vectors.
1436 */
1437

```

```

1440   * @returns {Array} The coordinate grids for each dimension.
1441   */
1442 meshgrid(...args) {
1443   var obj = this;
1444   const ndim = args.length;
1445   const sizes = args.map(a => a.length);
1446   const grids = [];
1447
1448   // Initialize the grids with N-dimensional arrays
1449   for(let k = 0; k < ndim; k++) {
1450     grids[k] = this.createArray(...sizes);
1451   }
1452
1453   // Recursive function to fill in the grids
1454   function fillGrid(indices, dim) {
1455     if(dim === ndim) {
1456       for(let k = 0; k < ndim; k++) {
1457         obj.setValueAt(grids[k], indices, args[k][indices[(k + 1) % ndim]]);
1458       }
1459     } else {
1460       for(let i = 0; i < sizes[dim]; i++) {
1461         indices[dim] = i;
1462         fillGrid(indices, dim + 1);
1463       }
1464     }
1465   }
1466
1467   fillGrid(new Array(ndim), 0);
1468
1469   return grids;
1470 }
1471
1472 /**
1473 * Determines if two arrays are equal by comparing each element.
1474 * @param {Array} A1 - The first array to compare.
1475 * @param {Array} A2 - The second array to compare.
1476 * @returns {boolean} True if the arrays are equal, otherwise false.
1477 */
1478 areEqual(A1, A2) {
1479   let n;
1480   if((n = A1.length) != A2.length) return false;
1481   for(let i = 0; i < n; i++) if(A1[i] !== A2[i]) return false;
1482   return true;
1483 }
1484
1485 /**
1486 * Displays a matrix with a variable name.
1487 * @param {string} varname - The name of the variable.
1488 * @param {number[]} A - The matrix array.
1489 * @param {number} rows - The number of rows in the matrix.
1490 * @param {number} cols - The number of columns in the matrix.
1491 * @returns {string} The string representation of the matrix.
1492 */
1493 dispMatrix(varname, A, rows, cols) {
1494   var str = ' ' + varname + ' = [ \n';

```

```

1495     for(var i = 0; i < rows; i++) {
1496         str += ',';
1497         for(var j = 0; j < cols; j++) {
1498             str += num2str(A[j * rows + i], 8) + ',';
1499         }
1500         str += '\n';
1501     }
1502     str += ']';
1503     this.jsl.env.disp(str);
1504     return str + '\n';
1505 }
1506
1507 /**
1508 * Displays a row vector with a variable name.
1509 * @param {string} varname - The name of the variable.
1510 * @param {number[]} A - The row vector array.
1511 * @param {number} length - The length of the row vector.
1512 * @returns {string} The string representation of the row vector.
1513 */
1514 dispRowVector(varname, A, length) {
1515     var str = ',' + varname + '='[ ';
1516     for(var i = 0; i < length; i++) {
1517         str += num2str(A[i], 8) + ',';
1518     }
1519     str += ']';
1520     this.jsl.env.disp(str);
1521     return str + '\n';
1522 }
1523
1524 /**
1525 * Displays a column vector with a variable name.
1526 * @param {string} varname - The name of the variable.
1527 * @param {number[]} A - The column vector array.
1528 * @param {number} length - The length of the column vector.
1529 * @returns {string} The string representation of the column vector.
1530 */
1531 dispColumnVector(varname, A, length) {
1532     var str = ',' + varname + '='[\n';
1533     for(var j = 0; j < length; j++) {
1534         str += ',' + num2str(A[j], 8) + '\n';
1535     }
1536     str += ']';
1537     this.jsl.env.disp(str);
1538     return str + '\n';
1539 }
1540 }
1541
1542 exports.PRDC_JSLAB_LIB_ARRAY = PRDC_JSLAB_LIB_ARRAY;

```

Listing 84 - array.js

```

1 /**
2 * @file JSLAB library basic submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com

```

```

6   */
7
8 /**
9  * Class for JSLAB basic submodule.
10 */
11 class PRDC_JSLAB_LIB_BASIC {
12
13 /**
14  * Initializes a new instance of the PRDC_JSLAB_LIB_BASIC class.
15  * @param {Object} jsl The JSLAB application instance this submodule is part
16  * of.
17 */
18 constructor(jsl) {
19   var obj = this;
20   this.jsl = jsl;
21
22   this._docs = JSON.parse(this.jsl.env.readFileSync(app_path + '/docs/
23   documentation.json', 'utf8'));
24
25 /**
26  * Stores the result of the last evaluated expression.
27  * @type {any}
28 */
29 this.ans;
30
31 /**
32  * Clears all defined variables in the current context.
33  * @name clear
34  * @kind function
35  * @memberof PRDC_JSLAB_LIB_BASIC
36 */
37 Object.defineProperty(this.jsl.context, 'clear', { configurable: true, get
38   : this._clear });
39
40 /**
41  * Clears the console screen.
42  * @name clc
43  * @kind function
44  * @memberof PRDC_JSLAB_LIB_BASIC
45 */
46 Object.defineProperty(this.jsl.context, 'clc', { configurable: true, get:
47   this._clc });
48
49 /**
50  * Clears the console screen. Alias for 'clc'.
51  * @name cls
52  * @kind function
53  * @memberof PRDC_JSLAB_LIB_BASIC
54 */
55 Object.defineProperty(this.jsl.context, 'cls', { configurable: true, get:
56   this._clc });
57
58 /**
59  * Returns the current version of the JSLAB.
60  * @name version

```

```
56     * @kind member
57     * @memberof PRDC_JSLAB_LIB_BASIC
58     */
59 Object.defineProperty(this.jsl.context, 'version', { configurable: true,
60     get: this._version });
61
62 /**
63 * Returns the platform on which JSLAB is running.
64 * @name platform
65 * @kind member
66 * @memberof PRDC_JSLAB_LIB_BASIC
67 */
68 Object.defineProperty(this.jsl.context, 'platform', { configurable: true,
69     get: this._platform });
70
71 /**
72 * Returns the file name of the current JSLAB script.
73 * @name js1_file_name
74 * @kind member
75 * @memberof PRDC_JSLAB_LIB_BASIC
76 */
77 Object.defineProperty(this.jsl.context, 'js1_file_name', { configurable:
78     true, get: this.js1FileName });
79
80 /**
81 * Provides information about the current environment.
82 * @name info
83 * @kind function
84 * @memberof PRDC_JSLAB_LIB_BASIC
85 */
86 Object.defineProperty(this.jsl.context, 'info', { configurable: true, get:
87     this._info });
88
89 /**
90 * Accesses or modifies the user settings for JSLAB.
91 * @name settings
92 * @kind function
93 * @memberof PRDC_JSLAB_LIB_BASIC
94 */
95 Object.defineProperty(this.jsl.context, 'settings', { configurable: true,
96     get: this._settings });
97
98 /**
99 * Provides help information for JSLAB commands.
100 * @name cmd_help
101 * @kind function
102 * @memberof PRDC_JSLAB_LIB_BASIC
103 */
104 Object.defineProperty(this.jsl.context, 'cmd_help', { configurable: true,
105     get: this._cmd_help });
106
107 /**
108 * Accesses the code editor interface within JSLAB.
109 * @name editor
110 * @kind function
```

```
105     * @memberof PRDC_JSLAB_LIB_BASIC
106     */
107    Object.defineProperty(this.jsl.context, 'editor', { configurable: true,
108      get: this._editor });
109
110 /**
111  * Returns the current working directory.
112  * @name pwd
113  * @kind member
114  * @memberof PRDC_JSLAB_LIB_BASIC
115 */
116 Object.defineProperty(this.jsl.context, 'pwd', { configurable: true, get:
117   this._pwd });
118
119 /**
120  * Sets a breakpoint in the code for debugging.
121  * @name breakpoint
122  * @kind function
123  * @memberof PRDC_JSLAB_LIB_BASIC
124 */
125 Object.defineProperty(this.jsl.context, 'breakpoint', { configurable: true
126   , get: this._breakpoint });
127
128 /**
129  * Returns the current debug flag status.
130  * @name debug_flag
131  * @kind member
132  * @memberof PRDC_JSLAB_LIB_BASIC
133 */
134 Object.defineProperty(this.jsl.context, 'debug_flag', { configurable: true
135   , get: this._debug_flag });
136
137 /**
138  * Enables or disables debug mode.
139  * @name debug
140  * @kind member
141  * @memberof PRDC_JSLAB_LIB_BASIC
142 */
143 Object.defineProperty(this.jsl.context, 'debug', { configurable: true, get
144   : this._debug });
145
146 /**
147  * Pauses the execution of the current script.
148  * @name pause
149  * @kind function
150  * @memberof PRDC_JSLAB_LIB_BASIC
151 */
152 Object.defineProperty(this.jsl.context, 'pause', { configurable: true, get
153   : this._pause });

154 /**
155  * Sets a stop point in the script execution.
156  * @name stoppoint
157  * @kind function
158  * @memberof PRDC_JSLAB_LIB_BASIC
```

```

154     */
155     Object.defineProperty(this.jsl.context, 'stoppoint', { configurable: true,
156         get: this._stoppoint });
157
158     /**
159      * Sets a log point to record information during execution.
160      * @name logpoint
161      * @kind function
162      * @memberof PRDC_JSLAB_LIB_BASIC
163      */
164     Object.defineProperty(this.jsl.context, 'logpoint', { configurable: true,
165         get: this._logpoint });
166
167     /**
168      * Updates specific points in the script during execution.
169      * @name updatepoint
170      * @kind function
171      * @memberof PRDC_JSLAB_LIB_BASIC
172      */
173     Object.defineProperty(this.jsl.context, 'updatepoint', { configurable:
174         true, get: this._updatepoint });
175
176     /**
177      * Checks if the execution should stop based on conditions.
178      * @name checkStop
179      * @kind function
180      * @memberof PRDC_JSLAB_LIB_BASIC
181      */
182     Object.defineProperty(this.jsl.context, 'checkStop', { configurable: true,
183         get: this._checkStop });
184
185     /**
186      * Marks the endpoint of a script or process.
187      * @name endPoint
188      * @kind function
189      * @memberof PRDC_JSLAB_LIB_BASIC
190      */
191     Object.defineProperty(this.jsl.context, 'endPoint', { configurable: true,
192         get: this._endPoint });
193
194     /**
195      * Runs a script from a specified path, optionally focusing on specific
196      * lines and controlling output visibility.
197      * @param {string} script_path The path to the script to run.
198      * @param {Array<number>} lines An array of line numbers to run or focus on
199      * within the script.
200      * @param {boolean} [silent=false] Whether to suppress output from the
201      * script execution.
202      * @param {Boolean} [force_run=false] If true, forces the script to run even
203      * if stop conditions are met.
204      */
205     async run(script_path, lines, silent = false, force_run = false) {
206         if(this.jsl.env.pathIsAbsolute(script_path)) {
207             this.jsl.last_script_path = script_path;
208         }
209     }

```

```

200     this.jsl.last_script_lines = lines;
201     this.jsl.last_script_silent = silent;
202     if(!force_run && this.jsl.env.checkScriptDir(script_path)) {
203         return;
204     }
205 }
206 return await this.jsl.eval.runScript(script_path, lines, silent, force_run
207 );
208
209 /**
210 * Retrieves documentation in JSON format based on the provided name and
211 * type.
212 * @param {string} [name] – The name of the documentation item.
213 * @param {string} [type] – The type of the documentation (e.g., 'category')
214 *
215 * @returns {string|undefined} The JSON string of the documentation or
216 * undefined if not found.
217 */
218 helpToJson(name, type) {
219     if(!name) {
220         return this.jsl.setDepthSafeStringify(this._docs, 4);
221     }
222
223     const parts = name.split('.');
224     if(parts.length === 2) {
225         const [libName, itemName] = parts;
226         const data = this._docs.lib[libName];
227         if(data.hasOwnProperty(itemName)) {
228             const result = data[itemName];
229             if(result) {
230                 result.type = 'lib';
231                 result.category = libName;
232                 return this.jsl.setDepthSafeStringify(result, Infinity);
233             }
234         } else {
235             if(type === 'category') {
236                 for(const category in this._docs) {
237                     const categoryData = this._docs[category];
238                     if(categoryData.hasOwnProperty(name)) {
239                         return this.jsl.setDepthSafeStringify(categoryData[name], 4);
240                     }
241                 }
242             } else {
243                 for(const category in this._docs.global) {
244                     const categoryData = this._docs.global[category];
245                     if(categoryData.hasOwnProperty(name)) {
246                         const result = categoryData[name];
247                         if(result) {
248                             result.type = 'global';
249                             result.category = category;
250                             return this.jsl.setDepthSafeStringify(result, Infinity);
251                         }
252                     }
253                 }
254             }
255         }
256     }
257 }
```

```

251         }
252     }
253   }
254   this.jsl.env.error('@help: ' + language.string(218) + name);
255 }
256
257 /**
258 * Retrieves documentation based on the provided name and type.
259 * @param {string} name - The name of the documentation item.
260 * @param {string} type - The type of the documentation.
261 * @returns {string|undefined} The JSON string of the documentation or
262   undefined if not found.
263 */
264 help(name, type) {
265   return this.helpToJSON(name, type);
266 }
267
268 /**
269 * Retrieves documentation based on the provided name and type.
270 * @param {string} name - The name of the documentation item.
271 * @param {string} type - The type of the documentation.
272 * @returns {string|undefined} The JSON string of the documentation or
273   undefined if not found.
274 */
275 doc(name, type) {
276   return this.help(name, type);
277 }
278
279 /**
280 * Retrieves documentation based on the provided name and type.
281 * @param {string} name - The name of the documentation item.
282 * @param {string} type - The type of the documentation.
283 * @returns {string|undefined} The JSON string of the documentation or
284   undefined if not found.
285 */
286 documentation(name, type) {
287   return this.help(name, type);
288 }
289
290 /**
291 * Searches the documentation for methods that match all words in the given
292 * query, regardless of order.
293 * @param {string} query - The search query containing keywords to match
294 *   within the documentation.
295 * @returns {Array<Object>} Array of matching documentation entries, each
296 *   entry containing 'type' and 'category' properties.
297 */
298 helpSearch(query) {
299   var obj = this;
300   var query_words = query.toLowerCase().split(' ');
301   var results = {};
302   Object.keys(this._docs).forEach(function(type) {
303     Object.keys(obj._docs[type]).forEach(function(function(category) {
304       Object.keys(obj._docs[type][category]).forEach(function(function(member) {
305         var member_obj = obj._docs[type][category][member];
306       });
307     });
308   });
309 }

```

```

300     var str = JSON.stringify(member_obj).toLowerCase();
301     var match = query_words.every((word) => str.includes(word));
302     if(match) {
303         member_obj.type = type;
304         member_obj.category = category;
305         results[member] = member_obj;
306     }
307   });
308 });
309 });
310 return results;
311 }
312
313 /**
314 * Searches the documentation for methods that match all words in the given
315 * query, regardless of order.
316 * @param {string} query - The search query containing keywords to match
317 * within the documentation.
318 * @returns {Array<Object>} Array of matching documentation entries, each
319 * entry containing 'type' and 'category' properties.
320 */
321 docSearch(query) {
322   return this.helpSearch(query);
323 }
324
325 /**
326 * Searches the documentation for methods that match all words in the given
327 * query, regardless of order.
328 * @param {string} query - The search query containing keywords to match
329 * within the documentation.
330 * @returns {Array<Object>} Array of matching documentation entries, each
331 * entry containing 'type' and 'category' properties.
332 */
333 documentationSearch(query) {
334   return this.helpSearch(query);
335 }
336
337 /**
338 * Opens the source file and navigates to the specified line based on the
339 * provided name.
340 * @param {string} name - The name of the source to locate.
341 */
342 source(name) {
343   const parts = name.split(".");
344   if(parts.length === 2) {
345     const [libName, itemName] = parts;
346     const data = this._docs.lib[libName];
347     if(data.hasOwnProperty(itemName)) {
348       const result = data[itemName];
349       if(result) {
350         this.jsl.env.editor(app_path + '/js/sandbox/' + result.
351             source_filename, result.source_lineno);
352       }
353     }
354   }

```

```

347 } else {
348   for (const category in this._docs.global) {
349     const categoryData = this._docs.global[category];
350     if (categoryData.hasOwnProperty(name)) {
351       const result = categoryData[name];
352       if (result) {
353         this.jsl.env.editor(app_path + '/js/sandbox/' + result.
354           source_filename, result.source_lineno);
355         return;
356       }
357     }
358   }
359   this.jsl.env.error('@source: ' + language.string(220) + name);
360 }
361 /**
362 * Showing graph of function.
363 * @param {string} name - The name of the function.
364 */
365 async docGraph(name) {
366   var obj = this;
367   function _docGraph(name) {
368     var result;
369     const parts = name.split('.');
370     if (parts.length === 2) {
371       const [libName, itemName] = parts;
372       const data = obj._docs.lib[libName];
373       if (data.hasOwnProperty(itemName)) {
374         result = data[itemName];
375       }
376     } else {
377       for (const category in obj._docs.global) {
378         const categoryData = obj._docs.global[category];
379         if (categoryData.hasOwnProperty(name)) {
380           result = categoryData[name];
381         }
382       }
383     }
384   }
385
386   var source;
387   var called = new Set();
388   if (result) {
389     if (result.source_range && result.kind === 'function') {
390       source = obj.jsl.file_system.getContentFromCharRange(
391         app_path + '/js/sandbox/' + result.source_filename, result.
392           source_range);
393
394     var ast = obj.jsl.env.recast.parse('function ' + source, {
395       parser: {
396         parse(src) {
397           return obj.jsl.env.babel_parser.parse(src, {
398             sourceType: 'module',
399             allowReturnOutsideFunction: true,
400             plugins: [
401               {
402                 visitorKeys: [
403                   'callExpression'
404                 ],
405                 visitor: {
406                   callExpression(node) {
407                     if (node.callee.type === 'Identifier' && node.callee.name ===
408                       'eval') {
409                       node.value = 'eval';
410                     }
411                   }
412                 }
413               }
414             ]
415           }
416         }
417       }
418     );
419   }
420 }
421 
```

```

400           'jsx',
401           'typescript',
402           'classProperties',
403           'dynamicImport',
404           'optionalChaining',
405           'nullishCoalescingOperator',
406           ],
407       );
408   },
409 },
410 );
411
412 obj.jsl.env.recast.visit(ast, {
413   visitCallExpression(path) {
414     var { callee } = path.node;
415     var name;
416
417     if(callee.type === 'Identifier') {
418       name = callee.name;
419     } else if(
420       callee.type === 'MemberExpression' &&
421       callee.property.type === 'Identifier'
422     ) {
423       name = callee.property.name;
424     }
425
426     if(name) called.add(name);
427     this.traverse(path);
428   },
429 });
430 }
431
432 var lines = [ 'graph TD', '  root["${name}"] '];
433 var id = 0;
434 for(var fn of called) {
435   id = id + 1;
436   lines.push('  root --> id${id}["${fn}"] ');
437 }
438 return lines.join('\n');
439 }
440 obj.jsl.env.error('@docGraph: ' + language.string(218) + name);
441 return false;
442 }
443
444 var graph = _docGraph(name);
445 if(graph) {
446   var graphWin = await this.jsl.windows.showMermaidGraph(graph);
447   graphWin.document.custom_style.textContent += '.node { cursor: pointer; }';
448   graphWin.document.addEventListener('click', async (e) => {
449     var node = e.target.closest('.node');
450     if(!node) return;
451     const labelEl = node.querySelector('.nodeLabel');
452     if(!labelEl) return;
453     const newGraph = _docGraph(labelEl.textContent.trim());

```

```
454         if (new_graph) await graph_win.showGraph(new_graph);
455     });
456 }
457 }
458
459 /**
460 * Retrieves the file name of the currently active JSL script.
461 * @returns {string} The file name of the JSL script.
462 */
463 js1FileName() {
464     return this.js1.jsl_file_name;
465 }
466
467 /**
468 * Clears the application's local storage.
469 */
470 clearStorage() {
471     return this.js1.env.clearStorage();
472 }
473
474 /**
475 * Saves a path to the application's list of saved paths.
476 * @param {string} new_path The path to save.
477 */
478 savePath(new_path) {
479     new_path = this.js1.env.addPathSep(new_path);
480     var i = this.js1.saved_paths.indexOf(new_path);
481     if(i < 0) {
482         this.js1.saved_paths.push(new_path);
483     }
484     this.js1.env.savePath(new_path);
485 }
486
487 /**
488 * Removes a previously saved path from the application's list of saved
489 * paths.
490 * @param {string} saved_path The path to remove.
491 */
492 removePath(saved_path) {
493     saved_path = this.js1.env.addPathSep(saved_path);
494     var i = this.js1.saved_paths.indexOf(saved_path);
495     if(i >= 0) {
496         this.js1.saved_paths.splice(i, 1);
497     }
498     this.js1.env.removePath(saved_path);
499 }
500
501 /**
502 * Opens the help documentation for CMD window.
503 */
504 _cmd_help() {
505     this.js1.env.cmd_help();
506     this.js1.no_ans = true;
507     this.js1.ignore_output = true;
508 }
```

```
508
509  /**
510  * Displays application info.
511  */
512 _info() {
513     this.jsl.env.info();
514     this.jsl.no_ans = true;
515     this.jsl.ignore_output = true;
516 }
517
518 /**
519 * Opens the settings menu.
520 */
521 _settings() {
522     this.jsl.env.settings();
523     this.jsl.no_ans = true;
524     this.jsl.ignore_output = true;
525 }
526
527 /**
528 * Retrieves the current working directory.
529 * @returns {string} The current working directory.
530 */
531 _pwd() {
532     return this.jsl.current_path;
533 }
534
535 /**
536 * Changes the current working directory to the specified path.
537 * @param {string} new_path The new path to set as the current working
538 *   directory.
539 */
540 cd(new_path) {
541     this.jsl.setPath(new_path);
542     this.jsl.env.cd(new_path);
543     this.jsl.no_ans = true;
544     this.jsl.ignore_output = true;
545 }
546
547 /**
548 * Lists the contents of the current directory.
549 * @returns {Array<string>} An array of filenames in the current directory.
550 */
551 _ls() {
552     return this.jsl.env.listFolderContents();
553 }
554
555 /**
556 * Retrieves the application version.
557 * @returns {string} The application version.
558 */
559 _version() {
560     return this.jsl.env.version;
561 }
```

```
562  /**
563   * Retrieves the operating system platform.
564   * @returns {string} The OS platform.
565   */
566   _platform() {
567     return this.jsl.env.platform;
568   }
569
570  /**
571   * Retrieves if the JSLAB application is currently in debug mode.
572   * @returns {boolean} True if the application is in debug mode; otherwise,
573   *                 false.
574   */
575   _debug_flag() {
576     return this.jsl.env.debug;
577   }
578
579  /**
580   * Retrieves the current workspace.
581   * @returns {Object} The current workspace object.
582   */
583   workspace() {
584     return this.jsl.getWorkspace();
585   }
586
587  /**
588   * Updates the workspace display based on the current state.
589   */
590   updateWorkspace() {
591     this.jsl.env.updateWorkspace();
592     this.jsl.no_ans = true;
593     this.jsl.ignore_output = true;
594   }
595
596  /**
597   * Updates the file browser display based on the current state.
598   */
599   updateFileBrowser() {
600     this.jsl.env.updateFileBrowser();
601     this.jsl.no_ans = true;
602     this.jsl.ignore_output = true;
603   }
604
605  /**
606   * Clears the workspace.
607   */
608   _clear() {
609     this.jsl.clear();
610   }
611
612  /**
613   * Clears the command window.
614   */
615   _clc() {
616     this.jsl.env.clc();
```

```
616     this.jsl.no_ans = true;
617     this.jsl.ignore_output = true;
618 }
619 /**
620 * Displays an error message.
621 * @param {string} msg The error message to display.
622 */
623 error(msg) {
624     this.jsl.env.error(msg);
625     this.jsl.no_ans = true;
626     this.jsl.ignore_output = true;
627 }
628
629 /**
630 * Displays a general message.
631 * @param {string} msg The message to display.
632 */
633 disp(msg) {
634     this.jsl.env.disp(msg);
635     this.jsl.no_ans = true;
636     this.jsl.ignore_output = true;
637     return msg+"\n";
638 }
639
640 /**
641 * Displays a general message with monospaced font.
642 * @param {string} msg The message to display.
643 */
644 dispMonospaced(msg) {
645     this.jsl.env.dispMonospaced(msg);
646     this.jsl.no_ans = true;
647     this.jsl.ignore_output = true;
648     return msg+"\n";
649 }
650
651 /**
652 * Displays a warning message.
653 * @param {string} msg The warning message to display.
654 */
655 warn(msg) {
656     this.jsl.env.warn(msg);
657     this.jsl.no_ans = true;
658     this.jsl.ignore_output = true;
659 }
660
661 /**
662 * Opens a specified file in the editor or just opens the editor if no file
663 * is specified.
664 * @param {string} [filepath] The path to the file to open in the editor.
665 */
666 _editor(filepath) {
667     this.jsl.env.editor(filepath);
668     this.jsl.no_ans = true;
669     this.jsl.ignore_output = true;
```

```

670     }
671
672     /**
673      * Logs a point in the script for debugging purposes.
674      */
675     _logpoint() {
676         this.jsl.env.setWorkspace();
677         this.checkStopLoop();
678         this.jsl.onStopLoop();
679     }
680
681     /**
682      * Updates the workspace or environment at a specific point during script
683      * execution.
684      */
685     _updatepoint() {
686         this._logpoint();
687     }
688
689     /**
690      * Checks whether script execution should be stopped at the current point.
691      */
692     _checkStop() {
693         this._logpoint();
694     }
695
696     /**
697      * Introduces a breakpoint in the script, pausing execution and potentially
698      * allowing the user to continue based on their input.
699      */
700     _breakpoint() {
701         this._logpoint();
702
703         var [line, column, script] = this.jsl.eval.getExpressionPosition();
704         var ret = this.jsl.env.showMessageBox({ title: language.currentString(15),
705             message: language.currentString(216)+': '+line+', '+language.
706             currentString(113)+': '+column+' ('+script+') '+language.
707             currentString(213), buttons: [language.currentString(214), language.
708             currentString(215)], cancelId: 0});
709
710         if(ret == 1) {
711             this.jsl.stop_loop = true;
712             this.jsl.onStopLoop();
713         }
714     }
715
716     /**
717      * Introduces a breakpoint in the script, pausing execution and potentially
718      * allowing the user to continue based on their input.
719      */
720     _debug() {
721         this._breakpoint();
722     }
723
724     /**
725      * Pauses script execution, providing an opportunity to inspect the current

```

```
    state or continue execution based on user input.

718  */
719 _pause() {
720     this._logpoint();

721
722     var [line, column, script] = this.jsl.eval.getExpressionPosition();
723     var ret = this.jsl.env.showMessageBox({ title: language.currentString(15),
724         message: language.currentString(217)+': '+line+', '+language.
725         currentString(113)+': '+column+' ('+script+') . '+language.
726         currentString(213), buttons: [language.currentString(214), language.
727         currentString(215)], cancelId: 0});

728     if(ret == 1) {
729         this.jsl.stop_loop = true;
730         this.jsl.onStopLoop();
731     }
732
733 /**
734 * Forces the script execution to stop at a designated point.
735 */
736 _stoppoint() {
737     this.jsl.stop_loop = true;
738     this.jsl.onStopLoop();
739
740 /**
741 * Marks an endpoint in the script, throwing an error to signify a critical
742 * stop or exit point in execution.
743 */
744 _endPoint() {
745     var [line, column, script] = this.jsl.eval.getExpressionPosition();
746     throw {name: 'JslabError', message: language.string(116)+': '+line+', '+
747         language.string(113)+': '+column+' ('+script+') . '};

748 /**
749 * Verifies if a loop within the script execution should be terminated,
750 * typically used to avoid infinite or lengthy unnecessary execution.
751 */
752 checkStopLoop() {
753     if(!this.jsl.stop_loop) {
754         this.jsl.stop_loop = this.jsl.env.checkStopLoop();
755     }
756     return this.jsl.stop_loop;
757
758 /**
759 * Opens a specified file in an editor or opens the editor to a default or
760 * previously specified file.
761 * @param {string} [filepath] - Path to the file to be opened in the editor.
762 */
763 edit(filepath) {
764     _editor(filepath);
765 }
```

```

764
765  /**
766   * Returns a list of all example scripts available within a predefined
767   * directory.
768   * @return {Array<string>} An array of paths to the example scripts.
769   */
770 getExamples() {
771   var obj = this;
772   return this.jsl.env.readDir(app_path + '/examples')
773     .filter(function(file) { return file.match(new RegExp('.jsl
774 % -----
775 % End of document
776 %
777 \end{document})); }).map(function(i) { return folder + '\\\\' + i; });
778 }
779
780 /**
781 * Opens a specified example script in the editor window.
782 * @param {string} filename - Name of the example file to open.
783 */
784 openExample(filename) {
785   if (!this.jsl.env.pathIsAbsolute(filename)) {
786     filename = app_path + '\\\\examples\\\\' + filename;
787   }
788   this.edit(filename);
789 }
790
791 /**
792 * Opens examples folder in File Explorer
793 */
794 openExamplesFolder() {
795   this.jsl.env.openFolder(app_path + '\\\\examples');
796 }
797
798 /**
799 * Opens examples folder in File Explorer
800 */
801 goToExamplesFolder() {
802   this.jsl.env.cd(app_path + '\\\\examples');
803 }
804
805 /**
806 * Displays a synchronous message box to the user and waits for their
807 * response.
808 * @param {Object} options - Configuration options for the message box.
809 * @return {number} The index of the button clicked by the user.
810 */
811 showMessageBox(options) {
812   return this.jsl.env.showMessageBox(options);
813 }
814
815 /**
816 * Saves specified variables to a JSON file.
817 * @param {string} file_path - Path where the JSON file will be saved.

```

```

817 * @param {... string} args - Variables to save. If 'all' is specified , saves
818     all available variables.
819 */
820 save(file_path, ... args) {
821     var obj = this;
822     var vars = {};
823     if(args.length == 0 || (args.length == 1 && args[0] == 'all')) {
824         var properties = this.jsl.getWorkspaceProperties();
825         properties.forEach(function(property) {
826             vars[property] = obj.jsl.context[property];
827         });
828     } else {
829         args.forEach(function(property) {
830             if(obj.jsl.context.hasOwnProperty(property)) {
831                 vars[property] = obj.jsl.context[property];
832             }
833         });
834     }
835     if(!this.jsl.env.pathIsAbsolute(file_path)) {
836         file_path = this.jsl.env.pathJoin(this.jsl.current_path, file_path);
837     }
838     var flag = this.jsl.env.writeFileSync(file_path, JSON.stringify(vars));
839     if(flag === false) {
840         this.jsl.env.error('@save: '+language.string(117)+': ' + file_path);
841     }
842 }
843 /**
844 * Loads variables from a specified JSON file into the specified scope or
845     the default script context.
846 * If an error occurs during file reading or parsing, it logs an error
847     message.
848 * @param {...*} args - A single filename or a scope and filename to specify
849     where to load the variables.
850 */
851 load(... args) {
852     var obj = this;
853     var scope = obj.jsl.context;
854     var file_path;
855     if(args.length > 1) {
856         scope = args[0];
857         file_path = args[1];
858     } else {
859         file_path = args[0];
860     }
861     file_path = this.jsl.pathResolve(file_path);
862     if(file_path) {
863         try {
864             var vars = JSON.parse(readFile(file_path));
865             Object.keys(vars).forEach(function(property) {
866                 scope[property] = vars[property];
867             });
868         } catch(err) {
869             this.jsl.env.error('@load: '+language.string(118)+'.');
870         }
871     }
872 }

```

```

868         }
869     }
870   }
871
872 /**
873 * Executes a system shell command.
874 * @param {...*} arg - The command and its arguments to be executed.
875 * @return {string} The output of the executed command.
876 */
877 system (...arg) {
878   try {
879     return this.jsl.env.execSync(...arg).toString();
880   } catch(err) {
881     return err.message + ', command output: \n' + err.stdout.toString();
882   }
883 }
884
885 /**
886 * Retrieves completion suggestions based on the current context and input.
887 * @param {Array} data - Data containing the start of the string to complete
888 * , context, and keywords.
889 * @return {Array<string>} An array of completion suggestions.
890 */
891 getCompletions(data) {
892   var start = data[0];
893   var context = data[1];
894   var keywords = data[2];
895   var found = [] , global = window;
896
897   function maybeAdd(str) {
898     if(str.lastIndexOf(start, 0) == 0 && !arrayContains(found, str)) found.push(str);
899   }
900   function gatherCompletions(obj) {
901     if(typeof obj == "string") forEach(("charAt charCodeAt indexOf
902       lastIndexOf substring substr slice trim trimLeft trimRight "
903       "toUpperCase toLowerCase split concat match replace
904       search").split(" "), maybeAdd);
905     else if(obj instanceof Array) forEach(("length concat join splice push
906       pop shift unshift slice reverse sort indexOf "
907       "lastIndexOf every some filter forEach map reduce
908       reduceRight").split(" "), maybeAdd);
909     else if(obj instanceof Function) forEach("prototype apply call bind".
910       split(" "), maybeAdd);
911     if(!Object.getOwnPropertyNames || !Object.getPrototypeOf) {
912       for(var name in obj) maybeAdd(name);
913     } else {
914       for(var o = obj; o; o = Object.getPrototypeOf(o))
915         Object.getOwnPropertyNames(o).forEach(maybeAdd);
916     }
917   }
918
919   if(context && context.length) {
920     context = JSON.parse(context);
921     // If this is a property, see if it belongs to some object we can

```

```

916     // find in the current environment.
917     var obj = context.pop(), base;
918     if (obj.type && obj.type.indexOf("variable") === 0) {
919         base = base || global[obj.string];
920     } else if (obj.type === "string") {
921         base = "";
922     } else if (obj.type === "atom") {
923         base = 1;
924     } else if (obj.type === "function") {
925         if (global.jQuery !== null && (obj.string === '
926
927 % -----
928 % End of document
929 %
930 \end{document} || obj.string === 'jQuery') &&
931     (typeof global.jQuery === 'function')) {
932         base = global.jQuery();
933     } else if (global._ !== null && (obj.string === '_') && (typeof global.
934         _ === 'function')) {
935         base = global._();
936     }
937     while (base !== null && context.length)
938         base = base[context.pop().string]; // switch base to variable
939     if (base !== null) gatherCompletions(base);
940 } else {
941     gatherCompletions(global);
942     forEach(keywords, maybeAdd);
943 }
944 if (found.length) {
945     found.sort(function(a, b) {
946         return a.length - b.length || a.localeCompare(b);
947     });
948 }
949 return found;
950 }

951 /**
952 * Retrieves an object by matching a specific property value.
953 * @param {Object} obj - The object to search through.
954 * @param {string} prop - The property name to match.
955 * @param {*} val - The value to match against the property.
956 * @returns {Object|null} The found object with key and value, or null if
957     not found.
958 */
959 getObjectByProp(obj, prop, val) {
960     const key = Object.keys(obj).find(function(key) { return obj[key][prop]
961         === val; });
962     return key ? { key, value: obj[key] } : null;
963 }

964 /**
965 * Retrieves multiple objects from a parent object by matching a specific
966     property value.
967 * @param {Object} obj - The parent object to search through.

```

```

967     * @param {string} prop - The property name to match.
968     * @param {*} val - The value to match against the property.
969     * @returns {Object} An object containing all matched key-value pairs.
970     */
971     getObjectsByProp(obj, prop, val) {
972         return Object.fromEntries(Object.entries(obj)
973             .filter(function([key, value]) { return value[prop] === val; }));
974     }
975
976     /**
977      * Compares two strings lexicographically.
978      * @param {string} x - The first string.
979      * @param {string} y - The second string.
980      * @return {number} The result of the comparison.
981      */
982     strcmp(x, y) {
983         return this.jsl.env.math.compareText(x, y);
984     }
985
986     /**
987      * Compare two version strings (e.g. "1.4.2", "v2.0.0-beta.1").
988      * @param {string} a First version.
989      * @param {string} b Second version.
990      * @returns {number} -1 if a < b, 0 if equal, 1 if a > b.
991      */
992     compareVersions(a, b) {
993         var clean = v => v.replace(/^v/i, '').split(/[-+]/)[0];
994         var toNums = v => clean(v).split('.').map(Number);
995
996         var x = toNums(a);
997         var y = toNums(b);
998         const len = Math.max(x.length, y.length);
999
1000        for(let i = 0; i < len; i++) {
1001            var xi = x[i] ?? 0;
1002            var yi = y[i] ?? 0;
1003            if (xi > yi) return 1;
1004            if (xi < yi) return -1;
1005        }
1006        return 0;
1007    }
1008
1009    /**
1010     * Checks if there is available update
1011     * @returns {boolean} True if there is available update; otherwise, false.
1012     */
1013     async checkForUpdate() {
1014         if(this.jsl.networking.isOnline()) {
1015             try {
1016                 var api_base = 'https://api.github.com/repos/PR-DC/JSLAB';
1017
1018                 const rel = await fetch(`/${api_base}/releases/latest`, {
1019                     headers: { 'Accept': 'application/vnd.github+json' }
1020                 });
1021             }
1022         }
1023     }

```

```

1022     var latest_version;
1023     if (rel.ok) {
1024         const { tag_name } = await rel.json(); // e.g. "v1.5.0"
1025         latest_version = tag_name;
1026     } else if (rel.status === 404) {
1027         const tagRes = await fetch(`/${api_base}/tags?per_page=1`);
1028         const [ { name } ] = await tagRes.json(); // e.g. "v1.5.0-beta.1"
1029         latest_version = name;
1030     } else {
1031         this.jsl._console.log(rel);
1032         this.jsl.env.error('@checkForUpdate: '+language.string(237));
1033     }
1034
1035     var check = this.compareVersions(this.jsl.env.version, latest_version)
1036         === -1;
1037     if (check) {
1038         this.jsl.env disp('@checkForUpdate: '+language.string(238) + '<a'
1039             href="https://github.com/PR-DC/JSLAB/releases" class="external-"
1040             link">https://github.com/PR-DC/JSLAB/releases</a>');
1041     }
1042     return check;
1043 } catch(err) {
1044     this.jsl._console.log(err);
1045     this.jsl.env.error('@checkForUpdate: '+language.string(237));
1046 }
1047 }
1048 /**
1049 * Unloads a previously required module from the cache.
1050 * @param {string} module - The module to unrequire.
1051 */
1052 unrequire(module) {
1053     module = this.jsl.pathResolve(module);
1054     if (this.jsl.required_modules.includes(module)) {
1055         var name = require.resolve(module);
1056         if (name) {
1057             delete require.cache[name];
1058         }
1059         this.jsl.array.removeElementByValue(this.jsl.required_modules, module);
1060     }
1061 }
1062 }
1063 /**
1064 * Resets app.
1065 */
1066 resetApp() {
1067     this.jsl.env.resetApp();
1068 }
1069 }
1070 /**
1071 * Resets the sandbox environment to its initial state.
1072 */
1073

```

```
1074     resetSandbox() {
1075         this.jsl.env.resetSandbox();
1076     }
1077
1078     /**
1079      * Opens the developer tools for the sandbox environment in the current
1080      * context.
1081      * @returns {void}
1082      */
1083     openDevTools() {
1084         this.jsl.env.openSandboxDevTools();
1085     }
1086
1087     /**
1088      * Compiles a N-API module located at the specified path.
1089      * @param {string} path - The path to the N-API module.
1090      * @param {boolean} [show_output=true] - Whether to show output in the
1091      * command window.
1092      * @return {Array} An array containing the result of the compilation and
1093      * targets.
1094      */
1095     compileNapi(path, show_output = false) {
1096         var result = false;
1097         var obj = this;
1098         if(typeof path == 'string') {
1099             path = this.jsl.pathResolve(path);
1100         }
1101         if(!path) {
1102             var options = {
1103                 title: language.currentString(141),
1104                 buttonLabel: language.currentString(142),
1105                 properties: [ 'openDirectory' ],
1106             };
1107             path = this.jsl.env.showOpenDialogSync(options);
1108             if(path == undefined) {
1109                 this.jsl.env.error('@compileNapi: '+language.string(119)+'.');
1110                 return false;
1111             } else {
1112                 path = path[0];
1113             }
1114         }
1115         path = this.jsl.env.addPathSep(path);
1116
1117         if(this.jsl.env.rmSync(path+'build/Release/', false) == false) {
1118             this.jsl.env.error('@compileNapi: '+language.string(171));
1119         }
1120
1121         var binding_file_path = path + 'binding.gyp';
1122         if(this.jsl.env.checkFile(binding_file_path)) {
1123             var targets = [];
1124             try {
1125                 var binding_file_data = JSON.parse(this.jsl.env.readFileSync(
1126                     binding_file_path).toString());
1127             } catch(err) {
1128                 this.jsl.env.error('@compileNapi: '+language.string(120)+'.');
```

```

1125         return false;
1126     }
1127     binding_file_data.targets.forEach(function(target) {
1128       targets.push(path + 'build/Release/' + target.target_name + '.node');
1129     });
1130     if(targets.length > 0) {
1131       var exe = this.jsl.env.exe_path;
1132       var node_gyp_path = app_path + '/node_modules/node-gyp/bin/node-gyp.js';
1133       var npm_path = this.jsl.env.pathJoin(app_path, 'node_modules', 'npm',
1134                                         'bin', 'npm-cli.js');
1135       var msg = this.system('set ELECTRON_RUN_AS_NODE=1 &"' + exe + '" "' +
1136                             npm_path + '" cache clean --force &"' + exe + '" "' +
1137                             npm_path + '" install --build-from-source=false &"' + exe + '" "' +
1138                             node_gyp_path + '" rebuild --target=' + process.version + ' 2>&1', {
1139       cwd: path, shell: false});
1140
1141     if(msg.endsWith('gyp info ok \n')) {
1142       if(show_output) {
1143         this.jsl.env.disp(msg.replaceAll('\n', '<br>'));
1144       }
1145       result = true;
1146     } else if(!msg.endsWith('gyp ERR! not ok \n')) {
1147       this.jsl.env.error('@compileNapi: ' + language.string(121) + ' ' + msg.
1148                          replaceAll('\n', '<br>'));
1149     } else {
1150       this.jsl.env.error('@compileNapi: ' + language.string(170) + ' ' + msg.
1151                          replaceAll('\n', '<br>'));
1152     }
1153
1154     if(result) {
1155       return [result, targets];
1156     } else {
1157       return [result, undefined];
1158     }
1159   }
1160 }
1161 /**
1162 * Installs a module located at the specified path.
1163 * @param {string} path - The path to the module.
1164 * @param {boolean} [show_output=true] - Whether to show output in the
1165   command window.
1166 */
1167 installModule(path, show_output = false) {
1168   if(typeof path === 'string') {

```

```

1169     path = this.jsl.pathResolve(path);
1170 }
1171 if(!path) {
1172     var options = {
1173         title: language.currentString(141),
1174         buttonLabel: language.currentString(142),
1175         properties: [ 'openDirectory' ],
1176     };
1177     path = this.jsl.env.showOpenDialogSync(options);
1178     if(path === undefined) {
1179         this.jsl.env.error('@installModule: '+language.string(119)+'.');
1180     } else {
1181         path = path[0];
1182     }
1183 }
1184 path = this.jsl.env.addPathSep(path);
1185
1186 var exe = this.jsl.env.exe_path;
1187 var npm_path = this.jsl.env.pathJoin(app_path, 'node_modules', 'npm', 'bin',
1188     , 'npm-cli.js');
1189 var msg = this.system('set ELECTRON_RUN_AS_NODE=1 & "' + exe + '" "' +
1190     npm_path + '" install 2>&1', { cwd: path, shell: false });
1191
1192 if(!msg.includes('\nnpm error')) {
1193     if(show_output) {
1194         this.jsl.env.disp(msg);
1195     } else {
1196         this.jsl.env.error('@installModule: '+msg.replaceAll('\n', '<br>'));
1197     }
1198 }
1199 /**
1200 * Registers an object for cleanup with a specified cleanup function.
1201 * @param {Object} obj - The object to be registered for cleanup.
1202 * @param {Function} fun - The function to execute during cleanup.
1203 */
1204 addForCleanup(...args) {
1205     this.jsl.addForCleanup(...args);
1206 }
1207 }
1208
1209 exports.PRDC_JSLAB_LIB_BASIC = PRDC_JSLAB_LIB_BASIC;

```

Listing 85 - basic.js

```

1 /**
2 * @file JSLAB library color submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB color submodule.
10 */

```



```
11 class PRDC_JSLAB_LIB_COLOR {
12
13     /**
14      * Constructs a color submodule object with access to JSLAB's color
15      * functions.
16      * @constructor
17      * @param {Object} js1 - Reference to the main JSLAB object.
18      */
19     constructor(js1) {
20         var obj = this;
21         this.js1 = js1;
22
23         /**
24          * Color order
25          * @type {Array}
26          */
27         this.colororder = ['#0072BD', '#D95319', '#EDB120', '#E2F8E', '#77AC30',
28                           '#4DBEEE', '#A2142F'];
29
30         var colors = {
31             alliceblue: 'rgb(240, 248, 255)',
32             antiquewhite: 'rgb(250, 235, 215)',
33             aqua: 'rgb(0, 255, 255)',
34             aquamarine: 'rgb(127, 255, 212)',
35             azure: 'rgb(240, 255, 255)',
36             beige: 'rgb(245, 245, 220)',
37             bisque: 'rgb(255, 228, 196)',
38             black: 'rgb(0, 0, 0)',
39             blanchedalmond: 'rgb(255, 235, 205)',
40             blue: 'rgb(0, 0, 255)',
41             blueviolet: 'rgb(138, 43, 226)',
42             brown: 'rgb(165, 42, 42)',
43             burlywood: 'rgb(222, 184, 135)',
44             cadetblue: 'rgb(95, 158, 160)',
45             chartreuse: 'rgb(127, 255, 0)',
46             chocolate: 'rgb(210, 105, 30)',
47             coral: 'rgb(255, 127, 80)',
48             cornflowerblue: 'rgb(100, 149, 237)',
49             cornsilk: 'rgb(255, 248, 220)',
50             crimson: 'rgb(220, 20, 60)',
51             cyan: 'rgb(0, 255, 255)',
52             darkblue: 'rgb(0, 0, 139)',
53             darkcyan: 'rgb(0, 139, 139)',
54             darkgoldenrod: 'rgb(184, 134, 11)',
55             darkgray: 'rgb(169, 169, 169)',
56             darkgreen: 'rgb(0, 100, 0)',
57             darkgrey: 'rgb(169, 169, 169)',
58             darkkhaki: 'rgb(189, 183, 107)',
59             darkmagenta: 'rgb(139, 0, 139)',
60             darkolivegreen: 'rgb(85, 107, 47)',
61             darkorange: 'rgb(255, 140, 0)',
62             darkorchid: 'rgb(153, 50, 204)',
63             darkred: 'rgb(139, 0, 0)',
64             darksalmon: 'rgb(233, 150, 122)',
65             darkseagreen: 'rgb(143, 188, 143)',
```

```
64 darkslateblue: 'rgb(72, 61, 139)',  
65 darkslategray: 'rgb(47, 79, 79)',  
66 darkslategrey: 'rgb(47, 79, 79)',  
67 darkturquoise: 'rgb(0, 206, 209)',  
68 darkviolet: 'rgb(148, 0, 211)',  
69 deeppink: 'rgb(255, 20, 147)',  
70 deepskyblue: 'rgb(0, 191, 255)',  
71 dimgray: 'rgb(105, 105, 105)',  
72 dimgrey: 'rgb(105, 105, 105)',  
73 dodgerblue: 'rgb(30, 144, 255)',  
74 firebrick: 'rgb(178, 34, 34)',  
75 floralwhite: 'rgb(255, 250, 240)',  
76 forestgreen: 'rgb(34, 139, 34)',  
77 fuchsia: 'rgb(255, 0, 255)',  
78 gainsboro: 'rgb(220, 220, 220)',  
79 ghostwhite: 'rgb(248, 248, 255)',  
80 gold: 'rgb(255, 215, 0)',  
81 goldenrod: 'rgb(218, 165, 32)',  
82 gray: 'rgb(128, 128, 128)',  
83 green: 'rgb(0, 128, 0)',  
84 greenyellow: 'rgb(173, 255, 47)',  
85 grey: 'rgb(128, 128, 128)',  
86 honeydew: 'rgb(240, 255, 240)',  
87 hotpink: 'rgb(255, 105, 180)',  
88 indianred: 'rgb(205, 92, 92)',  
89 indigo: 'rgb(75, 0, 130)',  
90 ivory: 'rgb(255, 255, 240)',  
91 khaki: 'rgb(240, 230, 140)',  
92 lavender: 'rgb(230, 230, 250)',  
93 lavenderblush: 'rgb(255, 240, 245)',  
94 lawngreen: 'rgb(124, 252, 0)',  
95 lemonchiffon: 'rgb(255, 250, 205)',  
96 lightblue: 'rgb(173, 216, 230)',  
97 lightcoral: 'rgb(240, 128, 128)',  
98 lightcyan: 'rgb(224, 255, 255)',  
99 lightgoldenrodyellow: 'rgb(250, 250, 210)',  
100 lightgray: 'rgb(211, 211, 211)',  
101 lightgreen: 'rgb(144, 238, 144)',  
102 lightgrey: 'rgb(211, 211, 211)',  
103 lightpink: 'rgb(255, 182, 193)',  
104 lightsalmon: 'rgb(255, 160, 122)',  
105 lightseagreen: 'rgb(32, 178, 170)',  
106 lightskyblue: 'rgb(135, 206, 250)',  
107 lightslategray: 'rgb(119, 136, 153)',  
108 lightslategrey: 'rgb(119, 136, 153)',  
109 lightsteelblue: 'rgb(176, 196, 222)',  
110 lightyellow: 'rgb(255, 255, 224)',  
111 lime: 'rgb(0, 255, 0)',  
112 limegreen: 'rgb(50, 205, 50)',  
113 linen: 'rgb(250, 240, 230)',  
114 magenta: 'rgb(255, 0, 255)',  
115 maroon: 'rgb(128, 0, 0)',  
116 mediumaquamarine: 'rgb(102, 205, 170)',  
117 mediumblue: 'rgb(0, 0, 205)',  
118 mediumorchid: 'rgb(186, 85, 211)',
```

```
119 mediumpurple: 'rgb(147, 112, 219)',  
120 mediumseagreen: 'rgb(60, 179, 113)',  
121 mediumslateblue: 'rgb(123, 104, 238)',  
122 mediumspringgreen: 'rgb(0, 250, 154)',  
123 mediumturquoise: 'rgb(72, 209, 204)',  
124 mediumvioletred: 'rgb(199, 21, 133)',  
125 midnightblue: 'rgb(25, 25, 112)',  
126 mintcream: 'rgb(245, 255, 250)',  
127 mistyrose: 'rgb(255, 228, 225)',  
128 moccasin: 'rgb(255, 228, 181)',  
129 navajowhite: 'rgb(255, 222, 173)',  
130 navy: 'rgb(0, 0, 128)',  
131 oldlace: 'rgb(253, 245, 230)',  
132 olive: 'rgb(128, 128, 0)',  
133 olivedrab: 'rgb(107, 142, 35)',  
134 orange: 'rgb(255, 165, 0)',  
135 orangered: 'rgb(255, 69, 0)',  
136 orchid: 'rgb(218, 112, 214)',  
137 palegoldenrod: 'rgb(238, 232, 170)',  
138 palegreen: 'rgb(152, 251, 152)',  
139 paleturquoise: 'rgb(175, 238, 238)',  
140 palevioletred: 'rgb(219, 112, 147)',  
141 papayawhip: 'rgb(255, 239, 213)',  
142 peachpuff: 'rgb(255, 218, 185)',  
143 peru: 'rgb(205, 133, 63)',  
144 pink: 'rgb(255, 192, 203)',  
145 plum: 'rgb(221, 160, 221)',  
146 powderblue: 'rgb(176, 224, 230)',  
147 purple: 'rgb(128, 0, 128)',  
148 red: 'rgb(255, 0, 0)',  
149 rosybrown: 'rgb(188, 143, 143)',  
150 royalblue: 'rgb(65, 105, 225)',  
151 saddlebrown: 'rgb(139, 69, 19)',  
152 salmon: 'rgb(250, 128, 114)',  
153 sandybrown: 'rgb(244, 164, 96)',  
154 seagreen: 'rgb(46, 139, 87)',  
155 seashell: 'rgb(255, 245, 238)',  
156 sienna: 'rgb(160, 82, 45)',  
157 silver: 'rgb(192, 192, 192)',  
158 skyblue: 'rgb(135, 206, 235)',  
159 slateblue: 'rgb(106, 90, 205)',  
160 slategray: 'rgb(112, 128, 144)',  
161 slategrey: 'rgb(112, 128, 144)',  
162 snow: 'rgb(255, 250, 250)',  
163 springgreen: 'rgb(0, 255, 127)',  
164 steelblue: 'rgb(70, 130, 180)',  
165 tan: 'rgb(210, 180, 140)',  
166 teal: 'rgb(0, 128, 128)',  
167 thistle: 'rgb(216, 191, 216)',  
168 tomato: 'rgb(255, 99, 71)',  
169 turquoise: 'rgb(64, 224, 208)',  
170 violet: 'rgb(238, 130, 238)',  
171 wheat: 'rgb(245, 222, 179)',  
172 white: 'rgb(255, 255, 255)',  
173 whitesmoke: 'rgb(245, 245, 245)',
```

```

174     yellow: 'rgb(255, 255, 0)',  

175     yellowgreen: 'rgb(154, 205, 50)',  

176   };  

177  

178   this.RE_RGB = /^rgb\(\s*(\d+)\s*,\s*(\d+)\s*,\s*(\d+)\s*\)$/;  

179   this.RE_RGBA = /^rgba\(\s*(\d+)\s*,\s*(\d+)\s*,\s*(\d+)\s*,\s*([\d\.]+)\s*\$/  

180   this.RE_HSL = /^hsl\(\s*([\d\.]+)\s*,\s*([\d\.]+)%\s*,\s*([\d\.]+)%\s*\)$/  

181   this.RE_HSLA = /^hsla\(\s*([\d\.]+)\s*,\s*([\d\.]+)%\s*,\s*([\d\.]+)%\s*,\s*([\d\.]+)\s*\$/  

182   this.RE_HEX = /^([0-9a-fA-F]{2})([0-9a-fA-F]{2})([0-9a-fA-F]{2})$/; // 6  

183   digit  

184  

185   // Global object  

186   var Color = {  

187     create: function(str) {  

188       str = str.replace(/\s*#/|\s*$/g, '');  

189       str = str.toLowerCase();  

190       if(KEYWORDS[str]) str = KEYWORDS[str];  

191  

192       var match;  

193  

194       // RGB(A)  

195       if((match = str.match(RE_RGB) || str.match(RE_RGBA))) {  

196         return new Color.RGBA(  

197           parseInt(match[1], 10),  

198           parseInt(match[2], 10),  

199           parseInt(match[3], 10),  

200           parseFloat(match.length === 4 ? 1 : match[4]))  

201       }  

202  

203       // HSL(A)  

204       if((match = str.match(RE_HSL) || str.match(RE_HSLA))) {  

205         return new Color.HSLA(  

206           parseFloat(match[1]),  

207           parseFloat(match[2]),  

208           parseFloat(match[3]),  

209           parseFloat(match.length === 4 ? 1 : match[4]))  

210       }  

211  

212       // Hex  

213       if(str.length === 3) {  

214         // Hex 3 digit -> 6 digit  

215         str = str.replace(/(.) /g, '$1$1$1');  

216       }  

217       if((match = str.match(RE_HEX))) {  

218         return new Color.RGBA(  

219           parseInt(match[1], 16),  

220           parseInt(match[2], 16),  

221           parseInt(match[3], 16),  

222           1  

223         );  

224

```

```

225     }
226
227     return null;
228 },
229
230   luminance: function(color) {
231     if(color instanceof Color.HSLA) color = color.toRGBA();
232     return 0.298912 * color.r + 0.586611 * color.g + 0.114478 * color.b;
233   },
234
235   greyscale: function(color) {
236     var lum = Color.luminance(color);
237     return new Color.RGBA(lum, lum, lum, this.a);
238   },
239
240   nagate: function(color) {
241     if(color instanceof Color.HSLA) color = color.toRGBA();
242     return new Color.RGBA(255 - color.r, 255 - color.g, 255 - color.b,
243                           color.a);
244   },
245
246 /**
247  * @see http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html#mix-
248  * instance_method
249 */
250 mix: function(color1, color2, weight) {
251   if(color1 instanceof Color.HSLA) color1 = color1.toRGBA();
252   if(color2 instanceof Color.HSLA) color2 = color2.toRGBA();
253   if(isNull(weight) || obj.jsl.format.isDefined(weight)) weight = 0.5;
254
255   var w0 = 1 - weight;
256   var w = w0 * 2 - 1;
257   var a = color1.a - color2.a;
258   var w1 = ((w * a === -1 ? w : (w + a) / (1 + w * a)) + 1) / 2;
259   var w2 = 1 - w1;
260
261   return new Color.RGBA(
262     Math.round(color1.r * w1 + color2.r * w2),
263     Math.round(color1.g * w1 + color2.g * w2),
264     Math.round(color1.b * w1 + color2.b * w2),
265     Math.round(color1.a * w0 + color2.a * weight)
266   );
267 }
268 /**
269  * Color.RGBA
270 */
271 Color.RGBA = function(r, g, b, a) {
272   if.isArray(r)) {
273     g = r[1];
274     b = r[2];
275     a = r[3];
276     r = r[0];
277   } else ifisObject(r)) {

```

```

278         g = r.g;
279         b = r.b;
280         a = r.a;
281         r = r.r;
282     }
283
284     this.r = r || 0;
285     this.g = g || 0;
286     this.b = b || 0;
287     this.a = !isNumber(a) ? 1 : a;
288 };
289
290 Color.RGBA.prototype = {
291     toHSLA: function() {
292         var hsl = rgbToHsl(Math.round(this.r), Math.round(this.g), Math.round(
293             this.b));
294         return new Hsla(hsl[0], hsl[1], hsl[2], this.a);
295     },
296     toArray: function() {
297         return [Math.round(this.r), Math.round(this.g), Math.round(this.b),
298             this.a];
299     },
300     clone: function() {
301         return new Color.RGBA(this);
302     },
303     toString: function() {
304         return 'rgba(' + Math.round(this.r) + ', ' + Math.round(this.g) + ', '
305             + Math.round(this.b) + ', ' + this.a + ')';
306     }
307 };
308
309 /**
310 * Color.HSLA
311 */
312 Color.HSLA = function(h, s, l, a) {
313     if(isArray(h)) {
314         s = h[1];
315         l = h[2];
316         a = h[3];
317         h = h[0];
318     } else ifisObject(h)) {
319         s = h.s;
320         l = h.l;
321         a = h.a;
322         h = h.h;
323     }
324
325     this.h = h || 0;
326     this.s = s || 0;
327     this.l = l || 0;
328     this.a = !isNumber(a) ? 1 : a;
329

```

```

330     };
331
332     Color.HSLA.prototype = {
333       toRGB: function() {
334         var rgb = this.hslToRgb(this.h, this.s, this.l);
335         return new Rgba(rgb[0], rgb[1], rgb[2], this.a);
336       },
337
338       toArray: function() {
339         return [this.h, this.s, this.l, this.a];
340       },
341
342       clone: function() {
343         return new Color.HSLA(this);
344       },
345
346       toString: function() {
347         return 'hsla(' + this.h + ', ' + this.s + '%, ' + this.l + '%, ' +
348             this.a + ')';
349       };
350     }
351
352   /**
353    * Returns a color code based on the provided identifier.
354    * @param {number|string|Array<number>} id - Identifier for the color. Can
355    *   be a numeric index, a predefined color name, or an RGB array.
356    * @return {string} The hex code of the color.
357    */
358   color(id) {
359     var c = '#0072BD';
360     if(typeof id == 'number') {
361       c = colororder[id % 7];
362     } else if(Array.isArray(id)) {
363       if(id.length == 3) {
364         c = rgb2hex(id);
365       }
366     } else {
367       switch(id) {
368         case 'y':
369           case 'yellow':
370             c = '#FFFF00';
371             break;
372         case 'm':
373           case 'magenta':
374             c = '#FF00FF';
375             break;
376         case 'c':
377           case 'cyan':
378             c = '#00FFFF';
379             break;
380         case 'r':
381           case 'red':
382             c = '#FF0000';
383             break;
384       }
385     }
386   }
387
388   /**
389    * Converts an RGB array to a hex color code.
390    * @param {Array<number>} rgb - An array containing three numbers representing
391    *   the red, green, and blue components of the color.
392    * @return {string} The hex color code.
393    */
394   rgb2hex(rgb) {
395     var r = rgb[0].toString(16);
396     var g = rgb[1].toString(16);
397     var b = rgb[2].toString(16);
398
399     if(r.length == 1) r = '0' + r;
400     if(g.length == 1) g = '0' + g;
401     if(b.length == 1) b = '0' + b;
402
403     return '#' + r + g + b;
404   }
405
406   /**
407    * Converts a hex color code to an RGB array.
408    * @param {string} hex - A hex color code.
409    * @return {Array<number>} An array containing three numbers representing
410    *   the red, green, and blue components of the color.
411    */
412   hex2rgb(hex) {
413     var r = parseInt(hex.substring(1, 3), 16);
414     var g = parseInt(hex.substring(3, 5), 16);
415     var b = parseInt(hex.substring(5, 7), 16);
416
417     return [r, g, b];
418   }
419
420   /**
421    * Converts a color name to its corresponding hex code.
422    * @param {string} name - A color name.
423    * @return {string} The hex color code.
424    */
425   colorName(name) {
426     switch(name) {
427       case 'black':
428         case 'white':
429           return '#000000';
430         case 'gray':
431           return '#AAAAAA';
432         case 'lightgray':
433           return '#CCCCCC';
434         case 'darkgray':
435           return '#333333';
436         case 'brown':
437           return '#A52A2A';
438         case 'darkbrown':
439           return '#8B4513';
440         case 'blue':
441           return '#0000CD';
442         case 'darkblue':
443           return '#00008B';
444         case 'green':
445           return '#008000';
446         case 'darkgreen':
447           return '#006400';
448         case 'red':
449           return '#FF0000';
450         case 'darkred':
451           return '#8B0000';
452         case 'purple':
453           return '#800080';
454         case 'darkpurple':
455           return '#4B0082';
456         case 'pink':
457           return '#FFB6C1';
458         case 'darkpink':
459           return '#E9967A';
460         case 'yellow':
461           return '#FFFF00';
462         case 'darkyellow':
463           return '#FFDAB9';
464         case 'orange':
465           return '#FF8C00';
466         case 'darkorange':
467           return '#FF8C00';
468         case 'teal':
469           return '#008080';
470         case 'darkteal':
471           return '#006400';
472         case 'cyan':
473           return '#00FFFF';
474         case 'darkcyan':
475           return '#00FFFF';
476       default:
477         return '#000000';
478     }
479   }
480
481   /**
482    * Converts a color index to its corresponding hex code.
483    * @param {number} index - A color index.
484    * @return {string} The hex color code.
485    */
486   colorIndex(index) {
487     var c = colororder[index % 7];
488
489     return c;
490   }
491
492   /**
493    * Converts a color name to its corresponding color index.
494    * @param {string} name - A color name.
495    * @return {number} The color index.
496    */
497   colorIndexName(name) {
498     switch(name) {
499       case 'black':
500         case 'white':
501           return 0;
502         case 'gray':
503           return 1;
504         case 'lightgray':
505           return 2;
506         case 'darkgray':
507           return 3;
508         case 'brown':
509           return 4;
510         case 'darkbrown':
511           return 5;
512         case 'blue':
513           return 6;
514         case 'darkblue':
515           return 7;
516         case 'green':
517           return 8;
518         case 'darkgreen':
519           return 9;
520         case 'red':
521           return 10;
522         case 'darkred':
523           return 11;
524         case 'purple':
525           return 12;
526         case 'darkpurple':
527           return 13;
528         case 'pink':
529           return 14;
530         case 'darkpink':
531           return 15;
532         case 'yellow':
533           return 16;
534         case 'darkyellow':
535           return 17;
536         case 'orange':
537           return 18;
538         case 'darkorange':
539           return 19;
540         case 'teal':
541           return 20;
542         case 'darkteal':
543           return 21;
544         case 'cyan':
545           return 22;
546         case 'darkcyan':
547           return 23;
548       default:
549         return 0;
550     }
551   }
552
553   /**
554    * Converts a color name to its corresponding color name.
555    * @param {string} name - A color name.
556    * @return {string} The color name.
557    */
558   colorNameName(name) {
559     switch(name) {
560       case 'black':
561         case 'white':
562           return 'black';
563         case 'gray':
564           return 'gray';
565         case 'lightgray':
566           return 'lightgray';
567         case 'darkgray':
568           return 'darkgray';
569         case 'brown':
570           return 'brown';
571         case 'darkbrown':
572           return 'darkbrown';
573         case 'blue':
574           return 'blue';
575         case 'darkblue':
576           return 'darkblue';
577         case 'green':
578           return 'green';
579         case 'darkgreen':
580           return 'darkgreen';
581         case 'red':
582           return 'red';
583         case 'darkred':
584           return 'darkred';
585         case 'purple':
586           return 'purple';
587         case 'darkpurple':
588           return 'darkpurple';
589         case 'pink':
590           return 'pink';
591         case 'darkpink':
592           return 'darkpink';
593         case 'yellow':
594           return 'yellow';
595         case 'darkyellow':
596           return 'darkyellow';
597         case 'orange':
598           return 'orange';
599         case 'darkorange':
600           return 'darkorange';
601         case 'teal':
602           return 'teal';
603         case 'darkteal':
604           return 'darkteal';
605         case 'cyan':
606           return 'cyan';
607         case 'darkcyan':
608           return 'darkcyan';
609       default:
610         return 'black';
611     }
612   }
613
614   /**
615    * Converts a color index to its corresponding color name.
616    * @param {number} index - A color index.
617    * @return {string} The color name.
618    */
619   colorIndexNameName(index) {
620     switch(index) {
621       case 0:
622         case 1:
623           return 'black';
624         case 2:
625           return 'gray';
626         case 3:
627           return 'lightgray';
628         case 4:
629           return 'brown';
630         case 5:
631           return 'darkbrown';
632         case 6:
633           return 'blue';
634         case 7:
635           return 'darkblue';
636         case 8:
637           return 'green';
638         case 9:
639           return 'darkgreen';
640         case 10:
641           return 'red';
642         case 11:
643           return 'darkred';
644         case 12:
645           return 'purple';
646         case 13:
647           return 'darkpurple';
648         case 14:
649           return 'pink';
650         case 15:
651           return 'darkpink';
652         case 16:
653           return 'yellow';
654         case 17:
655           return 'darkyellow';
656         case 18:
657           return 'orange';
658         case 19:
659           return 'darkorange';
660         case 20:
661           return 'teal';
662         case 21:
663           return 'darkteal';
664         case 22:
665           return 'cyan';
666         case 23:
667           return 'darkcyan';
668       default:
669         return 'black';
670     }
671   }
672
673   /**
674    * Converts a color name to its corresponding color index.
675    * @param {string} name - A color name.
676    * @return {number} The color index.
677    */
678   colorNameIndex(name) {
679     switch(name) {
680       case 'black':
681         case 'white':
682           return 0;
683         case 'gray':
684           return 1;
685         case 'lightgray':
686           return 2;
687         case 'darkgray':
688           return 3;
689         case 'brown':
690           return 4;
691         case 'darkbrown':
692           return 5;
693         case 'blue':
694           return 6;
695         case 'darkblue':
696           return 7;
697         case 'green':
698           return 8;
699         case 'darkgreen':
700           return 9;
701         case 'red':
702           return 10;
703         case 'darkred':
704           return 11;
705         case 'purple':
706           return 12;
707         case 'darkpurple':
708           return 13;
709         case 'pink':
710           return 14;
711         case 'darkpink':
712           return 15;
713         case 'yellow':
714           return 16;
715         case 'darkyellow':
716           return 17;
717         case 'orange':
718           return 18;
719         case 'darkorange':
720           return 19;
721         case 'teal':
722           return 20;
723         case 'darkteal':
724           return 21;
725         case 'cyan':
726           return 22;
727         case 'darkcyan':
728           return 23;
729       default:
730         return 0;
731     }
732   }
733
734   /**
735    * Converts a color index to its corresponding color name.
736    * @param {number} index - A color index.
737    * @return {string} The color name.
738    */
739   colorIndexNameIndex(index) {
740     switch(index) {
741       case 0:
742         case 1:
743           return 'black';
744         case 2:
745           return 'gray';
746         case 3:
747           return 'lightgray';
748         case 4:
749           return 'brown';
750         case 5:
751           return 'darkbrown';
752         case 6:
753           return 'blue';
754         case 7:
755           return 'darkblue';
756         case 8:
757           return 'green';
758         case 9:
759           return 'darkgreen';
760         case 10:
761           return 'red';
762         case 11:
763           return 'darkred';
764         case 12:
765           return 'purple';
766         case 13:
767           return 'darkpurple';
768         case 14:
769           return 'pink';
770         case 15:
771           return 'darkpink';
772         case 16:
773           return 'yellow';
774         case 17:
775           return 'darkyellow';
776         case 18:
777           return 'orange';
778         case 19:
779           return 'darkorange';
780         case 20:
781           return 'teal';
782         case 21:
783           return 'darkteal';
784         case 22:
785           return 'cyan';
786         case 23:
787           return 'darkcyan';
788       default:
789         return 'black';
790     }
791   }
792
793   /**
794    * Converts a color name to its corresponding hex color code.
795    * @param {string} name - A color name.
796    * @return {string} The hex color code.
797    */
798   colorNameHex(name) {
799     switch(name) {
800       case 'black':
801         case 'white':
802           return '#000000';
803         case 'gray':
804           return '#AAAAAA';
805         case 'lightgray':
806           return '#CCCCCC';
807         case 'darkgray':
808           return '#333333';
809         case 'brown':
810           return '#A52A2A';
811         case 'darkbrown':
812           return '#8B4513';
813         case 'blue':
814           return '#0000CD';
815         case 'darkblue':
816           return '#00008B';
817         case 'green':
818           return '#008000';
819         case 'darkgreen':
820           return '#006400';
821         case 'red':
822           return '#FF0000';
823         case 'darkred':
824           return '#8B0000';
825         case 'purple':
826           return '#800080';
827         case 'darkpurple':
828           return '#4B0082';
829         case 'pink':
830           return '#FFB6C1';
831         case 'darkpink':
832           return '#E9967A';
833         case 'yellow':
834           return '#FFFF00';
835         case 'darkyellow':
836           return '#FFDAB9';
837         case 'orange':
838           return '#FF8C00';
839         case 'darkorange':
840           return '#FF8C00';
841         case 'teal':
842           return '#008080';
843         case 'darkteal':
844           return '#006400';
845         case 'cyan':
846           return '#00FFFF';
847         case 'darkcyan':
848           return '#00FFFF';
849       default:
850         return '#000000';
851     }
852   }
853
854   /**
855    * Converts a color index to its corresponding hex color code.
856    * @param {number} index - A color index.
857    * @return {string} The hex color code.
858    */
859   colorIndexHex(index) {
860     var c = colororder[index % 7];
861
862     return c;
863   }
864
865   /**
866    * Converts a hex color code to its corresponding color index.
867    * @param {string} hex - A hex color code.
868    * @return {number} The color index.
869    */
870   hexColorIndex(hex) {
871     var r = parseInt(hex.substring(1, 3), 16);
872     var g = parseInt(hex.substring(3, 5), 16);
873     var b = parseInt(hex.substring(5, 7), 16);
874
875     return [r, g, b];
876   }
877
878   /**
879    * Converts a hex color code to its corresponding color name.
880    * @param {string} hex - A hex color code.
881    * @return {string} The color name.
882    */
883   hexColorName(hex) {
884     switch(hex) {
885       case '#000000':
886         case '#FFFFFF':
887           return 'black';
888         case '#AAAAAA':
889           return 'white';
890         case '#CCCCCC':
891           return 'gray';
892         case '#333333':
893           return 'lightgray';
894         case '#A52A2A':
895           return 'brown';
896         case '#8B4513':
897           return 'darkbrown';
898         case '#0000CD':
899           return 'blue';
900         case '#00008B':
901           return 'darkblue';
902         case '#008000':
903           return 'green';
904         case '#006400':
905           return 'darkgreen';
906         case '#FF0000':
907           return 'red';
908         case '#8B0000':
909           return 'darkred';
910         case '#800080':
911           return 'purple';
912         case '#4B0082':
913           return 'darkpurple';
914         case '#FFB6C1':
915           return 'pink';
916         case '#E9967A':
917           return 'darkpink';
918         case '#FFFF00':
919           return 'yellow';
920         case '#FFDAB9':
921           return 'darkyellow';
922         case '#FF8C00':
923           return 'orange';
924         case '#FF8C00':
925           return 'darkorange';
926         case '#008080':
927           return 'teal';
928         case '#006400':
929           return 'darkteal';
930         case '#00FFFF':
931           return 'cyan';
932         case '#00FFFF':
933           return 'darkcyan';
934       default:
935         return 'black';
936     }
937   }
938
939   /**
940    * Converts a color name to its corresponding color index.
941    * @param {string} name - A color name.
942    * @return {number} The color index.
943    */
944   colorNameColorIndex(name) {
945     switch(name) {
946       case 'black':
947         case 'white':
948           return 0;
949         case 'gray':
950           return 1;
951         case 'lightgray':
952           return 2;
953         case 'darkgray':
954           return 3;
955         case 'brown':
956           return 4;
957         case 'darkbrown':
958           return 5;
959         case 'blue':
960           return 6;
961         case 'darkblue':
962           return 7;
963         case 'green':
964           return 8;
965         case 'darkgreen':
966           return 9;
967         case 'red':
968           return 10;
969         case 'darkred':
970           return 11;
971         case 'purple':
972           return 12;
973         case 'darkpurple':
974           return 13;
975         case 'pink':
976           return 14;
977         case 'darkpink':
978           return 15;
979         case 'yellow':
980           return 16;
981         case 'darkyellow':
982           return 17;
983         case 'orange':
984           return 18;
985         case 'darkorange':
986           return 19;
987         case 'teal':
988           return 20;
989         case 'darkteal':
990           return 21;
991         case 'cyan':
992           return 22;
993         case 'darkcyan':
994           return 23;
995       default:
996         return 0;
997     }
998   }
999
1000  /**
1001   * Converts a color index to its corresponding color name.
1002   * @param {number} index - A color index.
1003   * @return {string} The color name.
1004   */
1005   colorIndexColorName(index) {
1006     switch(index) {
1007       case 0:
1008         case 1:
1009           return 'black';
1010         case 2:
1011           return 'gray';
1012         case 3:
1013           return 'lightgray';
1014         case 4:
1015           return 'brown';
1016         case 5:
1017           return 'darkbrown';
1018         case 6:
1019           return 'blue';
1020         case 7:
1021           return 'darkblue';
1022         case 8:
1023           return 'green';
1024         case 9:
1025           return 'darkgreen';
1026         case 10:
1027           return 'red';
1028         case 11:
1029           return 'darkred';
1030         case 12:
1031           return 'purple';
1032         case 13:
1033           return 'darkpurple';
1034         case 14:
1035           return 'pink';
1036         case 15:
1037           return 'darkpink';
1038         case 16:
1039           return 'yellow';
1040         case 17:
1041           return 'darkyellow';
1042         case 18:
1043           return 'orange';
1044         case 19:
1045           return 'darkorange';
1046         case 20:
1047           return 'teal';
1048         case 21:
1049           return 'darkteal';
1050         case 22:
1051           return 'cyan';
1052         case 23:
1053           return 'darkcyan';
1054       default:
1055         return 'black';
1056     }
1057   }
1058
1059   /**
1060    * Converts a color name to its corresponding color name.
1061    * @param {string} name - A color name.
1062    * @return {string} The color name.
1063    */
1064   colorNameColorName(name) {
1065     switch(name) {
1066       case 'black':
1067         case 'white':
1068           return 'black';
1069         case 'gray':
1070           return 'gray';
1071         case 'lightgray':
1072           return 'lightgray';
1073         case 'darkgray':
1074           return 'darkgray';
1075         case 'brown':
1076           return 'brown';
1077         case 'darkbrown':
1078           return 'darkbrown';
1079         case 'blue':
1080           return 'blue';
1081         case 'darkblue':
1082           return 'darkblue';
1083         case 'green':
1084           return 'green';
1085         case 'darkgreen':
1086           return 'darkgreen';
1087         case 'red':
1088           return 'red';
1089         case 'darkred':
1090           return 'darkred';
1091         case 'purple':
1092           return 'purple';
1093         case 'darkpurple':
1094           return 'darkpurple';
1095         case 'pink':
1096           return 'pink';
1097         case 'darkpink':
1098           return 'darkpink';
1099         case 'yellow':
1100           return 'yellow';
1101         case 'darkyellow':
1102           return 'darkyellow';
1103         case 'orange':
1104           return 'orange';
1105         case 'darkorange':
1106           return 'darkorange';
1107         case 'teal':
1108           return 'teal';
1109         case 'darkteal':
1110           return 'darkteal';
1111         case 'cyan':
1112           return 'cyan';
1113         case 'darkcyan':
1114           return 'darkcyan';
1115       default:
1116         return 'black';
1117     }
1118   }
1119
1120   /**
1121    * Converts a color index to its corresponding color name.
1122    * @param {number} index - A color index.
1123    * @return {string} The color name.
1124    */
1125   colorIndexColorName(index) {
1126     switch(index) {
1127       case 0:
1128         case 1:
1129           return 'black';
1130         case 2:
1131           return 'gray';
1132         case 3:
1133           return 'lightgray';
1134         case 4:
1135           return 'brown';
1136         case 5:
1137           return 'darkbrown';
1138         case 6:
1139           return 'blue';
1140         case 7:
1141           return 'darkblue';
1142         case 8:
1143           return 'green';
1144         case 9:
1145           return 'darkgreen';
1146         case 10:
1147           return 'red';
1148         case 11:
1149           return 'darkred';
1150         case 12:
1151           return 'purple';
1152         case 13:
1153           return 'darkpurple';
1154         case 14:
1155           return 'pink';
1156         case 15:
1157           return 'darkpink';
1158         case 16:
1159           return 'yellow';
1160         case 17:
1161           return 'darkyellow';
1162         case 18:
1163           return 'orange';
1164         case 19:
1165           return 'darkorange';
1166         case 20:
1167           return 'teal';
1168         case 21:
1169           return 'darkteal';
1170         case 22:
1171           return 'cyan';
1172         case 23:
1173           return 'darkcyan';
1174       default:
1175         return 'black';
1176     }
1177   }
1178
1179   /**
1180    * Converts a color name to its corresponding color index.
1181    * @param {string} name - A color name.
1182    * @return {number} The color index.
1183    */
1184   colorNameColorIndex(name) {
1185     switch(name) {
1186       case 'black':
1187         case 'white':
1188           return 0;
1189         case 'gray':
1190           return 1;
1191         case 'lightgray':
1192           return 2;
1193         case 'darkgray':
1194           return 3;
1195         case 'brown':
1196           return 4;
1197         case 'darkbrown':
1198           return 5;
1199         case 'blue':
1200           return 6;
1201         case 'darkblue':
1202           return 7;
1203         case 'green':
1204           return 8;
1205         case 'darkgreen':
1206           return 9;
1207         case 'red':
1208           return 10;
1209         case 'darkred':
1210           return 11;
1211         case 'purple':
1212           return 12;
1213         case 'darkpurple':
1214           return 13;
1215         case 'pink':
1216           return 14;
1217         case 'darkpink':
1218           return 15;
1219         case 'yellow':
1220           return 16;
1221         case 'darkyellow':
1222           return 17;
1223         case 'orange':
1224           return 18;
1225         case 'darkorange':
1226           return 19;
1227         case 'teal':
1228           return 20;
1229         case 'darkteal':
1230           return 21;
1231         case 'cyan':
1232           return 22;
1233         case 'darkcyan':
1234           return 23;
1235       default:
1236         return 0;
1237     }
1238   }
1239
1240   /**
1241    * Converts a color index to its corresponding color name.
1242    * @param {number} index - A color index.
1243    * @return {string} The color name.
1244    */
1245   colorIndexColorName(index) {
1246     switch(index) {
1247       case 0:
1248         case 1:
1249           return 'black';
1250         case 2:
1251           return 'gray';
1252         case 3:
1253           return 'lightgray';
1254         case 4:
1255           return 'brown';
1256         case 5:
1257           return 'darkbrown';
1258         case 6:
1259           return 'blue';
1260         case 7:
1261           return 'darkblue';
1262         case 8:
1263           return 'green';
1264         case 9:
1265           return 'darkgreen';
1266         case 10:
1267           return 'red';
1268         case 11:
1269           return 'darkred';
1270         case 12:
1271           return 'purple';
1272         case 13:
1273           return 'darkpurple';
1274         case 14:
1275           return 'pink';
1276         case 15:
1277           return 'darkpink';
1278         case 16:
1279           return 'yellow';
1280         case 17:
1281           return 'darkyellow';
1282         case 18:
1283           return 'orange';
1284         case 19:
1285           return 'darkorange';
1286         case 20:
1287           return 'teal';
1288         case 21:
1289           return 'darkteal';
1290         case 22:
1291           return 'cyan';
1292         case 23:
1293           return 'darkcyan';
1294       default:
1295         return 'black';
1296     }
1297   }
1298
1299   /**
1300    * Converts a color name to its corresponding hex color code.
1301    * @param {string} name - A color name.
1302    * @return {string} The hex color code.
1303    */
1304   colorNameHexColorName(name) {
1305     switch(name) {
1306       case 'black':
1307         case 'white':
1308           return '#000000';
1309         case 'gray':
1310           return '#AAAAAA';
1311         case 'lightgray':
1312           return '#CCCCCC';
1313         case 'darkgray':
1314           return '#333333';
1315         case 'brown':
1316           return '#A52A2A';
1317         case 'darkbrown':
1318           return '#8B4513';
1319         case 'blue':
1320           return '#0000CD';
1321         case 'darkblue':
1322           return '#00008B';
1323         case 'green':
1324           return '#008000';
1325         case 'darkgreen':
1326           return '#006400';
1327         case 'red':
1328           return '#FF0000';
1329         case 'darkred':
1330           return '#8B0000';
1331         case 'purple':
1332           return '#800080';
1333         case 'darkpurple':
1334           return '#4B0082';
1335         case 'pink':
1336           return '#FFB6C1';
1337         case 'darkpink':
1338           return '#E9967A';
1339         case 'yellow':
1340           return '#FFFF00';
1341         case 'darkyellow':
1342           return '#FFDAB9';
1343         case 'orange':
1344           return '#FF8C00';
1345         case 'darkorange':
1346           return '#FF8C00';
1347         case 'teal':
1348           return '#008080';
1349         case 'darkteal':
1350           return '#006400';
1351         case 'cyan':
1352           return '#00FFFF';
1353         case 'darkcyan':
1354           return '#00FFFF';
1355       default:
1356         return '#000000';
1357     }
1358   }
1359
1360   /**
1361    * Converts a color index to its corresponding hex color code.
1362    * @param {number} index - A color index.
1363    * @return {string} The hex color code.
1364    */
1365   colorIndexHexColorName(index) {
1366     var c = colororder[index % 7];
1367
1368     return c;
1369   }
1370
1371   /**
1372    * Converts a hex color code to its corresponding color index.
1373    * @param {string} hex - A hex color code.
1374    * @return {number} The color index.
1375    */
1376   hexColorIndexHexColorName(hex) {
1377     var r = parseInt(hex.substring(1, 3), 16);
1378     var g = parseInt(hex.substring(3, 5), 16);
1379     var b = parseInt(hex.substring(5, 7), 16);
1380
1381     return [r, g, b];
1382   }
1383
1384   /**
1385    * Converts a hex color code to its corresponding color name.
1386    * @param {string} hex - A hex color code.
1387    * @return {string} The color name.
1388    */
1389   hexColorNameHexColorName(hex) {
1390     switch(hex) {
1391       case '#000000':
1392         case '#FFFFFF':
1393           return 'black';
1394         case '#AAAAAA':
1395           return 'white';
1396         case '#CCCCCC':
1397           return 'gray';
1398         case '#333333':
1399           return 'lightgray';
1400         case '#A52A2A':
1401           return 'brown';
1402         case '#8B4513':
1403           return 'darkbrown';
1404         case '#0000CD':
1405           return 'blue';
1406         case '#00008B':
1407           return 'darkblue';
1408         case '#008000':
1409           return 'green';
1410         case '#006400':
1411           return 'darkgreen';
1412         case '#FF0000':
1413           return 'red';
1414         case '#8B0000':
1415           return 'darkred';
1416         case '#800080':
1417           return 'purple';
1418         case '#4B0082':
1419           return 'darkpurple';
1420         case '#FFB6C1':
1421           return 'pink';
1422         case '#E9967A':
1423           return 'darkpink';
1424         case '#FFFF00':
1425           return 'yellow';
1426         case '#FFDAB9':
1427           return 'darkyellow';
1428         case '#FF8C00':
1429           return 'orange';
1430         case '#FF8C00':
1431           return 'darkorange';
1432         case '#008080':
1433           return 'teal';
1434         case '#006400':
1435           return 'darkteal';
1436         case '#00FFFF':
1437           return 'cyan';
1438         case '#00FFFF':
1439           return 'darkcyan';
1440       default:
1441         return 'black';
1442     }
1443   }
1444
1445   /**
1446    * Converts a color name to its corresponding hex color code.
1447    * @param {string} name - A color name.
1448    * @return {string} The hex color code.
1449    */
1450   colorNameColorIndexHexColorName(name) {
1451     switch(name) {
1452       case 'black':
1453         case 'white':
1454           return '#000000';
1455         case 'gray':
1456           return '#AAAAAA';
1457         case 'lightgray':
1458           return '#CCCCCC';
1459         case 'darkgray':
1460           return '#333333';
1461         case 'brown':
1462           return '#A52A2A';
1463         case 'darkbrown':
1464           return '#8B4513';
1465         case 'blue':
1466           return '#0000CD';
1467         case 'darkblue
```

```

383         case 'g':
384         case 'green':
385             c = '#00FF00';
386             break;
387         case 'b':
388         case 'blue':
389             c = '#0000FF';
390             break;
391         case 'w':
392         case 'white':
393             c = '#FFFFFF';
394             break;
395         case 'k':
396         case 'black':
397             c = '#000000';
398             break;
399     }
400 }
401 return c;
402 }

404 /**
405 * Calculates a color on a gradient from green to red based on a value.
406 * @param {number} value - A number between 0 and 1 indicating position on
407 * the gradient.
408 * @param {number} [k=0.3] - A scaling factor to adjust the gradient effect.
409 * @return {string} The hsl color string.
410 */
411 getColorG2R(value, k = 0.3) {
412     if(value < 0) {
413         value = 0;
414     } else if(value < k) {
415         value = value/k;
416     } else {
417         value = 1;
418     }
419     var hue=(120*value).toString(10);
420     return ["hsl(",hue,",100%,50%)"].join("");
421 }

422 /**
423 * Calculates the gradient color between two colors based on a percentage.
424 * @param {number} p - The percentage (0 to 1) between the two colors.
425 * @param {Array<number>} rgb_beginning - The RGB values of the start color.
426 * @param {Array<number>} rgb_end - The RGB values of the end color.
427 * @return {Array<number>} The RGB values of the calculated gradient color.
428 */
429 colourGradient(p, rgb_beginning, rgb_end){
430     var w = p * 2 - 1;
431     var w1 = (w + 1) / 2.0;
432     var w2 = 1 - w1;

433     var rgb = [parseInt(rgb_beginning[0] * w1 + rgb_end[0] * w2),
434                 parseInt(rgb_beginning[1] * w1 + rgb_end[1] * w2),
435                 parseInt(rgb_beginning[2] * w1 + rgb_end[2] * w2)];

```

```

437     return rgb;
438 }
439
440 /**
441 * Converts RGB color values to HEX.
442 * @param {number} r - The red color value (0-255).
443 * @param {number} g - The green color value (0-255).
444 * @param {number} b - The blue color value (0-255).
445 * @return {Array<number>} The HEX representation of the color.
446 */
447 rgb2hex(r, g, b) {
448     // If the first argument is an array, unpack its values
449     if(Array.isArray(r)) {
450         [r, g, b] = r;
451     }
452
453     function toHex(c) {
454         // Scale to 0-255 if the value is between 0 and 1
455         c = (c <= 1) ? Math.round(c * 255) : Math.round(c);
456         // Clamp the value between 0 and 255
457         c = Math.min(255, Math.max(0, c));
458         // Convert to hexadecimal and pad with zeros if necessary
459         return c.toString(16).padStart(2, '0').toUpperCase();
460     }
461     return '#' + toHex(r) + toHex(g) + toHex(b);
462 }
463
464 /**
465 * Converts RGB color values to HSL (Hue, Saturation, Lightness).
466 * @param {number} r - The red color value (0-255).
467 * @param {number} g - The green color value (0-255).
468 * @param {number} b - The blue color value (0-255).
469 * @return {Array<number>} The HSL representation of the color.
470 */
471 rgbToHsl(r, g, b) {
472     r /= 255;
473     g /= 255;
474     b /= 255;
475
476     var max = Math.max(r, g, b);
477     var min = Math.min(r, g, b);
478     var h, s, l;
479
480     l = (max + min) / 2;
481
482     if(max === min) {
483         h = s = 0;
484     } else {
485         var d = max - min;
486         switch (max) {
487             case r: h = ((g - b) / d * 60 + 360) % 360; break;
488             case g: h = (b - r) / d * 60 + 120; break;
489             case b: h = (r - g) / d * 60 + 240; break;
490         }
491         s = l <= 0.5 ? d / (max + min) : d / (2 - max - min);

```

```

492     }
493
494     return [h, s * 100, l * 100];
495 }
496
497 /**
498 * Converts HSL color values to RGB.
499 * @param {number} h - The hue value (0-360).
500 * @param {number} s - The saturation value (0-100).
501 * @param {number} l - The lightness value (0-100).
502 * @return {Array<number>} The RGB representation of the color.
503 */
504 hslToRgb(h, s, l) {
505     s /= 100;
506     l /= 100;
507
508     var r, g, b;
509
510     if(s === 0){
511         r = g = b = l * 255;
512     } else {
513         var v2 = l < 0.5 ? l * (1 + s) : l + s - l * s;
514         var v1 = 2 * l - v2;
515         r = Math.round(this.hueToRgb(v1, v2, h + 120) * 255);
516         g = Math.round(this.hueToRgb(v1, v2, h) * 255);
517         b = Math.round(this.hueToRgb(v1, v2, h - 120) * 255);
518     }
519
520     return [r, g, b];
521 }
522
523 /**
524 * Helper function for converting a hue to RGB.
525 * @param {number} v1 - Helper value 1.
526 * @param {number} v2 - Helper value 2.
527 * @param {number} vh - The hue value to convert.
528 * @return {number} The RGB value for the hue.
529 */
530 hueToRgb(v1, v2, vh) {
531     vh /= 360;
532     if(vh < 0) vh++;
533     if(vh > 1) vh--;
534     if(vh < 1 / 6) return v1 + (v2 - v1) * 6 * vh;
535     if(vh < 1 / 2) return v2;
536     if(vh < 2 / 3) return v1 + (v2 - v1) * (2 / 3 - vh) * 6;
537     return v1;
538 }
539 }
540
541 exports.PRDC_JSLAB_LIB_COLOR = PRDC_JSLAB_LIB_COLOR;

```

Listing 86 - color.js

```

1 /**
2 * @file JSLAB library control submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>

```

```

4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9  * Class for JSLAB control submodule.
10 */
11 class PRDC_JSLAB_LIB_CONTROL {
12
13 /**
14  * Initializes the control submodule.
15  * @param {Object} js1 Reference to the main JSLAB object.
16  */
17 constructor(js1) {
18   var obj = this;
19   this.js1 = js1;
20 }
21
22 /**
23  * Create a transfer function representation.
24  * @param {number[]} num - Numerator coefficients of the transfer function.
25  * @param {number[]} den - Denominator coefficients of the transfer function
26  *
27  * @param {number} [Ts=0] - Sampling time, defaults to 0 for continuous-time
28  * systems.
29  * @returns {object} An object representing the transfer function { num, den
30  * , Ts }.
31  */
32 tf(num, den, Ts = 0) {
33   return { num, den, Ts };
34 }
35
36 /**
37  * Create a state-space representation.
38  * @param {number[][]} A - System matrix.
39  * @param {number[][]} B - Input matrix.
40  * @param {number[][]} C - Output matrix.
41  * @param {number[][]} D - Feedthrough matrix.
42  * @param {number} [Ts=0] - Sampling time, defaults to 0 for continuous-time
43  * systems.
44  * @returns {object} An object representing the state-space system { A, B, C
45  * , D, Ts }.
46  */
47 ss(A, B, C, D, Ts = 0) {
48   return { A, B, C, D, Ts };
49 }
50
51 /**
52  * Convert a transfer function to a state-space representation.
53  * @param {number[]} num - Numerator coefficients of the transfer function.
54  * @param {number[]} den - Denominator coefficients of the transfer function
55  *
56  * @returns {object} State-space representation of the system.
57  */
58 tf2ss(num, den) {

```

```

53   const n = den.length;
54   num = [...zeros(n - num.length), ...num];
55
56   let A = zeros(n-1).map(() => zeros(n-1));
57   for(let i = 1; i < n - 1; i++) A[i-1][i] = 1;
58   A[n - 2] = fliplr(den.slice(1).map(d => -d));
59
60   const C = fliplr(num.map((c, i) => c - den[i] * num[0]).slice(1));
61   const B = zeros(n - 2).concat([1]);
62   const D = num[0];
63
64   return this.ss(A, B, C, D);
65 }
66
67 /**
68 * Convert a state-space representation to a transfer function.
69 * @param {object} sys - State-space system { A, B, C, D }.
70 * @returns {object} Transfer function representation { num, den }.
71 */
72 ss2tf(sys) {
73   const { A, B, C, D } = sys;
74
75   const den = charpoly(A);
76   const num = plus(charpoly(minus(A, this.jsl.env.math.multiply(B, [C]))),
77     scale(den, D-1));
78
79   var p = den.length - num.length;
80   if(p > 0) {
81     num = [...zeros(p), ...num];
82   }
83   return { num, den };
84 }
85
86 /**
87 * Convert a continuous-time transfer function to discrete-time.
88 * @param {number[]} numc - Continuous-time numerator coefficients.
89 * @param {number[]} denc - Continuous-time denominator coefficients.
90 * @param {number} Ts - Sampling time.
91 * @returns {object} Discrete-time transfer function representation { num,
92   den }.
93 */
94 c2d(numc, denc, Ts) {
95   const sysc = this.tf2ss(numc, denc);
96   const sysd = this._c2dZOH(sysc, Ts);
97   return this.ss2tf(sysd);
98 }
99
100 /**
101 * Convert a continuous-time state-space system to discrete-time using Zero-
102 * Order Hold.
103 * @param {object} sysc - Continuous-time state-space system { A, B, C, D }.
104 * @param {number} Ts - Sampling time.
105 * @returns {object} Discrete-time state-space system { A, B, C, D, Ts }.
106 */
107 _c2dZOH(sysc, Ts) {

```

```

106  const { A, B, C, D } = sys;
107  const n = A.length;
108
109  const M = A.map((row, i) => [ ...row, B[i]] );
110  M.push(zeros(n + 1));
111  const M_exp = js1.env.math.expm(js1.env.math.multiply(M, Ts))._data.slice
112    (0, n);
113
114  const Ad = M_exp.map(row => row.slice(0, n));
115  const Bd = M_exp.map(row => row.slice(n));
116
117  return this.ss(Ad, Bd, C, D, Ts);
118 }
119 /**
120 * Simulate the time response of a discrete-time linear system.
121 * @param {number[]} sys - transfer function of system.
122 * @param {number[]} u - Input signal array.
123 * @param {number[]} t - Time vector array.
124 * @returns {object} An object containing the response:
125 *             - y: Output signal array.
126 *             - t: Time vector array (same as input).
127 */
128 lsim(sys, u, t, Ts) {
129   var sysd = sys;
130   if(!sys.Ts && Ts) {
131     sysd = this.c2d(sys.num, sys.den, Ts);
132   }
133   var num = sysd.num;
134   var den = sysd.den;
135
136 // Normalize the denominator coefficients so that den[0] = 1
137 if(den[0] !== 1) {
138   const den0 = den[0];
139   for(let i = 0; i < den.length; i++) {
140     den[i] /= den0;
141   }
142   for(let i = 0; i < num.length; i++) {
143     num[i] /= den0;
144   }
145 }
146
147 const N = u.length;
148 var y = zeros(N); // Initialize output array with zeros
149
150 // Iterate over each time step starting from n=1
151 for(let n = 0; n < N; n++) {
152   // Compute feedforward terms: sum(num[k] * u[n - k]) for k =0 to M
153   for(let k = 0; k < num.length; k++) {
154     const idx = n - k;
155     if(idx >= 0) {
156       y[n] += num[k] * u[idx];
157     }
158   }
159 }
```

```

160   // Compute feedback terms: sum(den[k] * y[n - k]) for k =1 to N
161   for(let k = 1; k < den.length; k++) {
162     const idx = n - k;
163     if(idx >= 0) {
164       y[n] -= den[k] * y[idx];
165     }
166   }
167 }
168
169 if(num.length < den.length) {
170   var n = den.length - num.length;
171   y = [...zeros(n), ...y.slice(0, y.length - n)];
172 }
173
174 return { y, t };
175 }
176
177 /**
178 * Simulates the system response over a specified time period.
179 * @param {Object} sys - The system to simulate.
180 * @param {number} Tfinal - The final time for the simulation.
181 * @returns {Object} The simulation result.
182 */
183 step(sys, Tfinal) {
184   var u = ones(101);
185   var t = linspace(0, Tfinal, 101);
186   var Ts = Tfinal/100;
187   return this.lsim(sys, u, t, Ts);
188 }
189
190 /**
191 * Estimates a transfer function model using numerical optimization.
192 * @param {Array<number>} t - Time vector.
193 * @param {Array<number>} u - Input signal vector.
194 * @param {Array<number>} y - Output signal vector.
195 * @param {number} np - Number of poles.
196 * @param {number} nz - Number of zeros.
197 * @param {string} [method='NelderMead'] - Optimization method ('NelderMead'
198 * or 'Powell').
199 * @returns {{sys: object, error: number}} Estimated transfer function and
200 * mean squared error.
201 */
202 tfest(t, u, y, np, nz, method = 'NelderMead') {
203   var obj = this;
204   var Ts = mean(diff(t));
205   var N = np+nz+1;
206   var f = (x) => { // funkcija prilagodjenosti
207     try {
208       var den = [1, ...x.slice(0, np)];
209       var num = x.slice(np);
210       var sys = tf(num, den);
211       var r = obj.lsim(sys, u, t, Ts);
212       var mse = obj.jsl.math.mse(y, r.y);
213       return mse;
214     } catch(err) {

```

```

213         return Infinity ;
214     }
215 };
216 var r ;
217 if(method == 'NelderMead') {
218     r = this . jsl . optim . optimNelderMead(f , zeros(N)) ;
219 } else if(method == 'Powell') {
220     r = this . jsl . optim . optimPowell(f , zeros(N)) ;
221 }
222 var den = [1, ...r.x.slice(0, np)] ;
223 var num = r.x.slice(np) ;
224 return {sys: tf(num, den) , error: r.fx} ;
225 }
226 }
227
228 exports.PRDC_JSLAB_LIB_CONTROL = PRDC_JSLAB_LIB_CONTROL;

```

Listing 87 - control.js

```

1 /**
2  * @file JSLAB library conversion submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB conversion submodule.
10 */
11 class PRDC_JSLAB_LIB_CONVERSION {
12
13 /**
14  * Constructs a conversion submodule object with access to JSLAB's
15  * conversion functions.
16  * @constructor
17  * @param {Object} jsl - Reference to the main JSLAB object .
18 */
19 constructor(jsl) {
20     var obj = this ;
21     this . jsl = jsl ;
22
23     this . speech = this . jsl . env . speech ;
24
25 /**
26  * Serializes BigInt values to JSON by converting them to strings .
27  */
28 BigInt.prototype.toJSON = function() { return this . toString() ; };
29
30 /**
31  * Converts text to speech using the Web Speech API .
32  * @param {string} msg - The text message to be spoken .
33  */
34 speak(msg) {
35     if(!global.is_worker) {
36         this . speech . text = msg ;

```

```

37         speechSynthesis.speak(this.speech);
38     }
39   }
40
41 /**
42 * Converts a number to a string with specified precision.
43 * @param {number} num - The number to convert.
44 * @param {number} precision - The number of digits after the decimal point.
45 * @returns {string} The formatted string.
46 */
47 num2str(num, precision = 2) {
48   if(isNumber(precision)) {
49     return num.toFixed(precision);
50   } else {
51     return sprintf(precision, num);
52   }
53 }
54
55 /**
56 * Changes the extension of a file path.
57 * @param {string} path - The original file path.
58 * @param {string} ext_new - The new extension without the dot.
59 * @returns {string} The file path with the new extension.
60 */
61 changeExtension(path, ext_new) {
62   var pos = path.lastIndexOf('.');
63   return path.substr(0, pos < 0 ? path.length : pos) + '.' + ext_new;
64 }
65
66 /**
67 * Remove the extension of a file path.
68 * @param {string} path - The original file path.
69 * @returns {string} The file path without extension.
70 */
71 removeExtension(path) {
72   var pos = path.lastIndexOf('.');
73   return path.substr(0, pos < 0 ? path.length : pos);
74 }
75
76 /**
77 * Transforms coordinates from NED (North, East, Down) frame to RPY (Roll,
78 * Pitch, Yaw) frame.
79 * @param {number} roll - The roll angle in radians.
80 * @param {number} pitch - The pitch angle in radians.
81 * @param {number} yaw - The yaw angle in radians.
82 * @param {Array<number>} [A] - Optional matrix to apply transformation to.
83 * @returns {Array<Array<number>>} The transformation matrix.
84 */
85 ned2rpy(roll, pitch, yaw, A) {
86   T = [
87     [Math.cos(yaw)*Math.cos(pitch),
88      Math.sin(yaw)*Math.cos(pitch),
89      -Math.sin(pitch)],
90     [Math.cos(yaw)*Math.sin(pitch)*Math.sin(roll)-Math.sin(yaw)*Math.cos(
91       roll),
92

```

```

90     Math.sin(yaw)*Math.sin(pitch)*Math.sin(roll)+Math.cos(yaw)*Math.cos(
91         roll),
92     Math.cos(pitch)*Math.sin(roll)], [
93     [Math.cos(yaw)*Math.sin(pitch)*Math.cos(roll)+Math.sin(yaw)*Math.sin(
94         roll),
95     Math.sin(yaw)*Math.sin(pitch)*Math.cos(roll)-Math.cos(yaw)*Math.sin(
96         roll),
97     Math.cos(pitch)*Math.cos(roll)]]
98 ];
99 }
100 }
101 }
102 }
103 /**
104 * Converts an array of uint8 numbers to a string.
105 * @param {Uint8Array} data - The array of uint8 numbers.
106 * @returns {string} The converted string.
107 */
108 uint8ToString(data) {
109   return String.fromCharCode(...data);
110 }
111 /**
112 * Converts an array of hexadecimal strings to an array of decimal numbers.
113 * @param {Array<string>} hex - The array of hexadecimal strings.
114 * @returns {Array<number>} The array of decimal numbers.
115 */
116 hex2dec(hex) {
117   return hex.map(function(e) {
118     return parseInt(e, 16);
119   });
120 }
121 /**
122 * Converts a number to its ASCII character equivalent.
123 * @param {number} num - The number to convert.
124 * @returns {string} The ASCII character.
125 */
126 numToASCII(num) {
127   return (''+num).charCodeAt(0);
128 }
129 /**
130 * Converts a number to a hexadecimal string with a fixed number of digits.
131 * @param {number} num - The number to convert.
132 * @param {number} dig - The number of digits in the resulting string.
133 * @param {boolean} prefix - Whether to add a '0x' prefix.
134 * @returns {string} The hexadecimal string.
135 */
136 numToHexStr(num, dig = 4, prefix = false) {
137   var str = ('0000'.repeat(dig) + num.toString(16).toUpperCase()).slice(-dig)
138 }
139 
```

```

142     );
143     if(prefix) {
144       return '0x'+str;
145     }
146     return str;
147   }
148
149   /**
150    * Converts an int8 number to two ASCII characters.
151    * @param {number} num - The int8 number.
152    * @returns {string} The two ASCII characters.
153   */
154   int8To2ASCII(num) {
155     var data = ("0" + (Uint8Array.from([num])[0]).toString(16)).substr(-2);
156     return [data[0].charCodeAt(0), data[1].charCodeAt(0)];
157   }
158
159   /**
160    * Converts an int16 number to four ASCII characters representing its
161    * hexadecimal value.
162    * @param {number} num - The int16 number to convert.
163    * @returns {string} A string of four ASCII characters.
164   */
165   int16To4ASCII(num) {
166     var data = ("000" + (Uint16Array.from([num])[0]).toString(16).toUpperCase()
167     ()).substr(-4);
168     return [data[0].charCodeAt(0), data[1].charCodeAt(0), data[2].charCodeAt(
169     0), data[3].charCodeAt(0)];
170   }
171
172   /**
173    * Combines two uint8 values into an int16 value.
174    * @param {number} part1 - The first uint8 value.
175    * @param {number} part2 - The second uint8 value.
176    * @returns {number} The combined int16 value.
177   */
178   uint8sToInt16(part1, part2) {
179     part1 &= 0xFF;
180     part2 &= 0xFF;
181
182     let result = (part1 << 8) | part2;
183     return (result & 0x8000) ? -((result ^ 0xFFFF) + 1) : result;
184   }
185
186   /**
187    * Combines four uint8 values into an int32 value.
188    * @param {number} part1 - The first uint8 value.
189    * @param {number} part2 - The second uint8 value.
190    * @param {number} part3 - The third uint8 value.
191    * @param {number} part4 - The fourth uint8 value.
192    * @returns {number} The combined int32 value.
193   */
194   uint8sToInt32(part1, part2, part3, part4) {
195     part1 &= 0xFF;
196     part2 &= 0xFF;

```

```

193     part3 &= 0xFF;
194     part4 &= 0xFF;
195
196     let result = (part1 << 24) | (part2 << 16) | (part3 << 8) | part4;
197     return (result & 0x80000000) ? -((result ^ 0xFFFFFFFF) + 1) : result;
198 }
199
200 /**
201 * Converts four uint8 values into a floating-point number.
202 * @param {number} part1 - The first uint8 value.
203 * @param {number} part2 - The second uint8 value.
204 * @param {number} part3 - The third uint8 value.
205 * @param {number} part4 - The fourth uint8 value.
206 * @returns {number} The floating-point number.
207 */
208 uint8sToFloat(part1, part2, part3, part4) {
209     // Combine the Uint8 values into a 32-bit integer
210     let combinedValue = (part4 << 24) | (part3 << 16) | (part2 << 8) | part1;
211
212     // Interpret the integer as a float without using Math.pow
213     let sign = (combinedValue & 0x80000000) ? -1 : 1;
214     let exponent = ((combinedValue >> 23) & 0xFF) - 127;
215     let mantissa = (combinedValue & 0x7FFFFF | 0x800000) / (1 << 23);
216
217     return sign * mantissa * (2 ** exponent);
218 }
219
220 /**
221 * Converts a uint16 value to an int16 value.
222 * @param {number} num - The uint16 value to convert.
223 * @returns {number} The converted int16 value.
224 */
225 uint16ToInt16(num) {
226     return (num & 0x8000) ? -((num ^ 0xFFFF) + 1) : num;
227 }
228
229 /**
230 * Converts a uint8 number to two ASCII characters.
231 * @param {number} num - The uint8 number to convert.
232 * @returns {string} A string containing two ASCII characters representing
233 *                 the hexadecimal value.
234 */
235 uint8To2ASCII(num) {
236     var data = ("0" + (num % 256).toString(16).toUpperCase()).substr(-2);
237     return [data[0].charCodeAt(0), data[1].charCodeAt(0)];
238 }
239
240 /**
241 * Converts milliseconds to a time string in mm:ss format.
242 * @param {number} ms - The time in milliseconds.
243 * @returns {string} The time string.
244 */
245 ms2time(ms) {
246     min = Math.floor((ms/1000/60) << 0),
247     sec = Math.floor((ms/1000) % 60);

```

```
247     return ('0' + min).slice(-2) + ':' + ('0' + sec).slice(-2);
248 }
249
250 /**
251 * Converts a decimal number to a binary string.
252 * @param {number} x - The decimal number.
253 * @returns {string} The binary string.
254 */
255 dec2bin(x) {
256     return this.jsl.env.math.bin(x);
257 }
258
259 /**
260 * Converts a decimal number to a hexadecimal string.
261 * @param {number} x - The decimal number.
262 * @returns {string} The hexadecimal string.
263 */
264 dec2hex(x) {
265     return this.jsl.env.math.hex(x);
266 }
267
268 /**
269 * Converts a decimal number to an octal string.
270 * @param {number} x - The decimal number.
271 * @returns {string} The octal string.
272 */
273 dec2oct(x) {
274     return this.jsl.env.math.oct(x);
275 }
276
277 /**
278 * Rounds a number to a specified number of decimal places.
279 * @param {number} number - The number to round.
280 * @param {number} [decimals=2] - The number of decimal places.
281 * @param {boolean} [string=false] - Whether to return the result as a
282 *     string.
283 * @returns {number|string} The rounded number.
284 */
285 round(number, decimals = 2, string = false) {
286     var result;
287     if(typeof value === 'number') {
288         result = number.toFixed(decimals);
289     } else {
290         result = Number(number).toFixed(decimals);
291     }
292     if(result === 0) {
293         result = '0'; // removes -0
294     }
295     if(string) {
296         return result;
297     } else {
298         return Number(result);
299     }
300 }
```

```

301  /**
302   * Rounds a number to a fixed number of decimal places if it is a number.
303   * @param {number} value - The value to round.
304   * @param {number} p - The number of digits after the decimal point.
305   * @returns {number} The rounded number with a fixed number of decimal
306   *     places, or the original value if it is not a number.
307   */
308  roundIf(value, p) {
309    if(typeof value === 'number') {
310      return Number(value.toFixed(p));
311    } else {
312      return value;
313    }
314  }
315  /**
316   * Rounds a number to a fixed number of decimal places if it is a number.
317   * @param {number} value - The value to round.
318   * @param {number} p - The number of digits after the decimal point.
319   * @returns {number} The rounded number with a fixed number of decimal
320   *     places, or the original value if it is not a number.
321   */
322  roundIfPrec(value, p) {
323    if(typeof p === 'number') {
324      return round(value, p);
325    } else {
326      return value;
327    }
328  }
329  /**
330   * Converts a uint8_t number to a bit string.
331   * @param {number} n - The number to convert.
332   * @returns {string} A bit string representing the number.
333   */
334  bitString(n) {
335    return ("00000000" + n.toString(2)).substr(-8);
336  }
337
338  /**
339   * Generates a set of flags from a bitfield based on a mapping.
340   * @param {Object} map - The mapping of bit positions to flag names.
341   * @param {string} name_column - The column name in the mapping that
342   *     contains the flag names.
343   * @param {number} val - The bitfield value.
344   * @returns {Object} An object with keys as flag names and values indicating
345   *     the presence (1) or absence (0) of each flag.
346   */
347  getBitFlags(map, name_column, val) {
348    var flags = {};
349    var keys = Object.keys(map);
350    var bits = [...Array(keys.length)].map(function(x, i) {
351      return val >> i & 1;
352    });
353    keys.forEach(function(key) {
354      flags[map[key]] = bits[i];
355    });
356  }

```

```

352         flags[map[key][name_column]] = bits[key];
353     });
354     return flags;
355 }
356
357 /**
358 * Retrieves the enumeration value based on a property match.
359 * @param {Object} enum_object - The enumeration object to search.
360 * @param {string} prop - The property name to match.
361 * @param {*} val - The property value to match.
362 * @returns {number} The enumeration key as a number, or the index if not
363 * found.
364 */
365 getEnumVal(enum_object, prop, val) {
366     var entries = Object.entries(enum_object);
367     var idx = entries.findIndex(function(f) {
368         return f[1][prop] === val;
369     });
370     if(idx > 0) {
371         return Number(entries[idx][0]);
372     }
373     return idx;
374 }
375 /**
376 * Inverts an enumeration, swapping keys and values, optionally based on a
377 * specific property of the enumeration values.
378 * @param {Object} enum_object - The enumeration object to invert.
379 * @param {string} [prop] - An optional property name to use from the
380 * enumeration values.
381 * @returns {Object} The inverted enumeration object.
382 */
383 invertEnum(enum_object, prop) {
384     var enum_object_inv = {};
385     Object.entries(enum_object).forEach(function([key, value]) {
386         var ind = value;
387         if(prop) {
388             ind = value[prop];
389         }
390         var num_key = Number(key);
391         if(num_key === key) {
392             enum_object_inv[ind] = num_key;
393         } else {
394             enum_object_inv[ind] = key;
395         }
396     });
397     return enum_object_inv;
398 }
399 /**
400 * Converts an array of numbers to a string of hexadecimal values,
401 * optionally prefixed with "0x".
402 * @param {Array<number>} A - The array to convert.
403 * @param {boolean} [prefix=true] - Whether to add a "0x" prefix to each hex
404 * value.

```

```

402   * @returns {string} A string of hexadecimal values.
403   */
404 arrayToHexStr(A, prefix = true) {
405   var A_uint8 = new Uint8Array(A);
406   var len = A.length;
407   var A_hex = Array(len);
408   for(let i = 0; i < len; i++) {
409     A_hex[i] = A_uint8[i].toString(16).toUpperCase().padStart(2, '0');
410   }
411   if(prefix) {
412     return '0x' + A_hex.join(' 0x');
413   } else {
414     return A_hex.join(' ');
415   }
416 }
417 /**
418 * Converts an array of numbers to an ASCII string.
419 * @param {Array<number>} array - The array of numbers to convert.
420 * @returns {string} The ASCII string representation of the array.
421 */
422 arrayToASCII(A) {
423   return String.fromCharCode(...A);
424 }
425 /**
426 * Extends an object with properties from additional objects.
427 * @param {...Object} objects - The objects to merge into the target object.
428 * @returns {Object} The extended object.
429 */
430 extend() {
431   var target = arguments[0] || {}, o, p;
432
433   for(var i = 1, len = arguments.length; i < len; i++) {
434     o = arguments[i];
435
436     if(!thisisObject(o)) continue;
437
438     for(p in o) {
439       target[p] = o[p];
440     }
441   }
442
443   return target;
444 }
445 /**
446 * Normalizes the value of a radio control (RC) input.
447 * @param {number} rc - The RC input value.
448 * @param {number} [deadzone=0] - The deadzone value below which the output
449   is set to zero.
450 * @returns {number} The normalized value.
451 */
452 normalizeRC(rc, deadzone = 0) {
453   var val = rc - 1500;

```

```

456     if (Math.abs(val) < deadzone) {
457         return 0;
458     }
459     return (val - Math.sign(val)*deadzone)/(500-deadzone);
460 }
461
462 /**
463 * Checks if a value has been updated and updates the last value if it has.
464 * @param {Object} data - An object containing the current value and the
465 * last value.
466 * @returns {boolean} True if the value has been updated; false otherwise.
467 */
468 checkValueUpdate(data) {
469     if (data.value != data.last_value) {
470         data.last_value = data.value;
471         return true;
472     } else {
473         return false;
474     }
475 }
476 /**
477 * Resets the value and last value properties of an object.
478 * @param {Object} data - The object whose value and last value properties
479 * will be reset.
480 */
481 resetValue(data) {
482     data.last_value = undefined;
483     data.value = undefined;
484 }
485 /**
486 * Converts an ADC count to force in Newtons.
487 * @param {number} adc_count - The raw ADC count value.
488 * @param {number} load_cell_capacity - The capacity of the load cell in
489 * Newtons.
490 * @param {number} [adc_resolution=24] - The ADC resolution in bits.
491 * @param {number} [adc_gain=128] - The gain applied to the ADC.
492 * @param {number} [adc_sensitivity=2] - The ADC sensitivity in mV/V.
493 * @param {boolean} [adc_bipolar=true] - Indicates if the ADC is bipolar.
494 * @returns {number} The calculated force in Newtons.
495 */
496 adcToNewtons(adc_count, load_cell_capacity,
497               adc_resolution = 24, adc_gain = 128,
498               adc_sensitivity = 2, adc_bipolar = true) {
499     if (adc_bipolar) {
500         adc_resolution = adc_resolution - 1;
501     }
502     adc_sensitivity = adc_sensitivity / 1000;
503     const max_adc_count = Math.pow(2, adc_resolution) - 1;
504     const measured_adc = (adc_count / max_adc_count) / adc_gain;
505     const force = (measured_adc / adc_sensitivity) * load_cell_capacity;
506     return force;
507 }

```

```

508  /**
509   * Converts file data from a given path to a blob URL.
510   * @param {string} path - Relative path from the application's base path to
511   * the file.
512   * @returns {string} A blob URL representing the file's data.
513   */
514  data2blobUrl(path) {
515    return URL.createObjectURL(new Blob([fs.readFileSync(app_path+'\\"+path)]))
516  }
517  /**
518   * Converts top, right, bottom, and left margins into x and y coordinates.
519   * @param {number} cont_width - Container width.
520   * @param {number} cont_height - Container height.
521   * @param {number} width - Element width.
522   * @param {number} height - Element height.
523   * @param {number} top - Top margin.
524   * @param {number} right - Right margin.
525   * @param {number} bottom - Bottom margin.
526   * @param {number} left - Left margin.
527   * @returns {Array} Array containing x and y coordinates.
528   */
529  trbl2xy(cont_width, cont_height, width, height, top, right, bottom, left) {
530    var x = 0;
531    var y = 0;
532
533    if(!isUndefined(left)) {
534      x = left;
535    } else if(!isUndefined(right)) {
536      x = cont_width - width - right;
537    }
538    if(!isUndefined(top)) {
539      y = top;
540    } else if(!isUndefined(bottom)) {
541      y = cont_height - height - bottom;
542    }
543    return [x, y];
544  }
545
546  /**
547   * Generates a CSV string from an simple object containing arrays of values.
548   * Each key in the object represents a column in the CSV. This function
549   * handles uneven array lengths by filling missing values with an empty
550   * string.
551   * @param {Object} data - The object containing arrays of data. Each key
552   * will be a column header.
553   * @param {string} [delimiter=','] - The delimiter to use for separating
554   * entries in the CSV (defaults to a comma).
555   * @returns {string} The generated CSV as a string, with each row
556   * representing an entry and each column representing data from the
557   * corresponding key in the input object.
558   */
559  simpleObj2Csv(data, delimiter = ',') {
560    const keys = Object.keys(data);

```

```

555     let csv_content = keys.join(delimiter) + '\n';
556
557     const maxEntries = Math.max(...keys.map(function(key) {return data[key].length;}));
558     for(let i = 0; i < maxEntries; i++) {
559         let row = keys.map(function(key) { return (i < data[key].length) ? data[key][i] : '' }).join(delimiter);
560         csv_content += row + '\n';
561     }
562
563     return csv_content;
564 }
565
566 /**
567 * Converts a 2D array into a CSV string format.
568 * @param {Array} data - An array of arrays to be converted into CSV format.
569 * @param {string} [delimiter=','] - The delimiter to separate the values in the CSV.
570 * @returns {string} The formatted CSV string.
571 */
572 simpleArray2Csv(data, delimiter = ',') {
573     let csv_content = '';
574     const maxEntries = Math.max(...data.map(function(e) {return e.length;}));
575     for(let i = 0; i < maxEntries; i++) {
576         let row = data.map(function(e) { return (i < e.length) ? e[i] : '' }).join(delimiter);
577         csv_content += row + '\n';
578     }
579
580     return csv_content;
581 }
582 }
583
584 exports.PRDC_JSLAB_LIB_CONVERSION = PRDC_JSLAB_LIB_CONVERSION;

```

Listing 88 - conversion.js

```

1 /**
2 * @file JSLAB library device gamepad submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for gamepad.
10 */
11 class PRDC_JSLAB_DEVICE_GAMEPAD {
12
13 /**
14 * Initializes the gamepad device instance.
15 * @param {Object} jsl - Reference to the main JSLAB object.
16 * @param {string} id - Unique identifier for the gamepad.
17 * @param {number} [dt=10] - Data reading interval in milliseconds.
18 */
19 constructor(jsl, id, dt = 10) {

```

```
20  var obj = this;
21  this.jsl = jsl;
22  this.id = id;
23  this.active = false;
24  this.data;
25
26  this.read_gamepad_loop;
27  this.read_gamepad_dt = dt;
28
29  this._checkGamepadFun = function() {
30    obj._checkGamepad();
31  };
32  this.jsl.context.addEventListener("gamepadconnected", obj._checkGamepadFun
33    );
34
35  this.jsl.addForCleanup(this, function() {
36    obj.close();
37  });
38
39  this._checkGamepad();
40
41 /**
42 * Checks if the gamepad is connected and updates its state.
43 */
44 _checkGamepad() {
45   var gamepad = this._getGamepad();
46   if(gamepad) {
47     if(!this.active) {
48       this._onConnect();
49     }
50   }
51 }
52
53 /**
54 * Retrieves the gamepad object if available.
55 * @returns {Gamepad|boolean} The gamepad object if found, otherwise false.
56 */
57 _getGamepad() {
58   var gamepads = this.jsl.env.navigator.getGamepads();
59   for(let i = 0; i < gamepads.length; i++) {
60     var gamepad = gamepads[i];
61     if(gamepad != null) {
62       if(gamepad.id == this.id) {
63         return gamepad.toJSON();
64       }
65     }
66   }
67   return false;
68 }
69
70 /**
71 * Handles gamepad connection events.
72 */
73
```

```

74     _onConnect() {
75         var obj = this;
76
77         // Read loop
78         this.detect_gamepad_loop = clearIntervalIf(this.detect_gamepad_loop);
79         this.active = true;
80         clearIntervalIf(this.read_gamepad_loop);
81         this.read_gamepad_loop = setInterval(function() {
82             var gamepad = obj._getGamepad();
83             if(gamepad) {
84                 if(gamepad.connected) {
85                     obj._onData(gamepad);
86                 } else {
87                     obj._onDisconnect();
88                 }
89             } else {
90                 obj._onDisconnect();
91             }
92         }, this.read_gamepad_dt);
93
94         if(this.jsl.format.isFunction(this.onConnectCallback)) {
95             this.onConnectCallback();
96         }
97     }
98
99     /**
100      * Handles gamepad disconnection events.
101     */
102     _onDisconnect() {
103         this.read_gamepad_loop = clearIntervalIf(this.read_gamepad_loop);
104         this.active = false;
105         this._checkGamepad();
106         if(this.jsl.format.isFunction(this.onDisconnectCallback)) {
107             this.onDisconnectCallback();
108         }
109     }
110
111     /**
112      * Handles incoming gamepad data.
113      * @param {Gamepad} gamepad - The connected gamepad object.
114     */
115     _onData(gamepad) {
116         this.data = gamepad;
117         if(this.jsl.format.isFunction(this.onDataCallback)) {
118             this.onDataCallback(gamepad);
119         }
120     }
121
122     /**
123      * Sets the callback function to handle incoming gamepad data.
124      * @param {Function} callback - Function to execute when data is received.
125     */
126     setOnData(callback) {
127         if(this.jsl.format.isFunction(callback)) {
128             this.onDataCallback = callback;

```

```

129     }
130 }
131
132 /**
133 * Sets the callback function for gamepad connection events.
134 * @param {Function} callback - Function to execute on connection.
135 */
136 setOnConnect(callback) {
137     if(this.jsl.format.isFunction(callback)) {
138         this.onConnectCallback = callback;
139         if(this.active) {
140             this.onConnectCallback();
141         }
142     }
143 }
144
145 /**
146 * Sets the callback function for gamepad disconnection events.
147 * @param {Function} callback - Function to execute on disconnection.
148 */
149 setOnDisconnect(callback) {
150     if(this.jsl.format.isFunction(callback)) {
151         this.onDisconnectCallback = callback;
152     }
153 }
154
155 /**
156 * Cleans up the gamepad instance and stops data reading.
157 */
158 close() {
159     this.active = false;
160     this.read_gamepad_loop = clearIntervalIf(this.read_gamepad_loop);
161     this.jsl.context.removeEventListener("gamepadconnected", this._checkGamepadFun);
162 }
163 }
164
165 exports.PRDC_JSLAB_DEVICE_GAMEPAD = PRDC_JSLAB_DEVICE_GAMEPAD;

```

Listing 89 - device-gamepad.js

```

1 /**
2 * @file JSLAB library device submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8
9 var { PRDC_JSLAB_DEVICE_GAMEPAD } = require('./device-gamepad');
10
11 /**
12 * Class for JSLAB device submodule.
13 */
14 class PRDC_JSLAB_LIB_DEVICE {
15

```

```

16  /**
17   * Constructs a device submodule object with access to JSLAB's device
18   * functions.
19   * @constructor
20   * @param {Object} js1 - Reference to the main JSLAB object.
21   */
22 constructor(js1) {
23   var obj = this;
24   this.js1 = js1;
25
26   this._camera_resolutions = [
27     { "label": "4K (UHD)", "width": 3840, "height": 2160, "ratio": "16:9" },
28     { "label": "1080p (FHD)", "width": 1920, "height": 1080, "ratio": "16:9" },
29     { "label": "720p (HD)", "width": 1280, "height": 720, "ratio": "16:9" },
30     { "label": "480p (VGA)", "width": 640, "height": 480, "ratio": "4:3" },
31     { "label": "360p (nHD)", "width": 640, "height": 360, "ratio": "16:9" },
32     { "label": "240p (QVGA)", "width": 320, "height": 240, "ratio": "4:3" },
33     { "label": "144p (QCIF)", "width": 176, "height": 144, "ratio": "4:3" },
34   ];
35
36 /**
37  * Checks if a specific driver is installed on the system.
38  * @param {string} driver_name - Name of the driver to check.
39  * @returns {boolean} True if the driver is found, false otherwise.
40  */
41 checkDriver(driver_name){
42   var val = this.js1.env.execSync('driverquery');
43   if(val.state == 'success') {
44     var output = val.data.split(/[\r\n]+/);
45     var output_lc = output.map(function(x) { return x.toLowerCase(); });
46     if(!Array.isArray(driver_name)) {
47       driver_name = [driver_name];
48     }
49     driver_name = driver_name.map(function(x) { return x.toLowerCase(); });
50     var data_out = [];
51     for(var i = 0; i < driver_name.length; i++) {
52       var idx = output_lc.map(function(x) { return x.startsWith(driver_name[i]); }).findIndex(function(x) { return x == true; });
53       if(idx >= 0) {
54         data_out[i] = output[idx];
55       } else {
56         data_out[i] = '';
57       }
58     }
59     if(data_out.every(function(x) { return x.length; })) {
60       if(this.js1.debug) {
61         this.js1.env.disp('@checkDriver: ' + data_out);
62       }
63       return true;
64     } else {
65       return false;
66     }
67   }
68 }

```

```

67     } else {
68       if(this.jsl.debug) {
69         this.jsl.env.error('@checkDriver: ' + val.data);
70       }
71       return false;
72     }
73   }
74
75 /**
76 * Checks if the drivers for FTDI devices are installed.
77 * @returns {boolean} True if the drivers are found, false otherwise.
78 */
79 checkDriverFTDI() {
80   return this.checkDriver(['FTDIBUS', 'FTSER2K']);
81 }
82
83 /**
84 * Checks if the drivers for Silicon Labs CP210x USB to UART bridge are
85 * installed.
86 * @returns {boolean} True if the drivers are found, false otherwise.
87 */
88 checkDriverCP210x() {
89   return this.checkDriver('silabser');
90 }
91
92 /**
93 * Checks if the drivers for CH340 USB to serial converter are installed.
94 * @returns {boolean} True if the drivers are found, false otherwise.
95 */
96 checkDriverCH340() {
97   return this.checkDriver('CH341SER_A64');
98 }
99
100 /**
101 * Check if Arduino CLI is available.
102 * @returns {boolean} True if available.
103 */
104 checkArduino() {
105   try {
106     var output = this.jsl.env.execSync('arduino-cli -h', { stdio: 'pipe' });
107     return true;
108   } catch(err) {
109     this.jsl.env.error('@checkArduino: '+language.string(224));
110   }
111 }
112 /**
113 * Compile Arduino project.
114 * @param {string} dir - Project directory.
115 * @returns {Object|boolean} Compilation result or false on error.
116 */
117 compileArduino(dir) {
118   var config_file = this.jsl.env.pathJoin(dir, 'config.json');
119   if(!this.jsl.env.checkFile(config_file)) {
120     this.jsl.env.error('@compileArduino: '+language.string(225));

```

```

121     return false;
122 }
123
124 this.checkArduino();
125 try {
126   var config = JSON.parse(this.jsl.env.readFileSync(config_file))
127   var build_property_str = '';
128   if(config.build_property) {
129     build_property_str = '--build-property ${config.build_property}';
130   }
131   var output = this.jsl.env.spawnSync(`arduino-cli compile --json -b ${config.b} ${build_property_str} ${dir}`, {
132     shell: true,
133     encoding: 'utf8',
134     stdio: 'pipe',
135   });
136   return this.parseArduinoOutput(output);
137 } catch(err) {
138   this.jsl.env.error('@compileArduino: Error: '+err.toString());
139 }
140
141 /**
142 * Upload Arduino project.
143 * @param {string} dir - Project directory.
144 * @param {string} [port] - Optional port.
145 * @returns {Object|boolean} Upload result or false on error.
146 */
147 uploadArduino(dir, port) {
148   var config_file = this.jsl.env.pathJoin(dir, 'config.json');
149   if(!this.jsl.env.isFile(config_file)) {
150     this.jsl.env.error('@uploadArduino: '+language.string(225));
151     return false;
152   }
153
154   this.checkArduino();
155   try {
156     var config = JSON.parse(this.jsl.env.readFileSync(config_file));
157     var build_property_str = '';
158     if(config.build_property) {
159       build_property_str = '--build-property ${config.build_property}';
160     }
161     var port_str = '';
162     if(port) {
163       port_str = '-p ${port}';
164     } else if(config.port) {
165       port_str = '-p ${config.port}';
166     }
167     var output = this.jsl.env.spawnSync(`arduino-cli compile --json -u -t -b
168       ${config.b} ${build_property_str} ${port_str} ${dir}`, {
169       shell: true,
170       encoding: 'utf8',
171       stdio: 'pipe',
172     });
173   return this.parseArduinoOutput(output);

```

```

174     } catch(err) {
175         this.jsl.env.error('@uploadArduino: Error: '+err.toString());
176     }
177 }
178
179 /**
180 * Parses Arduino output from stdout or stderr.
181 * @param {object} output
182 * @returns {object}
183 */
184 parseArduinoOtuput(output) {
185     var data_string;
186     if(output.stdout) {
187         data_string = output.stdout;
188     } else if(output.stderr) {
189         data_string = output.stderr;
190     }
191
192     var data = JSON.parse(data_string);
193     if (!data.success) {
194         this.jsl.env.warn('@parseArduinoOtuput: Compile failed!');
195         this.jsl.env.disp('@parseArduinoOtuput: Compiler error:
196 ${this.format.replaceEditorLinks(data.compiler_err)}');
197     }
198     return data;
199 }
200
201 /**
202 * Retrieves the current state of all connected gamepads.
203 * @returns {Object[]} An array of connected gamepad objects.
204 */
205 getGamepads() {
206     return this.jsl.env.navigator.getGamepads()
207         .filter((g) => !isNull(g))
208         .map(g => g.toJSON());
209 }
210
211 /**
212 * Registers a callback function to be called when a gamepad is connected.
213 * @param {Function} callback - The function to execute when a gamepad
214 * connects.
215 */
216 onGamepadConnected(callback) {
217     var obj = this;
218     var listener = function(e) {
219         if(e.gamepad) {
220             e.gamepad = e.gamepad.map(g => g.toJSON());
221         }
222         callback(e);
223     };
224     this.jsl.env.context.addEventListener("gamepadconnected", listener);
225     this.jsl.addForCleanup(this, function() {
226         obj.jsl.env.context.removeEventListener("gamepadconnected", listener);
227     });
228 }
```

```

228
229  /**
230   * Registers a callback function to be called when a gamepad is disconnected
231   *
232   * @param {Function} callback - The function to execute when a gamepad
233   * disconnects .
234   */
235   onGamepadDisconnected(callback) {
236     var obj = this;
237     var listener = function(e) {
238       if(e.gamepad) {
239         e.gamepad = e.gamepad.map(g => g.toJSON());
240       }
241       callback(e);
242     };
243     this.jsl.env.context.addEventListener("gamepaddirconnected", listener);
244     this.jsl.addForCleanup(this, function() {
245       obj.jsl.env.context.removeEventListener("gamepaddirconnected", listener)
246       ;
247     });
248   }
249
250 /**
251  * Retrieves a specific gamepad by its ID.
252  * @param {number} id - The index of the gamepad to retrieve.
253  * @param {number} dt - Data reading interval in milliseconds.
254  * @returns {PRDC_JSLAB_DEVICE_GAMEPAD} The corresponding gamepad object.
255  */
256 getGamepad(id, dt) {
257   return new PRDC_JSLAB_DEVICE_GAMEPAD(this.jsl, id, dt);
258 }
259
260 /**
261  * Retrieves a list of available webcam (video input) devices.
262  * @returns {Object[]} A promise that resolves to an array of video input
263  * devices .
264  */
265 async getWebcams() {
266   var devices = await this.jsl.env.navigator.mediaDevices.enumerateDevices()
267   ;
268   return devices
269     .filter(device => device.kind === 'videoinput')
270     .map(device => device.toJSON());
271 }
272
273 /**
274  * Retrieves a list of available microphone (audio input) devices.
275  * @returns {Object[]} A promise that resolves to an array of audio input
276  * devices .
277  */
278 async getMicrophones() {
279   var devices = await this.jsl.env.navigator.mediaDevices.enumerateDevices()
280   ;
281   return devices
282     .filter(device => device.kind === 'audioinput')

```

```

276         .map(device => device.toJSON());
277     }
278
279     /**
280      * Retrieves a list of available audio output devices.
281      * @returns {Object[]} A promise that resolves to an array of audio output
282      * devices.
283     */
284     async getAudioOutputs() {
285       var devices = await this.jsl.env.navigator.mediaDevices.enumerateDevices()
286       ;
287       return devices
288         .filter(device => device.kind === 'audiooutput')
289         .map(device => device.toJSON());
290     }
291
292     /**
293      * Opens a new window to display the webcam feed from the specified device.
294      * @param {string} device_id - The unique identifier of the webcam device to
295      * use.
296      * @returns {Promise<WebcamResult>} An object containing the window instance
297      * , video element, and media stream.
298     */
299     async webcam(device_id) {
300       var win = await openWindowBlank();
301       win.setTitle('Webcam');
302       win.document.body.innerHTML += '<video id="video"></video>';
303       var dom = win.document.getElementById('video');
304       Object.assign(dom.style, {
305         position: 'absolute',
306         top: '50%',
307         left: '50%',
308         transform: 'translate(-50%, -50%)',
309         width: '100%',
310         height: '100%',
311         objectFit: 'contain'
312       });
313       try {
314         var constraints = {
315           video: { deviceId: { exact: device_id } },
316           audio: false
317         };
318         var stream = await this.jsl.env.navigator.mediaDevices.getUserMedia(
319           constraints);
320       } catch(err) {
321         this.jsl._console.log(err, constraints);
322         this.jsl.env.error('@capture: '+language.string(222));
323       }
324       dom.srcObject = stream;
325       dom.play();
326       this.jsl.addForCleanup(this, function() {
327         stream.getTracks().forEach(track => track.stop());
328         dom.srcObject = null;
329       });
330       return { win, dom, stream };
331     }
332   }
333 }

```

```

326     }
327
328
329     /**
330      * Initiates webcam video capture.
331      * @param {Object} opts - Configuration options for webcam capture.
332      * @param {function} frameCallback - Callback invoked with each frame's
333      * image data buffer.
334      * @param {function} [editCallback] - Optional callback to edit each frame
335      * before processing.
336      */
337     webcamCapture(opts, frameCallback, editCallback) {
338       opts.type = 'webcam';
339       this.capture(opts, frameCallback, editCallback);
340     }
341
342     /**
343      * Retrieves desktop sources from the current environment.
344      * @returns {DesktopSource[]} An array of desktop sources.
345      */
346     getDesktopSources() {
347       return this.jsl.env.getDesktopSources();
348     }
349
350     /**
351      * Displays the available desktop sources by generating and injecting HTML
352      * elements for each source.
353      * @returns {void}
354      */
355     showDesktopSources() {
356       var html = '';
357       var sources = this.jsl.env.getDesktopSources();
358       for(source of sources) {
359         var { width, height } = source.thumbnail.getSize();
360         html += '<div style="padding:10px; margin: 10px; border: #ccc 1px solid;
361           border-radius: 5px;"><div><b>Name:</b> ' +source.name+ '</div><div><b>Id:</b> ' +source.id+ '</div><div><b>DisplayId:</b> ' +source.
362           display_id+ '</div></img
364           ></div>';
365       }
366       this.jsl.env.disp(html);
367     }
368
369     /**
370      * Initiates desktop screen capture.
371      * @param {Object} opts - Configuration options for desktop capture.
372      * @param {function} frameCallback - Callback invoked with each frame's
373      * image data buffer.
374      * @param {function} [editCallback] - Optional callback to edit each frame
375      * before processing.
376      */
377     desktopCapture(opts, frameCallback, editCallback) {
378       opts.type = 'desktop';
379       this.capture(opts, frameCallback, editCallback);

```

```
371 }
372 /**
373 * Captures media frames based on the provided options.
374 * @param {Object} opts - Configuration options for capturing.
375 * @param {string} opts.id - The ID of the media source.
376 * @param {string} opts.type - Type of capture ('webcam' or 'desktop').
377 * @param {function} frameCallback - Callback invoked with each frame's
378 * image data buffer.
379 * @param {function} [editCallback] - Optional callback to edit each frame
380 * before processing.
381 * @returns {Object} An object containing control functions and resources
382 * for the capture session.
383 * @returns {function} return.stop - Function to stop the capture.
384 * @returns {CanvasRenderingContext2D} return.ctx - The 2D rendering context
385 * of the OffscreenCanvas.
386 * @returns {OffscreenCanvas} return.offscreenCanvas - The OffscreenCanvas
387 * used for rendering frames.
388 * @returns {MediaStreamTrack} return.videoTrack - The video track being
389 * captured.
390 * @returns {MediaStreamTrackProcessor} return.trackProcessor - The
391 * processor for the video track.
392 * @returns {ReadableStreamDefaultReader} return.reader - Reader for the
393 * media stream.
394 */
395 async capture(opts, frameCallback, editCallback) {
396   var { id, type, ...otherOpts } = opts;
397
398   var active = true;
399   var constraints;
400
401   // Define media constraints based on capture type
402   if(type === 'webcam') {
403     constraints = {
404       video: {
405         deviceId: { exact: id },
406         ...otherOpts
407       },
408       audio: false
409     };
410   } else if(type === 'desktop') {
411     constraints = {
412       video: {
413         mandatory: {
414           chromeMediaSource: 'desktop',
415           chromeMediaSourceId: id,
416           ...otherOpts
417         }
418       },
419       audio: false
420     };
421   }
422
423   // Obtain media stream
424   if(isEmptyString(id)) {
425     return Promise.reject('Media source ID is required');
426   }
427
428   const stream = await navigator.mediaDevices.getUserMedia(constraints);
429
430   const videoTrack = stream.getVideoTracks()[0];
431   const trackProcessor = new MediaStreamTrackProcessor(videoTrack);
432
433   const offscreenCanvas = document.createElement('canvas');
434   const ctx = offscreenCanvas.getContext('2d');
435
436   const reader = stream.getReader();
437
438   let frameCount = 0;
439   let lastFrameTime = Date.now();
440
441   const frameCallbackWrapper = frame => {
442     const now = Date.now();
443     const duration = now - lastFrameTime;
444
445     if(duration > 100) {
446       const frameRate = frameCount / duration;
447       console.log(`Captured ${frameCount} frames in ${duration} ms. Frame rate: ${frameRate} fps.`);
448     }
449
450     lastFrameTime = now;
451     frameCount++;
452
453     if(editCallback) {
454       editCallback(frame);
455     }
456
457     if(active) {
458       frameCallback(frame);
459     }
460   }
461
462   const framePromise = reader.read();
463   framePromise.then(frameCallbackWrapper).catch(error => {
464     console.error(error);
465   });
466
467   const stopPromise = stream.getReader().cancel();
468   stopPromise.then(() => {
469     active = false;
470   });
471
472   return {
473     stop: () => {
474       active = false;
475     },
476     ctx,
477     offscreenCanvas,
478     reader,
479     trackProcessor
480   };
481 }
```

```

418     this.jsl.env.error(`@capture: ${language.string(222)}) ;
419   }
420
421   try {
422     var stream = await this.jsl.env.navigator.mediaDevices.getUserMedia(
423       constraints);
424   } catch(err) {
425     this.jsl._console.log(err);
426     this.jsl._console.log(constraints);
427     this.jsl.env.error(`@capture: ${language.string(222)}) ;
428   }
429
430   var videoTrack = stream.getVideoTracks()[0];
431   var settings = videoTrack.getSettings();
432
433   var width = settings.width || 1280;
434   var height = settings.height || 720;
435
436   // Initialize OffscreenCanvas with retrieved width and height
437   var offscreenCanvas = new OffscreenCanvas(width, height);
438   var ctx = offscreenCanvas.getContext('2d', {willReadFrequently: true});
439
440   /**
441    * Stops the frame capture by setting active to false.
442    */
443   function stop() {
444     active = false;
445   }
446
447   /**
448    * Continuously reads frames from the media stream, processes them, and
449    * invokes callbacks.
450    */
451   async function getFrames() {
452     while(active) {
453       var { value, done } = await reader.read();
454       if(done) {
455         break;
456       }
457
458       var frame = value;
459       ctx.drawImage(frame, 0, 0, width, height);
460
461       if(typeof editCallback === 'function') {
462         editCallback(frame, width, height);
463       }
464
465       var imageData = ctx.getImageData(0, 0, width, height);
466       frameCallback(imageData.data.buffer, width, height, frame);
467       frame.close();
468     }
469
470     // Cleanup after capturing frames
471     videoTrack.stop();
472     reader.releaseLock();

```

```

471     stream.getTracks().forEach(track => track.stop());
472 }
473
474 // Initialize MediaStreamTrackProcessor and reader
475 var trackProcessor = new MediaStreamTrackProcessor({ track: videoTrack });
476 var reader = trackProcessor.readable.getReader();
477
478 this.jsl.addForCleanup(this, stop);
479
480 // Start processing frames
481 getFrames();
482
483 // Return control functions and resources
484 return { stop, ctx, offscreenCanvas, videoTrack, trackProcessor, reader };
485 }
486
487 /**
488 * Gets supported camera resolutions for a specific device.
489 * @param {string} device_id - The camera device ID.
490 * @returns {Promise<Array<Object>>} Supported resolutions.
491 */
492 async getCameraResolutions(device_id) {
493   const resolutions = [];
494   for(const resolution of this._camera_resolutions) {
495     var constraints = {
496       audio: false,
497       video: {
498         deviceId: { exact: device_id },
499         width: { exact: resolution.width },
500         height: { exact: resolution.height }
501       }
502     };
503     try {
504       const stream = await this.jsl.env.navigator.mediaDevices.getUserMedia(
505         constraints);
506       stream.getTracks().forEach(track => track.stop());
507       resolutions.push(resolution);
508     } catch(err) {
509       this.jsl._console.log(err);
510     };
511   }
512   return resolutions;
513 }
514 /**
515 * Displays an audio waveform on the canvas.
516 * @param {string} device_id - The microphone device ID.
517 * @param {number} [fftSize=2048] - FFT size for analysis.
518 * @returns {Object} Controls to stop or reset the waveform.
519 */
520 async showAudioWaveform(device_id, fftSize = 2048) {
521   var obj = this;
522
523   var win = await openCanvas();
524   win.setTitle('Audio Waveform');

```

```
525     var draw_loop;
526     var audio_ctx = new AudioContext();
527     var analyser = audio_ctx.createAnalyser();
528     analyser.fftSize = fftSize;
529     var buffer_length = analyser.frequencyBinCount;
530     var data = new Uint8Array(buffer_length);
531
532     var canvas = win.canvas;
533     var canvas_ctx = canvas.getContext("2d");
534     var canvas_width = 1000;
535     var canvas_height = 500;
536     canvas.width = canvas_width;
537     canvas.height = canvas_height;
538     canvas.style.width = '100vw';
539     canvas.style.height = '100vh';
540
541     canvas_ctx.lineWidth = 1;
542     canvas_ctx.strokeStyle = "#000";
543     var slice_width = canvas_width / buffer_length;
544     reset();
545
546     var constraints = {
547       audio: {
548         deviceId: { exact: device_id }
549       }
550     };
551     var stream = await this.jsl.env.navigator.mediaDevices.getUserMedia(
552       constraints);
553     var source = audio_ctx.createMediaStreamSource(stream);
554     source.connect(analyser);
555
556     _update();
557
558     function _update() {
559       draw_loop = obj.jsl.context.requestAnimationFrame(function() {
560         _update();
561       });
562       analyser.getByteTimeDomainData(data);
563
564       canvas_ctx.clearRect(0, 0, canvas_width, canvas_height);
565       canvas_ctx.beginPath();
566
567       let x = 0;
568       for(let i = 0; i < buffer_length; i++) {
569         const v = data[i] / 128.0;
570         const y = v * (canvas_height / 2);
571
572         if(i === 0) {
573           canvas_ctx.moveTo(x, y);
574         } else {
575           canvas_ctx.lineTo(x, y);
576         }
577
578         x += slice_width;
```

```

579     }
580
581     canvas.ctx.lineTo(canvas.width, canvas.height / 2);
582     canvas.ctx.stroke();
583 }
584
585 function stop() {
586     obj.jsl.context.cancelAnimationFrame(draw_loop);
587     reset();
588 }
589
590 function reset() {
591     canvas.ctx.clearRect(0, 0, canvas.width, canvas.height);
592     canvas.ctx.beginPath();
593     canvas.ctx.moveTo(0, canvas.height/2);
594     canvas.ctx.lineTo(canvas.width, canvas.height / 2);
595     canvas.ctx.stroke();
596 }
597
598     this.jsl.addForCleanup(this, stop);
599
600     return { win, stop, reset };
601 }
602
603 /**
604 * Records video from the specified canvas element, webcam deviceId, or
605 * desktop sourceId and returns a MediaRecorder augmented with an async
606 * stopRecording() that finalizes and saves the file.
607 * @param {((HTMLCanvasElement|string))} source - Canvas element, webcam
608 * deviceId, or desktop sourceId to capture.
609 * @param {Object} [opts={}] - Optional settings: type ('canvas' | 'webcam'
610 * | 'desktop'), fps, mimeType, and videoBitsPerSecond.
611 * @returns {MediaRecorder} - MediaRecorder that streams the capture and
612 * provides a helper to stop and save.
613 */
614 async startVideoRecording(source, opts = {}) {
615     var obj = this;
616     var preferredMime = 'video/mp4; codecs="avc1.640028"';
617     var fallbackMime = 'video/webm; codecs=vp9,opus';
618
619     var standardMimeType = MediaRecorder.isTypeSupported(preferredMime) ?
620         preferredMime : fallbackMime;
621
622     var stream;
623     if(opts.type == 'canvas') {
624         stream = source.captureStream(opts.fps);
625     } else {
626         var constraints;
627         if(opts.type === 'webcam') {
628             constraints = {
629                 video: {
630                     deviceId: { exact: source },
631                     ... opts.constraints_opts
632                 },
633                 audio: false
634             };
635         }
636     }
637 }
```

```

629     };
630   } else {
631     constraints = {
632       video: {
633         mandatory: {
634           chromeMediaSource: 'desktop',
635           chromeMediaSourceId: source
636         },
637         ... opts.constraints_opts
638       },
639       audio: false
640     };
641   }
642   try {
643     stream = await this.jsl.env.navigator.mediaDevices.getUserMedia(
644       constraints);
645   } catch(err) {
646     this.jsl._console.log(err);
647     this.jsl._console.log(constraints);
648     this.jsl.env.error('@startVideoRecording: '+language.string(222));
649   }
650 }
651 var mimeType = opts.mimeType || standardMimeType;
652 var ext = mimeType.includes('mp4') ? 'mp4' : 'webm';
653 var recorder = new MediaRecorder(stream, {
654   mimeType: mimeType,
655   videoBitsPerSecond: opts.videoBitsPerSecond || 8_000_000
656 });
657
658 var chunks = [];
659 recorder.ondataavailable = function(e) {
660   chunks.push(e.data)
661 }
662
663 recorder.onstop = async function() {
664   var blob = new Blob(chunks, { type: mimeType });
665   var buffer = Buffer.from(await blob.arrayBuffer());
666
667   var options = {
668     title: language.currentString(236),
669     buttonLabel: language.currentString(236),
670     filters:[
671       {name: ext, extensions: [ext]},
672     ]
673   };
674   var video_path = obj.jsl.env.showSaveDialogSync(options);
675   if(video_path) {
676     obj.jsl.env.writeFileSync(video_path, buffer);
677   }
678 }
679
680 var recording = true;
681 recorder.start(1000);
682

```

```

683     recorder.stopRecording = async function() {
684       if(recording) {
685         recording = false;
686         recorder.stop();
687       }
688     }
689     return recorder;
690   }
691 }
692
693 exports.PRDC_JSLAB_LIB_DEVICE = PRDC_JSLAB_LIB_DEVICE;

```

Listing 90 - device.js

```

1  /**
2   * @file JSLAB library figures submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB figures submodule.
10  */
11 class PRDC_JSLAB_LIB FIGURES {
12
13 /**
14  * Initializes a new instance of the figures submodule.
15  * @param {Object} jsl Reference to the main JSLAB object.
16  */
17 constructor(jsl) {
18   var obj = this;
19   this.jsl = jsl;
20
21   this._fonts_registered = false;
22   this._fonts = [];
23   this._fid = 0;
24   this._pid = 0;
25   this._html_figure = this.jsl.env.readFileSync(app_path + '/html/
26     html_figure.html').toString();
26   this._i_html_figure = this.jsl.env.readFileSync(app_path + '/html/
27     i_html_figure.html').toString();
27   this._io_html_figure = this.jsl.env.readFileSync(app_path + '/html/
28     io_html_figure.html').toString();
29
29 /**
30  * Array of open figures.
31  * @type {Array}
32  */
33 this.open_figures = {};
34
35 /**
36  * Current active figure ID.
37  * @type {Number}
38  */
39 this.active_figure = -1;

```

```

40 }
41
42 /**
43 * Opens or updates a figure with specified options.
44 * @param {Number} id Identifier for the figure.
45 * @returns {Number} The identifier of the opened or updated figure.
46 */
47 figure(fid) {
48   if (!(fid >= 0) || !this.open_figures.hasOwnProperty(fid)) {
49     if (!(fid >= 0)) {
50       this._fid += 1;
51       fid = this._fid;
52     }
53     this.open_figures[fid] = new PRDC_JSLAB FIGURE(this.jsl, fid);
54     this.open_figures[fid].init();
55   } else {
56     this.open_figures[fid].focus();
57   }
58   this._ setActiveFigure(fid);
59   this.jsl.no_ans = true;
60   this.jsl.ignore_output = true;
61   return fid;
62 }
63
64 /**
65 * Retrieves the figure object associated with the specified figure ID.
66 * @param {string} fid – The identifier of the figure to retrieve.
67 * @returns {(Object|boolean)} The figure object if found, otherwise ‘false’.
68 */
69 getFigure(fid) {
70   if (this.open_figures.hasOwnProperty(fid)) {
71     return this.open_figures[fid];
72   } else {
73     return false;
74   }
75 }
76
77 /**
78 * Retrieves the window of the figure object associated with the specified
79 * figure ID.
80 * @param {string} fid – The identifier of the figure to retrieve.
81 * @returns {(Object|boolean)} The figure object if found, otherwise ‘false’.
82 */
83 getFigureWindow(fid) {
84   if (this.open_figures.hasOwnProperty(fid)) {
85     return this.open_figures[fid].win;
86   } else {
87     return false;
88   }
89
90 /**
91 * Retrieves current active figure object.

```

```
92     * @returns {((Object|boolean))} The figure object if found, otherwise 'false
 93     *
 94     */
 95    getCurrentFigure() {
 96      if(this.open_figures.hasOwnProperty(this.active_figure)) {
 97        return this.open_figures[this.active_figure];
 98      } else {
 99        return false;
100      }
101  }
102 /**
103 * Retrieves current active figure object.
104 * @returns {((Object|boolean))} The figure object if found, otherwise 'false
 105 *
 106 */
 107 gcf() {
 108   return this.getCurrentFigure();
 109 }
110 /**
111 * Retrieves the plot object for a specified figure ID.
112 * @param {string} fid - The identifier of the figure to retrieve the plot
 113 * for.
 114 * @returns {((Object|boolean))} The plot object if it exists, otherwise 'false'.
 115 */
 116 getPlot(fid) {
 117   if(this.open_figures.hasOwnProperty(fid) && this.open_figures[fid].plot) {
 118     return this.open_figures[fid].plot;
 119   } else {
 120     return false;
 121   }
 122 }
123 /**
124 * Retrieves the plot object for a specified figure ID.
125 * @param {string} fid - The identifier of the figure to retrieve the plot
 126 * for.
 127 * @returns {((Object|boolean))} The plot object if it exists, otherwise 'false'.
 128 */
 129 getAxes(fid) {
 130   return this.getPlot(fid);
 131 }
132 /**
133 * Retrieves plot from current active figure object.
134 * @returns {((Object|boolean))} The figure object if found, otherwise 'false
 135 *
 136 */
 137 getCurrentPlot() {
 138   if(this.open_figures.hasOwnProperty(this.active_figure)) {
 139     return this.open_figures[this.active_figure].plot;
 140   } else {
```

```
140         return false;
141     }
142 }
143 /**
144 * Retrieves plot from current active figure object.
145 * @returns {(Object|boolean)} The figure object if found, otherwise 'false'
146 *
147 */
148 gcp() {
149     return this.getCurrentPlot();
150 }
151 /**
152 * Retrieves plot from current active figure object.
153 * @returns {(Object|boolean)} The figure object if found, otherwise 'false'
154 *
155 */
156 getCurrentAxes() {
157     return this.getCurrentPlot();
158 }
159 /**
160 * Retrieves plot from current active figure object.
161 * @returns {(Object|boolean)} The figure object if found, otherwise 'false'
162 *
163 */
164 gca() {
165     return this.getCurrentPlot();
166 }
167 /**
168 * Brings the specified figure to the foreground.
169 * @param {number} fid - The ID of the figure to focus.
170 * @returns {boolean|undefined} - Returns false if the figure ID is invalid.
171 *
172 */
173 focusFigure(fid) {
174     if(this.open_figures.hasOwnProperty(fid)) {
175         return this.open_figures[fid].focus();
176     } else {
177         return false;
178     }
179 }
180 /**
181 * Sets the size of a specified figure.
182 * @param {number} fid - The ID of the figure.
183 * @param {number} width - The new width of the figure.
184 * @param {number} height - The new height of the figure.
185 * @returns {boolean|undefined} - Returns false if the figure ID is invalid.
186 *
187 */
188 setFigureSize(fid, width, height) {
189     if(this.open_figures.hasOwnProperty(fid)) {
190         return this.open_figures[fid].setSize(width, height);
191     } else {
```

```
192         return false;
193     }
194 }
195 /**
196 * Sets the position of a specified figure.
197 * @param {number} fid - The ID of the figure.
198 * @param {number} left - The new left position of the figure.
199 * @param {number} top - The new top position of the figure.
200 * @returns {boolean|undefined} - Returns false if the figure ID is invalid.
201 */
202 setFigurePos(fid, left, top) {
203     if(this.open_figures.hasOwnProperty(fid)) {
204         return this.open_figures[fid].setPos(left, top);
205     } else {
206         return false;
207     }
208 }
209 /**
210 * Sets the title of the specified figure.
211 * @param {string} fid - The figure ID.
212 * @param {string} title - The new title for the figure.
213 * @returns {boolean| *} The result of setting the title, or false if the
214 * figure does not exist.
215 */
216 setFigureTitle(fid, title) {
217     if(this.open_figures.hasOwnProperty(fid)) {
218         return this.open_figures[fid].setTitle(title);
219     } else {
220         return false;
221     }
222 }
223 /**
224 * Retrieves the size of a specified figure.
225 * @param {number} fid - The ID of the figure.
226 * @returns {Array|boolean} - Returns an array [width, height] or false if
227 * the figure ID is invalid.
228 */
229 getSize(fid) {
230     if(this.open_figures.hasOwnProperty(fid)) {
231         return this.open_figures[fid].getSize();
232     } else {
233         return false;
234     }
235 }
236 /**
237 * Retrieves the position of a specified figure.
238 * @param {number} fid - The ID of the figure.
239 * @returns {Array|boolean} - Returns an array [left, top] or false if the
240 * figure ID is invalid.
241 */
242 getFigurePos(fid) {
```

```
244     if (this.open_figures.hasOwnProperty(fid)) {
245         return this.open_figures[fid].getPos();
246     } else {
247         return false;
248     }
249 }
250
251 /**
252 * Retrieves the media source id of a specified figure.
253 * @param {number} fid - The ID of the figure.
254 * @returns {String|boolean} - Returns Media source id or false if the
255 * figure ID is invalid.
256 */
257 getFigureMediaSourceId(fid) {
258     if (this.open_figures.hasOwnProperty(fid)) {
259         return this.open_figures[fid].getMediaSourceId();
260     } else {
261         return false;
262     }
263 }
264 /**
265 * Starts video recording of a specified figure.
266 * @param {number} fid - The ID of the figure.
267 * @param {Object} - Optional settings.
268 * @returns {Object|boolean} - Returns recorder object or false if the
269 * figure ID is invalid.
270 */
271 startFigureVideoRecording(fid, opts) {
272     if (this.open_figures.hasOwnProperty(fid)) {
273         return this.open_figures[fid].startVideoRecording(opts);
274     } else {
275         return false;
276     }
277 }
278 /**
279 * Closes a specified figure.
280 * @param {number} fid - The ID of the figure to close.
281 * @returns {boolean|undefined} - Returns false if the figure ID is invalid.
282 */
283 closeFigure(fid) {
284     if (this.open_figures.hasOwnProperty(fid)) {
285         return this.open_figures[fid].close();
286     } else {
287         return false;
288     }
289 }
290
291 /**
292 * Closes a figure or window by its identifier.
293 * @param {number|string} id - The identifier of the figure or window to
294 * close. Use "all" to close all.
295 * @param {string} [type='figure'] - The type of object to close ('figure'
296 * or 'window').
```

```

295     */
296     close(id, type = 'figure') {
297       if(id == "all") {
298         this.jsl.env.closeWindow(id);
299       } else {
300         if(type == 'window') {
301           this.jsl.env.closeWindow(id);
302         } else if(type == 'figure') {
303           this.jsl.env.closeFigure(id);
304         }
305       }
306       this.jsl.no_ans = true;
307       this.jsl.ignore_output = true;
308     }
309
310 /**
311 * Opens a dialog for saving a figure in various formats.
312 * @param {String} fid - The figure identifier.
313 */
314 async saveFileDialog(fid) {
315   let options = {
316     title: language.currentString(143),
317     defaultPath: fid + '.svg',
318     buttonLabel: language.currentString(143),
319     filters:[
320       {name: 'svg', extensions: ['svg']},
321       {name: 'pdf', extensions: ['pdf']},
322       {name: 'png', extensions: ['png']},
323       {name: 'jpg', extensions: ['jpg', 'jpeg']},
324       {name: 'webp', extensions: ['webp']},
325       {name: 'json', extensions: ['json']},
326       {name: 'static html', extensions: ['html']},
327       {name: 'interactive html', extensions: ['i.html']},
328       {name: 'interactive offline html', extensions: ['io.html']}
329     ]
330   };
331   var figure_path = this.jsl.env.showSaveDialogSync(options);
332   if(figure_path) {
333     await this.saveFigure(fid, figure_path);
334   }
335 }
336
337 /**
338 * Saves a figure to a specified path in various formats.
339 * @param {String} fid - The figure identifier.
340 * @param {String} figure_path - The path where the figure should be saved.
341 * @param {Array} size - Optional dimensions [width, height] to use if
342   saving as a PDF.
343 */
344 async saveFigure(fid, figure_path, size) {
345   var pdf_flag = false;
346   var html_flag = false;
347   var ext = figure_path.split('.').pop();
348   if(['svg', 'pdf', 'png', 'jpg', 'jpeg', 'webp', 'html', 'json'].includes(
349     ext)) {

```

```

348   if(ext === 'json' || figure_path.endsWith('.i.html') || figure_path.
349     endsWith('.io.html')) {
350     var data = this.open_figures[fid].context.plot.toJSON();
351     if(figure_path.endsWith('.i.html')) {
352       var html = this._i_html_figure.replaceAll('%title%', this.
353         open_figures[fid].dom.title);
354       data = html.replace('%figure_data%', data);
355     } else if(figure_path.endsWith('.io.html')) {
356       var html = this._io_html_figure.replaceAll('%title%', this.
357         open_figures[fid].dom.title);
358       data = html.replace('%figure_data%', data);
359     }
360     this.jsl.env.writeFileSync(figure_path, data);
361     return;
362   } else if(ext === 'jpg') {
363     ext = 'jpeg';
364   } else if(ext === 'pdf') {
365     pdf_flag = true;
366     ext = 'svg';
367   } else if(ext === 'html') {
368     html_flag = true;
369     ext = 'svg';
370   }

371   var data_url = await this.open_figures[fid].context.plot.toImage(ext,
372     size);
373   var data;
374   if(ext === 'svg') {
375     if(html_flag) {
376       var html = this._html_figure.replaceAll('%title%', this.open_figures
377         [fid].dom.title);
378       data = html.replace('%image_source%', data_url);
379     } else {
380       data = decodeURIComponent(data_url.replace('data:image/svg+xml,', '')) ;
381     }
382     data = data.replace(/\bLatinModern\b/g, "LatinModernMath");
383   } else {
384     data_url = data_url.replace('data:image/png;base64,', '');
385     data_url = data_url.replace('data:image/jpeg;base64,', '');
386     data_url = data_url.replace('data:image/webp;base64,', '');
387     data = new Buffer(data_url, 'base64');
388   }
389   if(pdf_flag) {
390     var pdf_data = await this._svg2pdf(fid, data, size);
391     try {
392       this.jsl.env.writeFileSync(figure_path, pdf_data);
393     } catch(err) {
394       this.jsl.env.error('@saveFigure: '+err.stack);
395     }
396   } else {
397     try {
398       this.jsl.env.writeFileSync(figure_path, data);
399     } catch(err) {
400       this.jsl.env.error('@saveFigure: '+err.stack);
401     }
402   }
403 }

```

```
397         }
398     }
399   } else {
400     this.jsl.env.error('@saveFigure: '+language.string(124));
401   }
402 }
403
404 /**
405 * Sets the label for the x-axis of the active figure.
406 * @param {String} label - The label for the x-axis.
407 */
408 legend(state) {
409   if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
410   {
411     this.open_figures[this.active_figure].plot.legend(state);
412   }
413   this.jsl.no_ans = true;
414   this.jsl.ignore_output = true;
415 }
416 /**
417 * Sets the label for the x-axis of the active figure.
418 * @param {String} label - The label for the x-axis.
419 */
420 xlabel(label) {
421   if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
422   {
423     this.open_figures[this.active_figure].plot.xlabel(label);
424   }
425   this.jsl.no_ans = true;
426   this.jsl.ignore_output = true;
427 }
428 /**
429 * Sets the label for the y-axis of the active figure.
430 * @param {String} label - The label for the y-axis.
431 */
432 ylabel(label) {
433   if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
434   {
435     this.open_figures[this.active_figure].plot.ylabel(label);
436   }
437   this.jsl.no_ans = true;
438   this.jsl.ignore_output = true;
439 }
440 /**
441 * Sets the label for the z-axis of the active figure.
442 * @param {String} label - The label for the z-axis.
443 */
444 zlabel(label) {
445   if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
446   {
447     this.open_figures[this.active_figure].plot.zlabel(label);
448   }
449 }
```

```
448     this.jsl.no_ans = true;
449     this.jsl.ignore_output = true;
450 }
451 /**
452 * Sets the title of the active figure.
453 * @param {String} label - The title text.
454 */
455 title(label) {
456     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
457     {
458         this.open_figures[this.active_figure].plot.title(label);
459     }
460     this.jsl.no_ans = true;
461     this.jsl.ignore_output = true;
462 }
463 /**
464 * Sets the xlim of the active figure.
465 * @param {String} lim - x limits.
466 */
467 xlim(lim) {
468     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
469     {
470         this.open_figures[this.active_figure].plot.xlim(lim);
471     }
472     this.jsl.no_ans = true;
473     this.jsl.ignore_output = true;
474 }
475 /**
476 * Sets the ylim of the active figure.
477 * @param {String} lim - y limits.
478 */
479 ylim(lim) {
480     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
481     {
482         this.open_figures[this.active_figure].plot.ylim(lim);
483     }
484     this.jsl.no_ans = true;
485     this.jsl.ignore_output = true;
486 }
487 /**
488 * Sets the zlim of the active figure.
489 * @param {String} lim - z limits.
490 */
491 zlim(lim) {
492     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
493     {
494         this.open_figures[this.active_figure].plot.zlim(lim);
495     }
496     this.jsl.no_ans = true;
497     this.jsl.ignore_output = true;
498 }
```

```
499
500  /**
501   * Adjusts the view based on azimuth and elevation angles.
502   * @param {number} azimuth - The azimuth angle.
503   * @param {number} elevation - The elevation angle.
504   */
505   view(azimuth, elevation) {
506     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
507     {
508       this.open_figures[this.active_figure].plot.view(azimuth, elevation);
509     }
510     this.jsl.no_ans = true;
511     this.jsl.ignore_output = true;
512   }
513
514  /**
515   * Adjusts the zoom based on zoom factor.
516   * @param {number} factor - The zoom factor.
517   */
518   zoom(factor) {
519     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
520     {
521       this.open_figures[this.active_figure].plot.zoom(factor);
522     }
523     this.jsl.no_ans = true;
524     this.jsl.ignore_output = true;
525   }
526
527  /**
528   * Applies the specified style to the active figure's plot axis.
529   * @param {Object} style - The style configuration to apply to the axis.
530   */
531   axis(style) {
532     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
533     {
534       this.open_figures[this.active_figure].plot.axis(style);
535     }
536     this.jsl.no_ans = true;
537     this.jsl.ignore_output = true;
538   }
539
540  /**
541   * Prints the currently active figure to a file.
542   * @param {String} filename - The name of the file where the figure should
543   *   be printed.
544   * @param {Object} options - Printing options.
545   */
546   async printFigure(filename, options) {
547     if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
548     {
549       await this.open_figures[this.active_figure].plot.print(filename, options
550     );
551     }
552     this.jsl.no_ans = true;
553     this.jsl.ignore_output = true;
```

```

548     }
549
550     /**
551      * Plots data on the active figure.
552      * @param {Array} traces - Data traces to plot.
553      * @param {Object} options - Configuration options for plotting.
554      * @returns {Number} The plot identifier.
555      */
556     plot(traces, options) {
557       var fid = this.active_figure;
558       if(options == undefined) {
559         options = {};
560       } else {
561         if(options.hasOwnProperty(fid)) {
562           fid = options.fid;
563         }
564       }
565       this._pid += 1;
566       var id = this._pid;
567
568       fid = this.figure(fid);
569       this.open_figures[fid]._newPlot(id, traces, options);
570
571       this.jsl.no_ans = true;
572       this.jsl.ignore_output = true;
573       return this.open_figures[fid].plot;
574     }
575
576     /**
577      * Updates plot data by delegating to the 'update' method.
578      * @param {Object} traces - The trace data to be updated in the plot.
579      * @param {number} N - The data length or index for updating the plot.
580      */
581     updatePlot(traces, N) {
582       if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
583       {
584         this.open_figures[this.active_figure].plot.update(data);
585       }
586       this.jsl.no_ans = true;
587       this.jsl.ignore_output = true;
588     }
589
590     /**
591      * Updates plot data by id by delegating to the 'updateById' method.
592      * @param {Object|Object[]} data - Trace update object(s) to apply to the
593      * active plot.
594      */
595     updatePlotById(data) {
596       if(this.active_figure >= 0 && this.open_figures[this.active_figure].plot)
597       {
598         this.open_figures[this.active_figure].plot.updateById(data);
599       }
600       this.jsl.no_ans = true;
601       this.jsl.ignore_output = true;
602     }

```

```

600
601  /**
602   * Hides figure menu.
603   */
604 hideFigureMenu() {
605   if(this.active_figure >= 0) {
606     this.open_figures[this.active_figure].hideMenu();
607   }
608   this.jsl.no_ans = true;
609   this.jsl.ignore_output = true;
610 }
611
612
613 /**
614 * Shows figure menu.
615 */
616 showFigureMenu() {
617   if(this.active_figure >= 0) {
618     this.open_figures[this.active_figure].showMenu();
619   }
620   this.jsl.no_ans = true;
621   this.jsl.ignore_output = true;
622 }
623
624 /**
625 * Loads figure from JSON file
626 * @param {String} file_path - Absolute or relative path to the JSON file of
627 *   figure.
628 * @param {Number} id Identifier for the figure.
629 * @returns {Number} The identifier of the opened or updated figure.
630 */
631 loadJsonFigure(fid, file_path) {
632   if(!file_path) {
633     var options = {
634       title: language.currentString(247),
635       buttonLabel: language.currentString(231)
636     };
637     file_path = this.jsl.env.showOpenDialogSync(options);
638     if(file_path === undefined) {
639       this.jsl.env.error('loadJsonFigure: '+language.string(132)+'.');
640       return false;
641     } else {
642       file_path = file_path[0];
643     }
644   }
645   if(!this.jsl.file_system.existFile(file_path)) {
646     this.jsl.env.error('@loadJsonFigure: '+language.string(248));
647     return false;
648   }
649   var data = JSON.parse(this.jsl.env.readFileSync(file_path).toString());
650   this._pid += 1;
651
652   fid = this.figure(fid);
653   this.open_figures[fid]._fromJSON(this._pid, data);

```

```

654     return fid ;
655   }
656
657   /**
658    * Updates the language of the text elements within all open figures .
659    */
660   _updateLanguage() {
661     Object.values(this.open_figures).forEach(function(figure) {
662       figure._updateLanguage(false);
663     });
664   }
665
666   /**
667    * Sets the specified figure as the active figure .
668    * @param {string} fid - The identifier of the figure to set as active .
669    */
670   setActiveFigure(fid) {
671     if(this.open_figures.hasOwnProperty(fid)) {
672       this.active_figure = fid;
673     } else {
674       this.active_figure = -1;
675     }
676   }
677
678   /**
679    * Closes a figure identified by the given ID and updates the active figure
680    * if needed .
681    * @param {string} id - The identifier of the figure to close .
682    */
683   closedFigure(fid) {
684     if(this.open_figures.hasOwnProperty(fid)) {
685       var new_fid = -1;
686       if(this.active_figure === fid) {
687         var fids = Object.keys(this.open_figures);
688         var N = fids.length;
689         if(N > 1) {
690           if(fids[N-1] !== fid) {
691             new_fid = fids[N-1];
692           } else {
693             new_fid = fids[N-2];
694           }
695           this._setActiveFigure(new_fid);
696         }
697         delete this.open_figures[fid];
698       }
699     }
700
701   /**
702    * Reads and returns font data from a specified path .
703    * @param {String} font_path Path to the font file .
704    * @returns {Buffer} The font data .
705    */
706   _getFontData(font_path) {
707     try {

```

```

708     return this.jsl.env.readFileSync(font_path);
709 } catch(err) {
710     this.jsl.env.error('@getFontData: '+err.stack);
711 }
712 return false;
713 }
714 /**
715 * Registers fonts for use in figures.
716 */
717 _registerFonts() {
718     if(!this._fonts_registered) {
719         this._fonts.push(this._getFontData(
720             app_path+'/font/roboto-v20-latin-ext_latin_greek-ext_greek_cyrillic-
721             ext_cyrillic-regular.ttf',
722         ));
723         this._fonts.push(this._getFontData(
724             app_path+'/font/latinmodern-math.otf',
725         ));
726         this._fonts_registered = true;
727     }
728 }
729 /**
730 * Converts SVG data to PDF format.
731 * @param {String} fid - The figure identifier.
732 * @param {String} data - The SVG data to convert.
733 * @param {Array} size - The dimensions [width, height] to use for the PDF.
734 * @returns {Promise<Buffer>} A promise that resolves with the generated PDF
735     data.
736 */
737 _svg2pdf(fid, data, size) {
738     var obj = this;
739     this._registerFonts();
740     var plot_cont = this.open_figures[fid].dom.querySelector('#figure-content
741         .plot-cont');
742     var width = plot_cont.clientWidth;
743     var height = plot_cont.clientHeight;
744     if(typeof size != 'undefined') {
745         width = size[0];
746         height = size[1];
747     }
748     return new Promise(function(resolve) {
749         var doc = new obj.jsl.env.PDFDocument({
750             size: [width, height]
751         });
752         doc.registerFont('Roboto', obj._fonts[0]);
753         doc.registerFont('LatinModernMath', obj._fonts[1]);
754         obj.jsl.env.SVGtoPDF(doc, data, 0, 0, {
755             width: width,
756             height: height,
757             assumePt: true
758         });
759         var buf = [];
760         doc.on('data', buf.push.bind(buf));

```

```

760     doc.on('end', function() {
761       var pdfData = Buffer.concat(buf);
762       resolve(pdfData);
763     });
764     doc.end();
765   });
766 }
767 }
768
769 exports.PRDC_JSLAB_LIB FIGURES = PRDC_JSLAB_LIB FIGURES;
770
771 /**
772 * Represents an individual figure within the JSLAB environment, providing
773 * detailed configuration and interaction capabilities.
774 */
775 class PRDC_JSLAB FIGURE {
776
777   #jsl;
778
779   /**
780    * Initializes a new instance of a JSLAB figure.
781    * @param {Object} js1 Reference to the main JSLAB object.
782    * @param {Number} id Identifier for the figure.
783    * @param {Object} options Configuration options for the figure.
784    */
785   constructor(js1, fid) {
786     var obj = this;
787
788     this.#jsl = js1;
789     this.fid = fid;
790
791     this.wid;
792     this.context;
793     this.dom;
794     this.fig_ready = false;
795     this.opened = false;
796
797     this.plot = undefined;
798
799     this.ready = new Promise((resolve) => {
800       obj._readyResolve = resolve;
801     });
802
803     this.wid = this.#jsl.windows.openWindow('figure.html');
804     this.#jsl.windows.open_windows[this.wid].onClosed = function() {
805       obj.#jsl.figures._closedFigure(obj.fid);
806     }
807   }
808
809   /**
810    * Initializes figure.
811    */
812   async init() {
813     if(!this.opened) {

```

```
814     await this.#jsl.windows.open_windows[this.wid].ready;
815     await this._onReady();
816     this.opened = true;
817 }
818 }
819
820 /**
821 * Brings the figure window to the foreground.
822 */
823 async focus() {
824     await this.#jsl.promiseOrStoped(this.ready);
825     return await this.#jsl.windows.open_windows[this.wid].focus();
826 }
827
828 /**
829 * Sets the size of the window.
830 * @param {number} width - The desired width of the window.
831 * @param {number} height - The desired height of the window.
832 * @returns {Promise} - Resolves when the window size is set.
833 */
834 async setSize(width, height) {
835     await this.#jsl.promiseOrStoped(this.ready);
836     return await this.#jsl.windows.open_windows[this.wid].setSize(width,
837         height);
838 }
839 /**
840 * Sets the position of the window.
841 * @param {number} left - The desired left position of the window.
842 * @param {number} top - The desired top position of the window.
843 * @returns {Promise} - Resolves when the window position is set.
844 */
845 async setPos(left, top) {
846     await this.#jsl.promiseOrStoped(this.ready);
847     return await this.#jsl.windows.open_windows[this.wid].setPos(left, top);
848 }
849
850 /**
851 * Sets the title of the current window.
852 * @param {string} title - The new title for the window.
853 * @returns {Promise<*>} A promise that resolves when the title is set.
854 */
855 async setTitle(title) {
856     await this.#jsl.promiseOrStoped(this.ready);
857     return await this.#jsl.windows.open_windows[this.wid].setTitle(title);
858 }
859
860 /**
861 * Retrieves the size of the window.
862 * @returns {Promise<Array>} - Resolves with an array [width, height].
863 */
864 async getSize() {
865     await this.#jsl.promiseOrStoped(this.ready);
866     return await this.#jsl.windows.open_windows[this.wid].getSize();
867 }
```

```
868
869  /**
870   * Retrieves the position of the window.
871   * @returns {Promise<Array>} - Resolves with an array [ left , top ].
872   */
873  async getPos() {
874    await this.#jsl.promiseOrStoped(this.ready);
875    return await this.#jsl.windows.open_windows[this.wid].getPos();
876  }
877
878  /**
879   * Retrieves the media source id of the figure.
880   * @returns {String} - Media source id.
881   */
882  async getMediaSourceId() {
883    await this.#jsl.promiseOrStoped(this.ready);
884    return await this.#jsl.windows.open_windows[this.wid].getMediaSourceId();
885  }
886
887  /**
888   * Starts video recording of the figure.
889   * @param {Object} - Optional settings.
890   * @returns {Object|boolean} - Returns recorder object.
891   */
892  async startVideoRecording(opts) {
893    await this.#jsl.promiseOrStoped(this.ready);
894    return await this.#jsl.windows.open_windows[this.wid].startVideoRecording(
895      opts);
896
897  /**
898   * Closes the window.
899   * @returns {Promise} - Resolves when the window is closed.
900   */
901  async close() {
902    await this.#jsl.promiseOrStoped(this.ready);
903    return await this.#jsl.windows.open_windows[this.wid].close();
904  }
905
906  /**
907   * Hides figure menu.
908   */
909  hideMenu() {
910    if(this.dom) {
911      this.dom.getElementById('figure-menu-button').style.display = 'none';
912      this.dom.getElementById('figure-menu-container').style.display = 'none';
913    }
914  }
915
916  /**
917   * Shows figure menu.
918   */
919  showMenu() {
920    if(this.dom) {
921      this.dom.getElementById('figure-menu-button').style.display = '';
```

```

922     this.dom.getElementById('figure-menu-container').style.display = '';
923   }
924 }
925
926 /**
927 * Creates a new plot in the figure.
928 * @param {Number} id Identifier for the new plot.
929 * @param {Array} traces Data traces for the plot.
930 * @param {Object} options Plot configuration options.
931 */
932 _newPlot(id, traces, options) {
933   if(this.plot) {
934     this.plot.remove();
935   }
936   this.plot = new PRDC_JSLAB_PLOT(this.#jsl, this.fid, id, traces, options);
937   if(this.fig_ready) {
938     this.plot._onFigureReady();
939   }
940 }
941
942 /**
943 * Creates a new plot in the figure based on JSON file data.
944 * @param {Number} id Identifier for the new plot.
945 * @param {Array} data Data for the plot.
946 */
947 async _fromJSON(id, data) {
948   if(this.plot) {
949     this.plot.remove();
950   }
951   this.plot = new PRDC_JSLAB_PLOT(this.#jsl, this.fid, id);
952   this.plot.fromJSON(data);
953   if(this.fig_ready) {
954     this.plot._onFigureReady();
955   }
956 }
957
958 /**
959 * Method called when the figure is ready. Initializes interactive elements
960 * within the figure's DOM.
961 * @param {Element} dom The DOM element associated with the figure.
962 */
963 async _onReady() {
964   var obj = this;
965   this.fig_ready = true;
966   this.win = this.#jsl.windows.open_windows[this.wid];
967   this.context = this.win.context;
968
969   this.dom = this.context.document;
970
971   this.context.addEventListener("resize", function() {
972     if(obj.#jsl.figures.open_figures.hasOwnProperty(obj.fid)) {
973       obj._onResize();
974     }
975   });

```

```

976   this.dom.title = "Figure " + this.fid + " - JSLAB | PR-DC";
977
978 // Menu showing
979 var menu_button = this.dom.getElementById('figure-menu-button');
980 var menu = this.dom.getElementById('figure-menu-container');
981
982 menu_button.addEventListener('click', function(e) {
983   e.stopPropagation();
984   menu.classList.toggle('active');
985   menu_button.classList.toggle('active');
986 });
987
988 this.dom.addEventListener('click', function(e) {
989   if(!menu.contains(e.target) && !menu_button.contains(e.target)) {
990     menu.classList.remove('active');
991     menu_button.classList.remove('active');
992   }
993 });
994
995 var interval;
996 menu.addEventListener('mouseenter', function() {
997   clearInterval(interval);
998   menu.classList.add('hovered');
999   menu_button.classList.add('hovered');
1000 });
1001
1002 menu.addEventListener('mouseleave', function() {
1003   interval = setTimeout(function() {
1004     menu.classList.remove('hovered');
1005     menu_button.classList.remove('hovered');
1006   }, 300);
1007 });
1008
1009 // Menu buttons
1010 this.dom.getElementById('save-as-menu')
1011   .addEventListener('click', function() {
1012     obj.#jsl.figures.saveFileDialog(obj.fid);
1013   });
1014 this.dom.getElementById('zoom-menu')
1015   .addEventListener('click', function() {
1016     var btn = obj.dom.querySelector('a[data-attr="dragmode"] [data-val="zoom"]');
1017     if(btn) {
1018       btn.click();
1019     }
1020   });
1021 this.dom.getElementById('zoom-in-menu')
1022   .addEventListener('click', function() {
1023     var btn = obj.dom.querySelector('a[data-attr="zoom"] [data-val="in"]');
1024     if(btn) {
1025       btn.click();
1026     }
1027   });
1028 this.dom.getElementById('zoom-out-menu')
1029   .addEventListener('click', function() {

```

```

1030   var btn = obj.dom.querySelector('a[data-attr="zoom"] [data-val="out"]');
1031   if(btn) {
1032     btn.click();
1033   }
1034 });
1035 this.dom.getElementById('pan-menu')
1036   .addEventListener('click', function() {
1037     var btn = obj.dom.querySelector('a[data-attr="dragmode"] [data-val="pan"]');
1038     if(btn) {
1039       btn.click();
1040     }
1041 });
1042 this.dom.getElementById('rotate-menu')
1043   .addEventListener('click', function() {
1044     var btn = obj.dom.querySelector('a[data-attr="dragmode"] [data-val="orbit"]');
1045     if(btn) {
1046       btn.click();
1047     }
1048 });
1049 this.dom.getElementById('fit-menu')
1050   .addEventListener('click', function() {
1051     var btn = obj.dom.querySelector('a[data-attr="zoom"] [data-val="auto"]');
1052     if(btn) {
1053       btn.click();
1054     }
1055 });
1056 this.dom.getElementById('reset-menu')
1057   .addEventListener('click', function() {
1058     obj.#jsl.ploter.updatePlotLayout(obj.plot.fid);
1059 });
1060 this.dom.getElementById('pan-menu-3d')
1061   .addEventListener('click', function() {
1062     var btn = obj.dom.querySelector('a[data-attr="scene.dragmode"] [data-val="pan"]');
1063     if(btn) {
1064       btn.click();
1065     }
1066 });
1067 this.dom.getElementById('rotate-menu-3d')
1068   .addEventListener('click', function() {
1069     var btn = obj.dom.querySelector('a[data-attr="scene.dragmode"] [data-val="orbit"]');
1070     if(btn) {
1071       btn.click();
1072     }
1073 });
1074 this.dom.getElementById('fit-menu-3d')
1075   .addEventListener('click', function() {
1076     var btn = obj.dom.querySelector('a[data-attr="resetDefault"]');
1077     if(btn) {
1078       btn.click();
1079     }
1080 });

```

```

1081     this.dom.getElementById('reset-menu-3d')
1082         .addEventListener('click', function() {
1083             obj.#jsl.ploter.updatePlotLayout(obj.plot.fid);
1084         });
1085
1086     this._readyResolve(true);
1087     if(this.plot) {
1088         await this.plot._onFigureReady();
1089     }
1090 }
1091
1092 /**
1093 * Handles figure resize events by updating the plot layout to fit the new
1094 * dimensions.
1095 */
1096 _onResize() {
1097     if(this.plot) {
1098         this.plot._onResize();
1099     }
1100 }
1101
1102 /**
1103 * Represents an individual plot within the JSLAB environment, holding
1104 * configuration details and facilitating interaction with the plot.
1105 */
1106 class PRDC_JSLAB_PLOT {
1107
1108     #jsl;
1109
1110     /**
1111      * Constructs a PRDC_JSLAB_PLOT instance with specified plot data and
1112      * configuration.
1113      * @param {Object} js1 Reference to the main JSLAB object.
1114      * @param {Number} fid Identifier for the figure containing this plot.
1115      * @param {Number} id Unique identifier for this plot.
1116      * @param {Array} traces Data traces to be displayed in the plot.
1117      * @param {Object} options Configuration options for the plot.
1118      */
1119     constructor(js1, fid, id, traces = [], options = []) {
1120         var obj = this;
1121
1122         this.#jsl = js1;
1123         this.fid = fid;
1124         this.id = id;
1125         this.traces = traces;
1126         this.options = options;
1127
1128         this.title_val;
1129         this.xlabel_val;
1130         this.ylabel_val;
1131         this.zlabel_val;
1132         this.xlim_val;
1133         this.ylim_val;
1134         this.zlim_val;

```

```
1133     this.view_val = [37.5+180, 30];
1134     this.zoom_val = 1;
1135     this.axis_style_val;
1136     this.json_val;
1137     this.legend_state_val;
1138
1139     this.plot_ready = false;
1140     this.lim_update = false;
1141
1142     this.ready = new Promise((resolve) => {
1143         obj._readyResolve = resolve;
1144     });
1145 }
1146
1147 /**
1148 * Sets the label for the x-axis of the plot.
1149 * @param {String} label Label text for the x-axis.
1150 */
1151 legend(state) {
1152     this.legend_state_val = state;
1153     if(this.plot_ready) {
1154         this.#jsl.ploter.updatePlotLayout(this.fid);
1155     }
1156 }
1157
1158 /**
1159 * Sets the label for the x-axis of the plot.
1160 * @param {String} label Label text for the x-axis.
1161 */
1162 xlabel(label) {
1163     this.xlabel_val = label;
1164     if(this.plot_ready) {
1165         this.#jsl.ploter.updatePlotLayout(this.fid);
1166     }
1167 }
1168
1169 /**
1170 * Sets the label for the y-axis of the plot.
1171 * @param {String} label Label text for the y-axis.
1172 */
1173 ylabel(label) {
1174     this.ylabel_val = label;
1175     if(this.plot_ready) {
1176         this.#jsl.ploter.updatePlotLayout(this.fid);
1177     }
1178 }
1179
1180 /**
1181 * Sets the label for the z-axis of the plot.
1182 * @param {String} label Label text for the z-axis.
1183 */
1184 zlabel(label) {
1185     this.zlabel_val = label;
1186     if(this.plot_ready) {
1187         this.#jsl.ploter.updatePlotLayout(this.fid);
```

```
1188     }
1189 }
1190
1191 /**
1192 * Sets the title of the plot.
1193 * @param {String} label Title text for the plot.
1194 */
1195 title(label) {
1196     this.title_val = label;
1197     if(this.plot_ready) {
1198         this.#jsl.ploter.updatePlotLayout(this.fid);
1199     }
1200 }
1201
1202 /**
1203 * Sets the limits for the x-axis of the plot.
1204 * @param {String} lim Limits for the x-axis.
1205 */
1206 xlim(lim) {
1207     this.lim_update = true;
1208     this.xlim_val = lim;
1209     if(this.plot_ready) {
1210         this.#jsl.ploter.updatePlotLayout(this.fid);
1211     }
1212 }
1213
1214 /**
1215 * Sets the limits for the y-axis of the plot.
1216 * @param {String} lim Limits for the y-axis.
1217 */
1218 ylim(lim) {
1219     this.lim_update = true;
1220     this.ylim_val = lim;
1221     if(this.plot_ready) {
1222         this.#jsl.ploter.updatePlotLayout(this.fid);
1223     }
1224 }
1225
1226 /**
1227 * Sets the limits for the z-axis of the plot.
1228 * @param {String} lim Limits for the z-axis.
1229 */
1230 zlim(lim) {
1231     this.lim_update = true;
1232     this.zlim_val = lim;
1233     if(this.plot_ready) {
1234         this.#jsl.ploter.updatePlotLayout(this.fid);
1235     }
1236 }
1237
1238 /**
1239 * Adjusts the view based on azimuth and elevation angles.
1240 * @param {number} azimuth - The azimuth angle.
1241 * @param {number} elevation - The elevation angle.
1242 */

```

```
1243     view(azimuth, elevation) {
1244         this.view_val = [azimuth, elevation];
1245         if(this.plot_ready) {
1246             this.#jsl.ploter.updatePlotLayout(this.fid);
1247         }
1248     }
1249
1250     /**
1251      * Adjusts the zoom based on factor.
1252      * @param {number} factor - The zoom factor.
1253      */
1254     zoom(factor) {
1255         this.zoom_val = factor;
1256         if(this.plot_ready) {
1257             this.#jsl.ploter.updatePlotLayout(this.fid);
1258         }
1259     }
1260
1261     /**
1262      * Sets the axis style value and updates the plot layout if the plot is
1263      * ready.
1264      * @param {Object} style - The style configuration to set for the axis.
1265      */
1266     axis(style) {
1267         this.axis_style_val = style;
1268         if(this.plot_ready) {
1269             this.#jsl.ploter.updatePlotLayout(this.fid);
1270         }
1271     }
1272
1273     /**
1274      * Sets the plot data from JSON.
1275      * @param {Array} data Data for the plot.
1276      */
1277     fromJSON(data) {
1278         this.json_val = data;
1279         if(this.plot_ready) {
1280             this.#jsl.ploter.fromJSON(data);
1281         }
1282     }
1283
1284     /**
1285      * Adds a print job to the queue and prints it if the system is ready.
1286      * @param {String} filename - The filename for the print job.
1287      * @param {Object} options - Options for the print job.
1288      */
1289     async print(filename, options) {
1290         await this.#jsl.promiseOrStoped(this.ready);
1291         var type = 'png';
1292         var size;
1293         if(options) {
1294             if(options.type) {
1295                 type = options.type;
1296             }
1297             if(options.size) {
```

```

1297         size = options.size;
1298     }
1299   }
1300   await this.#jsl.figures.saveFigure(this.fid, filename+'.'+type, size);
1301 }
1302
1303 /**
1304 * Updates plot data by delegating to the 'updateData' method.
1305 * @param {Object} traces - The trace data to be updated in the plot.
1306 * @param {number} N - The data length or index for updating the plot.
1307 */
1308 update(traces, N) {
1309   this.#jsl.ploter.updateData(this.fid, traces, N);
1310 }
1311
1312 /**
1313 * Updates plot data by delegating to the 'updateDataById' method.
1314 * @param {Object|Object[]} data - Trace update object(s) addressed by 'id'.
1315 */
1316 updateById(data) {
1317   this.#jsl.ploter.updateDataById(this.fid, data);
1318 }
1319
1320 /**
1321 * Removes the plot from the figure, cleaning up any resources associated
1322 * with it.
1323 */
1324 remove() {
1325   if(this.plot_ready) {
1326     this.#jsl.ploter.remove(this.fid);
1327   }
1328 }
1329
1330 /**
1331 * Called when the figure containing this plot is ready, allowing for final
1332 * adjustments or updates before displaying.
1333 */
1334 async _onFigureReady() {
1335   if(this.json_val) {
1336     await this.#jsl.ploter.fromJSON(this.fid, this.json_val);
1337     this.plot_ready = true;
1338     this._readyResolve(true);
1339   } else {
1340     await this.#jsl.ploter.plot(this.fid);
1341     this.plot_ready = true;
1342     this.#jsl.ploter.updatePlotLayout(this.fid);
1343     this._readyResolve(true);
1344   }
1345 }
1346
1347 /**
1348 * Handles plot resize events, updating the plot layout to accommodate new
1349 * dimensions.
1350 */
1351 _onResize() {

```

```

1349     this.#jsl.ploter.onResize(this.fid);
1350   }
1351 }
```

Listing 91 - figures.js

```

1  /**
2   * @file JSLAB library file system submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB file system submodule.
10  */
11 class PRDC_JSLAB_LIB_FILE_SYSTEM {
12
13  /**
14   * Initializes a new instance of the file system submodule.
15   * @param {Object} js1 Reference to the main JSLAB object.
16   */
17 constructor(js1) {
18   var obj = this;
19   this.js1 = js1;
20 }
21
22 /**
23  * Reads the content of a file at the specified path.
24  * @param {string} file_path Path to the file.
25  * @returns {((Buffer|string|false)} The content of the file or false in case
26  *       of an error.
27  */
28 readFile(...args) {
29   return this.js1.env.readFileSync(...args);
30 }
31
32 /**
33  * Extract substring from file using range
34  * @param {string} filepath - Path to file
35  * @param {Array} range - Character range [start, end]
36  * @returns {string} - Extracted substring
37  */
38 getContentFromCharRange(filepath, range) {
39   const fileContent = this.readFile(filepath, "utf8");
40   const [start, end] = range;
41   return fileContent.slice(start, end);
42 }
43
44 /**
45  * Writes data to a specified file synchronously. This method should
46  * overwrite the file if it already exists.
47  * @param {string} file_path The path to the file where data will be written
48  *
49  * @param {Buffer|string} data The data to write to the file.
50  * @returns {boolean} Returns true if the file was written successfully,
51  */
52 writeFile(...args) {
53   const [filePath, data] = args;
54   const fileContent = this.readFile(filePath, "utf8");
55   const [start, end] = range;
56   const fileContentWithWrittenData =
57     fileContent.slice(0, start) + data + fileContent.slice(end);
58   this.writeFile(fileContentWithWrittenData, filePath);
59   return true;
60 }
```

```
        false if an error occurred.  
48     */  
49     writeFile(... args) {  
50         return this.jsl.env.writeFileSync(... args);  
51     }  
52  
53     /**  
54      * Deletes a specified file synchronously.  
55      * @param {string} file_path The path to the file that should be deleted.  
56      * @returns {boolean} Returns true if the file was deleted successfully,  
57      *               false if an error occurred.  
58      */  
59     deleteFile(file_path) {  
60         return this.jsl.env.rmSync(file_path);  
61     }  
62  
63     /**  
64      * Reads the contents of a directory synchronously.  
65      * @param {string} folder The path to the directory.  
66      * @returns {string[]} An array of filenames or false in case of an  
67      *                 error.  
68      */  
69     readDir(... args) {  
70         return this.jsl.env.readDir(... args);  
71     }  
72  
73     /**  
74      * Deletes a specified file synchronously.  
75      * @param {string} file_path The path to the file that should be deleted.  
76      * @returns {boolean} Returns true if the file was deleted successfully,  
77      *               false if an error occurred.  
78      */  
79     deleteDir(file_path) {  
80         return this.jsl.env.rmSync(file_path);  
81     }  
82  
83     /**  
84      * Moves a file from source to destination.  
85      * @param {string} source - The path to the source file.  
86      * @param {string} destination - The path to the destination file.  
87      */  
88     moveFile(source, destination) {  
89         if(comparePaths(source, destination)) {  
90             return true;  
91         }  
92         try {  
93             this.jsl.env.copyFileSync(source, destination);  
94             this.jsl.env.rmSync(source);  
95             return true;  
96         } catch(err) {  
97             this.jsl.error('@moveFile: ' + err);  
98         }  
99     }  
100  
101    /**
```

```

99   * Copies a file from source to destination .
100  * @param {string} source - The path to the source file .
101  * @param {string} destination - The path to the destination file .
102  */
103 copyFile(source , destination) {
104   if(comparePaths(source , destination)) {
105     return true ;
106   }
107   try {
108     this . jsl . env . copyFileSync(source , destination);
109     return true ;
110   } catch(err) {
111     this . jsl . error('@copyFile: ' + err);
112   }
113 }
114 /**
115  * Lists files in a specified folder , optionally filtering by extension .
116  * @param {string} folder Path to the folder .
117  * @param {string} ext File extension filter .
118  * @returns {string[]|void} Array of file paths matching the extension in
119  * the specified folder .
120  */
121 filesInFolder(folder , ext) {
122   var obj = this ;
123   var files = this . jsl . env . readDir(folder);
124   if(Array.isArray(files)) {
125     return files
126       .filter(function(file) {
127         if (!ext) return file.includes('.');
128         return file.endsWith('.' + ext);
129       })
130       .map(function(file) { return obj . jsl . env . pathJoin(folder , file); });
131   } else {
132     this . jsl . env . error('@filesInFolder: '+language . string(128)+': ' + folder
133       );
134   }
135   return false ;
136 }
137 /**
138  * Lists all files in a specified folder
139  * @param {string} folder Path to the folder .
140  * @returns {string[]|void} Array of file names .
141  */
142 allFilesInFolder(folder) {
143   return this . jsl . env . readDir(folder) . reduce((acc , file) => {
144     const file_path = this . jsl . env . pathJoin(folder , file);
145     return this . jsl . env . checkDirectory(file_path)
146       ? acc . concat(this . allFilesInFolder(file_path))
147       : acc . concat(file);
148   }, []);
149 }
150 /**

```

```
152 * Opens a dialog for the user to choose a file , synchronously .
153 * @param {Object} options Configuration options for the dialog .
154 * @returns {string|string[]} The selected file path(s) or an empty array if
155 * canceled .
156 */
157 chooseFile(options) {
158     var file_path = this.jsl.env.showOpenDialogSync(options);
159     if(file_path === undefined) {
160         this.jsl.env.error('@chooseFile: '+language.string(126));
161         return [];
162     }
163     return file_path;
164 }
165 /**
166 * Opens a dialog for the user to choose a folder , synchronously .
167 * @param {Object} options Configuration options for the dialog .
168 * @returns {string|string[]} The selected folder path(s) or an empty array
169 * if canceled .
170 */
171 chooseFolder(options_in) {
172     var options = {
173         properties: [ 'openDirectory' ],
174         ...options_in
175     };
176     var file_path = this.jsl.env.showOpenDialogSync(options);
177     if(file_path === undefined) {
178         this.jsl.env.error('@chooseFolder: '+language.string(126));
179         return [];
180     }
181     return file_path;
182 }
183 /**
184 * Retrieves a default path based on a specified type .
185 * @param {string} type Type of the default path (e.g., 'root' , 'documents' )
186 *
187 * @returns {string} The default path for the specified type .
188 */
189 getDefaultPath(type) {
190     return this.jsl.env.getDefaultPath(type);
191 }
192 /**
193 * Opens the specified folder in the system's file manager .
194 * @param {string} filepath Path to the folder .
195 */
196 openFolder(filepath) {
197     this.jsl.env.openFolder(filepath);
198 }
199 /**
200 * Creates a directory at the specified path if it does not already exist .
201 * This method delegates the directory creation task to the environment's
makeDirectory function .
```

```
203     * @param {string} directory - The path where the directory will be created.  
204     * @returns {boolean} True if the directory was successfully created or  
205       already exists, false if an error occurred.  
206     */  
207   makeDirectory(directory) {  
208     return this.jsl.env.makeDirectory(directory);  
209   }  
210  
211   /**
212    * Alias for makeDirectory. Creates a directory at the specified path if it  
213      does not already exist.  
214    * This method delegates the directory creation task to the environment's  
215      makeDirectory function.  
216    * @param {string} directory - The path where the directory will be created.  
217    * @returns {boolean} True if the directory was successfully created or  
218      already exists, false if an error occurred.  
219    */  
220   mkdir(directory) {  
221     return this.jsl.env.makeDirectory(directory);  
222   }  
223  
224   /**
225    * Opens the specified directory in the system's file manager. Alias for 'openFolder'.  
226    * @param {string} filepath Path to the directory.  
227    */  
228   openDir(filepath) {  
229     this.jsl.env.openDir(filepath);  
230   }  
231  
232   /**
233    * Shows the specified folder in the system's file manager. Alias for 'openFolder'.  
234    * @param {string} filepath Path to the folder.  
235    */  
236   showFolder(filepath) {  
237     this.jsl.env.openFolder(filepath);  
238   }  
239  
240   /**
241    * Shows the specified directory in the system's file manager. Alias for 'openDir'.  
242    * @param {string} filepath Path to the directory.  
243    */  
244   showDir(filepath) {  
245     this.jsl.env.openDir(filepath);  
246   }  
247  
248   /**
249    * Opens the program's root folder in the system's file manager.  
250    */  
251   openProgramFolder() {  
252     this.jsl.env.openFolder(this.jsl.env.getDefaultPath('root'));  
253   }
```

```

251  /**
252   * Shows the specified file in its containing folder within the system's
253   * file manager.
254   * @param {string} filepath Path to the file.
255   */
256   showFileInFolder(filepath) {
257     this.jsl.env.showFileInFolder(filepath);
258   }
259
260 /**
261  * Shows the specified file in its containing directory within the system's
262  * file manager. Alias for 'showFileInFolder'.
263  * @param {string} filepath Path to the file.
264  */
265   showFileInDir(filepath) {
266     this.jsl.env.showFileInDir(filepath);
267   }
268
269 /**
270  * Reads a CSV file and returns a promise that resolves with the parsed data
271  *
272  * @param {string} filePath - Path to the CSV file.
273  * @param {string} delimiter - Delimiter used in the CSV file (e.g., ',',',
274  *                           ';', '\t').
275  * @returns {Array<Object>} - Parsed CSV data as an array of objects.
276  */
277   readcsv(filePath, delimiter = ',', hasHeader = false) {
278     var data = this.jsl.env.readFileSync(filePath, 'utf-8');
279     var lines = data.split('\n').filter(function(line) { return line.trim() !== '' });
280     var headers = lines[0].split(delimiter).map(function(header) { return header.trim() });
281     var result = [];
282
283     if(hasHeader) {
284       // If there is a header, parse as objects
285       var row_object = {};
286       headers.forEach(function(header, index) {
287         row_object[header] = lines[1][index];
288       });
289       result.push(row_object);
290     } else {
291       // If no header, parse as arrays
292       for(let i = 0; i < lines.length; i++) {
293         var row = lines[i].split(delimiter).map(function(cell) { return cell.trim() });
294         result.push(row);
295       }
296     }
297   }

```

```

298     return result ;
299 }
300
301 /**
302 * Checks if the specified file exists.
303 * @param {string} file - The path to the file to check.
304 */
305 checkFile(file) {
306     return this.jsl.env.checkFile(file);
307 }
308
309 /**
310 * Checks if the specified file exists.
311 * @param {string} file - The path to the file to check.
312 */
313 existFile(file) {
314     return this.checkFile(file);
315 }
316
317 /**
318 * Checks if the specified directory exists.
319 * @param {string} directory - The path to the directory to check.
320 */
321 checkDirectory(directory) {
322     return this.jsl.env.checkDirectory(directory);
323 }
324
325 /**
326 * Checks if the specified directory exists.
327 * @param {string} directory - The path to the directory to check.
328 */
329 existDirectory(directory) {
330     return this.checkDirectory(directory);
331 }
332
333 /**
334 * Recursively copies a directory from the source path to the destination
335 * path.
336 * @param {string} src - The source directory path.
337 * @param {string} dest - The destination directory path.
338 */
339 copyDir(src, dest) {
340     // Check if the source directory exists
341     if(!this.jsl.env.checkDirectory(src)) {
342         this.jsl.env.error('@copyDir: '+language.string(173));
343     }
344     // Create the destination directory if it doesn't exist
345     this.jsl.env.makeDirectory(dest);
346
347     // Read all the files and directories in the source directory
348     const entries = this.jsl.env.readDir(src, { withFileTypes: true });
349
350     // Iterate through each entry (file or directory)
351     for(const entry of entries) {

```

```

352     const src_path = this.jsl.env.pathJoin(src, entry.name);
353     const dest_path = this.jsl.env.pathJoin(dest, entry.name);
354
355     if(entry.isDirectory()) {
356         // Recursively copy directories
357         this.copyDir(src_path, dest_path);
358     } else {
359         // Copy files
360         this.jsl.env.copyFileSync(src_path, dest_path);
361     }
362 }
363
364 /**
365 * Copies a folder from the source path to the destination path.
366 * @param {string} src - The source folder path.
367 * @param {string} dest - The destination folder path.
368 */
369 copyFolder(src, dest) {
370     return this.copyDir(src, dest);
371 }
372
373 /**
374 * Copies a directory from the source path to the destination path.
375 * @param {string} src - The source directory path.
376 * @param {string} dest - The destination directory path.
377 */
378 cp(src, dest) {
379     return this.copyDir(src, dest);
380 }
381
382 /**
383 * Copies a 7z archive from the source path to the destination path,
384 * extracts it, and removes the archive.
385 * @param {string} src - The source 7z archive path.
386 * @param {string} dest - The destination directory path.
387 */
388 copyDir7z(src, dest) {
389     var name = this.jsl.path.pathFileName(src);
390     var ext = this.jsl.path.pathExtName(src);
391     var filePath = this.jsl.env.pathJoin(dest, name + ext);
392
393     this.jsl.env.copyFileSync(src, filePath);
394     this.jsl.env.execSync(`"${this.jsl.env.bin7zip}" x "${filePath}" -o"${dest
395         }" -y`);
396     this.jsl.env.rmSync(filePath);
397 }
398
399 exports.PRDC_JSLAB_LIB_FILE_SYSTEM = PRDC_JSLAB_LIB_FILE_SYSTEM;

```

Listing 92 - file-system.js

```

1 /**
2 * @file JSLAB library format submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>

```

```
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9  * Class for JSLAB format submodule.
10 */
11 class PRDC_JSLAB_LIB_FORMAT {
12
13 /**
14  * Initializes a new instance of the format submodule.
15  * @param {Object} js1 Reference to the main JSLAB object.
16  */
17 constructor(js1) {
18     var obj = this;
19     this.js1 = js1;
20 }
21
22 /**
23  * Retrieves the MIME type based on the file extension.
24  * @param {string} filePath - The path to the file.
25  * @returns {string} The corresponding MIME type.
26  */
27 getContentType(filePath) {
28     const mime_types = {
29         // Text files
30         '.html': 'text/html',
31         '.htm': 'text/html',
32         '.js': 'text/javascript',
33         '.mjs': 'text/javascript',
34         '.css': 'text/css',
35         '.json': 'application/json',
36         '.txt': 'text/plain',
37         '.xml': 'application/xml',
38
39         // Image files
40         '.png': 'image/png',
41         '.jpg': 'image/jpeg',
42         '.jpeg': 'image/jpeg',
43         '.gif': 'image/gif',
44         '.bmp': 'image/bmp',
45         '.webp': 'image/webp',
46         '.svg': 'image/svg+xml',
47         '.ico': 'image/x-icon',
48
49         // Audio files
50         '.mp3': 'audio/mpeg',
51         '.wav': 'audio/wav',
52         '.ogg': 'audio/ogg',
53         '.m4a': 'audio/mp4',
54
55         // Video files
56         '.mp4': 'video/mp4',
57         '.avi': 'video/x-msvideo',
58         '.mov': 'video/quicktime',
```

```

59   '.wmv': 'video/x-ms-wmv',
60   '.flv': 'video/x-flv',
61   '.webm': 'video/webm',
62   '.mkv': 'video/x-matroska',
63
64   // Application files
65   '.pdf': 'application/pdf',
66   '.zip': 'application/zip',
67   '.rar': 'application/vnd.rar',
68   '.7z': 'application/x-7z-compressed',
69   '.tar': 'application/x-tar',
70   '.gz': 'application/gzip',
71   '.exe': 'application/vnd.microsoft.portable-executable',
72   '.msi': 'application/x-msdownload',
73   '.doc': 'application/msword',
74   '.docx': 'application/vnd.openxmlformats-officedocument.wordprocessingml
              .document',
75   '.xls': 'application/vnd.ms-excel',
76   '.xlsx': 'application/vnd.openxmlformats-officedocument.spreadsheetml
              .sheet',
77   '.ppt': 'application/vnd.ms-powerpoint',
78   '.pptx': 'application/vnd.openxmlformats-officedocument.presentationml
              .presentation',
79   '.eot': 'application/vnd.ms-fontobject',
80   '.ttf': 'font/ttf',
81   '.woff': 'font/woff',
82   '.woff2': 'font/woff2',
83
84   // Model files
85   '.glb': 'model/gltf-binary',
86   '.gltf': 'model/gltf+json',
87   '.obj': 'application/octet-stream', // Common for OBJ, but can vary
88   '.fbx': 'application/octet-stream',
89
90   // Others
91   '.csv': 'text/csv',
92   '.md': 'text/markdown',
93   '.apk': 'application/vnd.android.package-archive',
94   '.iso': 'application/x-iso9660-image',
95   '.sh': 'application/x-sh',
96   '.bat': 'application/x-msdownload',
97   '.php': 'application/x-httpd-php',
98   '.asp': 'application/x-aspx',
99   '.aspx': 'application/x-aspx',
100  '.jsp': 'application/java-archive',
101  '.rb': 'application/x-ruby',
102  '.py': 'application/x-python-code',
103  '.swift': 'application/x-swift',
104  '.lua': 'application/x-lua',
105 };
106
107 const ext = String(this.jsl.env.pathExtName(file_path)).toLowerCase();
108 return mime_types[ext] || 'application/octet-stream';
109 }
110

```

```

111  /**
112   * Formats the given byte count into a readable string.
113   * @param {Number} bytes Number of bytes.
114   * @param {Number} [decimals=2] Number of decimal places to include in the
115   *     formatted string.
116   * @returns {String} Formatted bytes string with appropriate unit.
117   */
118  formatBytes(bytes, decimals = 2) {
119    if(!+bytes) return '0 Bytes';
120
121    const k = 1024;
122    const dm = decimals < 0 ? 0 : decimals;
123    const units = [ 'Bytes', 'KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB' ];
124
125    const i = Math.floor(Math.log(bytes) / Math.log(k));
126
127    return `${parseFloat((bytes / Math.pow(k, i)).toFixed(dm))} ${units[i]}`;
128
129  /**
130   * Formats the given bits per second (bps) into a readable string.
131   * @param {Number} bps Number of bits per second.
132   * @param {Number} [decimals=2] Number of decimal places to include in the
133   *     formatted string.
134   * @returns {String} Formatted bits per second string with appropriate unit.
135   */
136  formatBPS(bps, decimals = 2) {
137    if(!+bps) return '0 bps';
138
139    const k = 1000;
140    const dm = decimals < 0 ? 0 : decimals;
141    const units = [ "bps", "kbps", "Mbps", "Gbps", "Tbps", "Pbps", "Ebps", "Zbps",
142      "Ybps" ];
143
144    const i = Math.floor(Math.log(bps) / Math.log(k));
145
146    return `${parseFloat((bps / Math.pow(k, i)).toFixed(dm))} ${units[i]}`;
147
148  /**
149   * Formats a number with metric prefixes (k, M, G, etc.) based on its value.
150   * @param {Number} number The number to format.
151   * @param {Number} [decimals=2] Number of decimal places to include in the
152   *     formatted string.
153   * @returns {String} Formatted number string with metric prefix.
154   */
155  formatPrefix(number, decimals = 2) {
156    if(!+number) return '0';
157
158    const k = 1000;
159    const dm = decimals < 0 ? 0 : decimals;
160    const units = [ "", "k", "M", "G", "T", "P", "E", "Z", "Y" ];
161
162    const i = Math.floor(Math.log(number) / Math.log(k));

```

```

161
162     return `${parseFloat((number / Math.pow(k, i)).toFixed(dm))} ${units[i]}`;
163 }
164 /**
165 * Formats a number to a specified number of decimal places.
166 * @param {Number} number The number to format.
167 * @param {Number} [decimals=2] The number of decimal places.
168 * @returns {String} The formatted number as a string.
169 */
170 formatNum(number, decimals = 2) {
171   return Number(number).toFixed(decimals);
172 }
173 /**
174 * Formats a floating-point number to have a fixed number of significant
175 * digits.
176 * @param {number} number - The number to format.
177 * @param {number} [digits=1] - The number of significant digits.
178 * @returns {string} The formatted number as a string.
179 */
180 formatFloatDigits(number, digits = 1) {
181   var precision = digits - (number.toString().split('.')[0]).length;
182   if(precision < 0) {
183     precision = 0;
184   }
185   return round(number, precision, true);
186 }
187 /**
188 * Clips a number to a specified value based on a condition.
189 * @param {number} number - The number to clip.
190 * @param {number} control_number - The reference number for the clipping
191 * condition.
192 * @param {number} clip_value - The value to clip to.
193 * @param {boolean} [direction=true] - The direction of clipping (true for
194 * max, false for min).
195 * @returns {number} The clipped number.
196 */
197 clip(number, control_number, clip_value, direction = true) {
198   if((direction && number >= control_number) || (!direction && number <=
199     control_number)) {
200     number = clip_value;
201   }
202   return number;
203 }
204 /**
205 * Clips a height value to a specified range.
206 * @param {number} value - The value to clip.
207 * @param {number} [min=0] - The minimum value of the range.
208 * @param {number} [max=100] - The maximum value of the range.
209 * @returns {number} The clipped height value.
210 */
211 clipHeight(value, min = 0, max = 100) {
212   value = Number(value);

```

```

212     if(isNaN(value)) {
213         value = min;
214     } else if(value > max) {
215         value = max;
216     } else if(value < min) {
217         value = min;
218     }
219     return value;
220 }
221
222 /**
223 * Retrieves the field names (keys) of the given object.
224 * @param {Object} obj - The object from which to extract keys.
225 * @returns {string[]} An array containing the keys of the object.
226 */
227 fieldnames(obj) {
228     return Object.keys(obj);
229 }
230
231 /**
232 * Replaces all occurrences of a specified substring within a string.
233 * @param {string} str - The original string.
234 * @param {string} old_string - The substring to be replaced.
235 * @param {string} new_string - The substring to replace with.
236 * @returns {string} The resulting string after replacements.
237 */
238 strrep(str, old_string, new_string) {
239     return str.replaceAll(old_string, new_string);
240 }
241
242 /**
243 * Clips a value to a specified range.
244 * @param {number} value - The value to clip.
245 * @param {number} [min=0] - The minimum value of the range.
246 * @param {number} [max=100] - The maximum value of the range.
247 * @returns {number} The clipped value.
248 */
249 clipVal(value, min = 0, max = 100) {
250     value = Number(value);
251     if(isNaN(value)) {
252         value = min;
253     } else if(value > max) {
254         value = max;
255     } else if(value < min) {
256         value = min;
257     }
258     return value;
259 }
260
261 /**
262 * Rounds a number to a fixed number of decimal places, but only if it's a
263 * floating point number.
264 * @param {Number} value The value to round.
265 * @param {Number} p The number of decimal places to round to.
266 * @returns {Number} The rounded number, or the original number if it wasn't

```

```

      a float.
266  */
267 toFixedIf(value, p) {
268   return +parseFloat(value).toFixed(p);
269 }
270
271 /**
272  * Provides a replacer function for JSON.stringify() to prevent circular
273  * references.
274  * @returns {Function} A replacer function that can be used with JSON.
275  *         stringify to handle circular references.
276  */
277 getCircularReplacer() {
278   const seen = new WeakSet();
279   return function(key, value) {
280     if(typeof value === 'object' && value !== null) {
281       if(seen.has(value)) {
282         return;
283       }
284       seen.add(value);
285     }
286     return value;
287   };
288 }
289 /**
290  * Determines if the provided value is NaN.
291  * @param {*} value - The value to check.
292  * @returns {boolean} True if the value is NaN, false otherwise.
293  */
294 isNaN(value) {
295   if.isArray(value)) {
296     return mathjs._isNaN(value);
297   } else {
298     return this.jsl._isNaN(value);
299   }
300 }
301 /**
302  * Determines if the provided value is an object.
303  * @param {*} value - The value to check.
304  * @param {boolean} [ignore_array=false] - Whether to consider arrays as not
305  *         objects.
306  * @returns {boolean} True if the value is an object, false otherwise.
307  */
308 isObject(value, ignore_array) {
309   if(ignore_array && Array.isArray(value)) {
310     return false;
311   }
312   return typeof value === 'object' && value !== null;
313 }
314 /**
315  * Determines if the provided value is a number.
316  * @param {*} value - The value to check.

```

```
317     * @returns {boolean} True if the value is a number, false otherwise.
318     */
319     isNumber(value) {
320         return typeof value === 'number';
321     }
322
323     /**
324      * Determines if the provided value is a string.
325      * @param {*} value - The value to check.
326      * @returns {boolean} True if the value is a string, false otherwise.
327      */
328     isString(value) {
329         return typeof value === 'string';
330     }
331
332     /**
333      * Checks if a string is empty or contains only whitespace.
334      * @param {string} str - The string to check.
335      * @returns {boolean} - True if the string is empty or contains only
336      * whitespace, otherwise false.
337      */
338     isEmptyString(str) {
339         return typeof str === 'string' && str.trim().length === 0;
340     }
341
342     /**
343      * Determines if the provided value is a function.
344      * @param {*} value - The value to check.
345      * @returns {boolean} True if the value is a function, false otherwise.
346      */
347    isFunction(value) {
348         return typeof value === 'function';
349     }
350
351     /**
352      * Determines if the provided value is an array.
353      * @param {*} value - The value to check.
354      * @returns {boolean} True if the value is an array, false otherwise.
355      */
356     isArray(value) {
357         return Object.prototype.toString.call(value) === '[object Array]';
358     }
359
360     /**
361      * Determines if the provided value is null.
362      * @param {*} value - The value to check.
363      * @returns {boolean} True if the value is null, false otherwise.
364      */
365    isNull(value) {
366         return value === null;
367     }
368
369     /**
370      * Checks if an array is empty.
371      * @param {Array} array - The array to check.
```

```

371   * @returns {boolean} True if the array is empty, false otherwise.
372   */
373 isEmpty(array) {
374   return !(typeof array === 'object' && array.length != 0);
375 }
376
377 /**
378  * Checks if the given value(s) are infinite.
379  * @param {number|number[]} value - The value or array of values to check.
380  * @returns {boolean|boolean[]} - Returns true if infinite, otherwise false.
381  *           Returns an array of booleans if input is an array.
382  */
383 isInfinity(value) {
384   if(Array.isArray(value)) {
385     return value.map((v) => v === Infinity || v === -Infinity);
386   } else {
387     return value === Infinity || value === -Infinity;
388   }
389 }
390 /**
391  * Checks if a variable is numeric.
392  * @param {*} variable - The variable to check.
393  * @returns {boolean} True if the variable is numeric, false otherwise.
394  */
395 isNumeric(variable) {
396   return !isNaN(parseFloat(variable)) && isFinite(variable);
397 }
398
399 /**
400  * Checks if the specified object has the given key.
401  * @param {Object} object - The object to check for the presence of the key.
402  * @param {string} key - The key to check for in the object.
403  * @returns {boolean} - True if the object has the key, false otherwise.
404  */
405 hasKey(object, key) {
406   return object.hasOwnProperty(key);
407 }
408
409 /**
410  * Determines if the provided value is undefined.
411  * @param {*} value - The value to check.
412  * @returns {boolean} True if the value is undefined, false otherwise.
413  */
414 isUndefined(value) {
415   return typeof value === 'undefined';
416 }
417
418 /**
419  * Checks if a string is a valid UUID.
420  * @param {string} str The string to check.
421  * @returns {boolean} True if the string is a valid UUID, false otherwise.
422  */
423 isUUID(str) {
424   var uuid_pattern = /^[0-9a-f]{8}-[0-9a-f]{4}-[0-5][0-9a-f]{3}-[089ab][0-9a

```

```

425     -f]{3}-[0-9a-f]{12}\$/i;
426     return uuid_pattern.test(str);
427 }
428 /**
429 * Normalizes an angle to the range of -180 to 180 degrees.
430 * @param {number} angle - The angle to normalize.
431 * @returns {number} The normalized angle.
432 */
433 normalizeAngle(angle) {
434   return ((angle + 180) % 360 + 360) % 360 - 180;
435 }
436 /**
437 * Normalizes an angle to the range of 0 to 360 degrees.
438 * @param {number} angle - The angle to normalize.
439 * @returns {number} The normalized angle.
440 */
441 normalizeAngle360(angle) {
442   return (angle % 360 + 360) % 360;
443 }
444 /**
445 * Normalizes latitude to the range of -90 to 90 degrees with specified
446 * precision.
447 * @param {number} latitude - The latitude to normalize.
448 * @param {number} precision - The precision of the normalization.
449 * @returns {number} The normalized latitude.
450 */
451 normalizeLat(latitude, precision) {
452   return round(Math.max(-90, Math.min(90, latitude)), precision);
453 }
454 /**
455 * Normalizes longitude to the range of -180 to 180 degrees with specified
456 * precision.
457 * @param {number} longitude - The longitude to normalize.
458 * @param {number} precision - The precision of the normalization.
459 * @returns {number} The normalized longitude.
460 */
461 normalizeLon(longitude, precision) {
462   return round(normalizeAngle(longitude), precision);
463 }
464 /**
465 * Checks a value for undefined and returns an empty string if it is
466 * undefined, otherwise returns the value.
467 * @param {*} val - The value to check.
468 * @returns {*} The original value if not undefined, otherwise an empty
469 * string.
470 */
471 checkUndefined(val) {
472   if(val === undefined) {
473     return '';
474   } else {

```

```

475         return val;
476     }
477 }
478 /**
479 * Pretty-prints data, converting it into a more readable format for display
480 * . Handles strings, objects, and other data types.
481 * @param {*} data The data to pretty-print.
482 * @returns {Array} An array containing the pretty-printed data as a string
483 * and a boolean indicating if the data was an object.
484 */
485 prettyPrint(data) {
486     if(Array.isArray(data)) {
487         return [this.safeStringify(data, 5), true];
488     } else if(typeof data === 'object' && typeof data.toPrettyString ===
489     'function') {
490         return [data.toPrettyString(), false];
491     } else if(typeof data === 'object' && typeof data.toSafeJSON === 'function')
492     {
493         return [this.safeStringify(data), true];
494     } else if(typeof data === 'string') {
495         return ["'" + data.replace(/\n/g, '<br/>') + "'", false];
496     } else if(typeof data === 'object') {
497         return [this.safeStringify(data), true];
498     } else {
499         return [String(data), false];
500     }
501 }
502 /**
503 * Converts an object to a JSON string if it is an object, otherwise returns
504 * the object as is.
505 * @param {*} object - The object to stringify.
506 * @returns {string|*} The JSON string representation of the object or the
507 * object itself if not an object.
508 */
509 stringify(object) {
510     if(typeof object.toPrettyString === 'function') {
511         return object.toPrettyString();
512     }
513     return JSON.stringify(object);
514 }
515 /**
516 * Safely serializes an object into a JSON string, handling circular
517 * references and deep structures, with depth control.
518 * It also escapes strings to prevent HTML injection.
519 * @param {Object} data - The object to stringify.
520 * @param {number} [depth_limit=3] - The maximum depth to traverse in the
521 * object, beyond which the traversal is stopped.
522 * @returns {string} A JSON string representation of the object, with
523 * special handling for deep objects, circular references, and HTML
524 * escaping of strings.
525 */

```

```

520  safeStringify(data, depth_limit = 3) {
521    if(data == null || typeof data != 'object') {
522      return data;
523    }
524    if(typeof data.toPrettyString === 'function') {
525      return data.toPrettyString();
526    }
527    if(typeof data.toSafeJSON === 'function') {
528      return data.toSafeJSON();
529    }
530    if(data.hasOwnProperty('_safeStringifyDepth')) {
531      depth_limit = data._safeStringifyDepth;
532      delete data._safeStringifyDepth;
533    }
534
535    const seen = new WeakSet();
536
537    function escapeString(str) {
538      return str.replace(/[&<>]=\//g, function (s) {
539        return ({
540          '&': '&amp;',
541          '<': '&lt;',
542          '>': '&gt;',
543          '"': '&quot;',
544          "'": '&#39;',
545          '&nbsp;': '&#x60;',
546          '=': '&#x3D;',
547          '/': '&#xF;',
548        })[s];
549      });
550    }
551
552    function helper(value, depth, path) {
553      if(depth > depth_limit) {
554        return '{...}';
555      }
556
557      if(value !== null && typeof value === 'object') {
558        if(seen.has(value)) {
559          return '[Circular Reference]';
560        }
561        seen.add(value);
562
563        // Handle arrays separately
564        if(Array.isArray(value)) {
565          return value.map((item, index) => helper(item, depth + 1, path.concat(index)));
566        }
567
568        const result = {};
569
570        // Retrieve property descriptors without invoking getters
571        const descriptors = Object.getOwnPropertyDescriptors(value);
572        const keys = Reflect.ownKeys(descriptors); // Includes symbol
573        properties

```

```

573
574     for (const key of keys) {
575         const descriptor = descriptors[key];
576         const newPath = path.concat(key); // Build the full path
577
578         try {
579             if ('value' in descriptor) {
580                 // Data property; process the value
581                 result[key] = helper(descriptor.value, depth + 1, newPath);
582             } else if (typeof descriptor.get === 'function') {
583                 // Accessor property with a getter
584                 result[key] = '[Getter]';
585             } else {
586                 result[key] = '[Unknown Property]';
587             }
588         } catch (err) {
589             result[key] = '[Error: ${escapeString(err.message)}]';
590         }
591     }
592     return result;
593 } else if (typeof value === 'string') {
594     // Escape strings to prevent HTML injection
595     return escapeString(value);
596 }
597 return value;
598 }
599
600 return JSON.stringify(helper(data, 0, []), null, 2);
601 }
602
603 /**
604 * Escapes special HTML characters in a string to prevent HTML injection.
605 * @param {string} string - The string to escape.
606 * @returns {string} - The escaped string with HTML characters replaced.
607 */
608 escapeHTML(string) {
609     var escapeHtmlMap = {
610         '&': '&amp;',
611         '<': '&lt;',
612         '>': '&gt;',
613         '"': '&quot;',
614         "'": '&#39;',
615         '/': '&#x2F;',
616         '=': '&#x60;',
617         '_': '&#x3D;'
618    };
619
620    return String(string).replace(/[\&<>"'=\\]/g, function(s) {
621        return escapeHtmlMap[s];
622    });
623 }
624
625 /**
626 * Escapes special LaTeX characters in a string to prevent LaTeX injection.
627 * @param {string} string - The string to escape.

```

```

628     * @returns {string} - The escaped string with LaTeX characters replaced .
629     */
630     escapeLatex(string) {
631       if (typeof string !== 'string') {
632         return string;
633       }
634       return string
635         .replace(/\$/g, '\\textbackslash{}')
636         .replace(/\&/g, '\\&')
637         .replace(/\%/g, '\\%')
638         .replace(/\$/g, '\\
639
640 % -----
641 % End of document
642 %
643 \end{document})
644   .replace(/\#/g, '\\#')
645   .replace(/\_g, '\\_')
646   .replace(/\{g, '\\{')
647   .replace(/\}/g, '\\}')
648   .replace(/\^/g, '\\textasciitilde{}')
649   .replace(/\^/g, '\\textasciicircum{}')
650   .replace(/\`/g, '\\textasciigrave{}');
651 }
652
653 /**
654  * Generates a unique object key by appending a number to the original key
655  * if it already exists.
656  * @param {string} object - Object to add unique key.
657  * @param {string} key - The original object key.
658  * @returns {string} A unique object key.
659  */
660 getUniqueKey(object, key) {
661   var i = 0;
662   var unique_key = key;
663   while (hasKey(object, unique_key)) {
664     i = i+1;
665     unique_key = key+i;
666   }
667   return unique_key;
668 }
669 /**
670  * Generates a random string of the specified length.
671  * @param {number} num - The desired length of the random string.
672  * @returns {string} A random string.
673  */
674 randomString(num) {
675   return Math.random().toString(36).substr(2, num);
676 }
677
678 /**
679  * Calculates the number of decimal places in a number.
680  * @param {number} num - The number to evaluate.
681  * @returns {number} The count of decimal places.

```

```

682   */
683   countDecimalPlaces(num) {
684     return num > 1 ? 0 : (num.toString().split('.')[1] || '').match(/^0*/)[0].length+1;
685   }
686
687 /**
688 * Replaces file links in a text with HTML span elements.
689 * @param {string} text - The multiline error log text.
690 * @returns {string} The updated text with file links replaced by HTML spans
691 */
692 replaceEditorLinks(text) {
693   var regex = /(.*?):(\d+):(\d+)/g;
694
695   return text.replace(regex, (match, filePath, lineNumber, charPos) => {
696     return '<span class="open-editor" file_path="$' + filePath + '" line_number="' +
697       lineNumber + '" char_pos="' + charPos + '">' + match + '</span>';
698   });
699
700 /**
701 * Checks that an input contains a finite number
702 * @param {HTMLInputElement} n Target input element.
703 * @returns {boolean} True when the value is valid.
704 */
705 numberValidator(n) {
706   var str = n.value.trim();
707   var num = Number(str);
708   return str !== '' && Number.isFinite(num);
709 }
710
711 /**
712 * Checks that an input contains a finite number
713 * and if supplied is within the inclusive range [min, max].
714 * @param {HTMLInputElement} n Target input element.
715 * @param {number} [min] Minimum allowed value (optional).
716 * @param {number} [max] Maximum allowed value (optional).
717 * @returns {boolean} True when the value is valid.
718 */
719 limitedNumberValidator(n, min, max) {
720   var str = n.value.trim();
721   var num = Number(str);
722   if(str === '' || !Number.isFinite(num)) return false;
723   if(min !== undefined && num < min) return false;
724   if(max !== undefined && num > max) return false;
725   return true;
726 }
727 }
728
729 exports.PRDC_JSLAB_LIB_FORMAT = PRDC_JSLAB_LIB_FORMAT;

```

Listing 93 - format.js

```

1 /**
2 * @file JSLAB FreeCADLink submodule

```



```

56   this.loaded = true;
57   this.jsl.env disp('@FreeCADLink: '+language.string(179));
58   await this.findServer();
59 } else {
60   this.jsl.system.exec('' + this.exe + '--single-instance');
61   var [flag, pids] = this.jsl.system.isProgramRunning('FreeCAD.exe');
62   if(flag) {
63     this.loaded = true;
64     await this.findServer();
65   } else {
66     this.jsl.env disp('@FreeCADLink: '+language.string(180));
67   }
68 }
69 }
70 /**
71 * Attempts to locate the FreeCAD TCP server within the network, respecting
72 * the startup timeout.
73 * Checks if the TCP server is reachable by sending a 'PING' command.
74 */
75 async findServer() {
76   var t = tic;
77   while(true) {
78     if(await this.send('PING|', 1000)) {
79       break;
80     } else if(toc(t) > this.startup_timeout/1000) {
81       this.loaded = false;
82       this.jsl.env disp('@FreeCADLink: '+language.string(181));
83       break;
84     }
85     await waitSeconds(0.5);
86   }
87 }
88 /**
89 * Sends a message to the FreeCAD TCP server and waits for a response.
90 * Manages the TCP communication by ensuring message integrity and handling
91 * timeouts.
92 * @param {string} message - The message to send.
93 * @param {number} timeout - Timeout in milliseconds to wait for a response.
94 * @returns {Buffer|boolean} - The response from the server or false if the
95 * request times out.
96 */
97 async send(message, timeout) {
98   var obj = this;
99   var buf_in = [];
100  var N_in;
101  var got_header = false;
102  if(this.jsl.format.isDefined(timeout)) {
103    timeout = this.timeout;
104  }
105  if(this.loaded) {
106    var t = tic;
107

```

```

108   this.tcp_com = this.jsl.networking.tcp(host, port, function() {
109     var length = message.length;
110     var buf_out = Buffer.alloc(2 + length);
111     buf_out.writeUInt16BE(length, 0);
112     buf_out.write(message, 2, 'utf8');
113     obj.tcp_com.write(buf_out);
114   });
115   while(true) {
116     await waitMSeconds(1);
117     if(toc(t) < timeout/1000) {
118       if(!got_header && this.tcp_com.availableBytes() > 2) {
119         got_header = true;
120         var data = this.tcp_com.read();
121         var header = data.splice(0, 2);
122         N_in = (header[0] << 8) | header[1];
123         buf_in.push(...data);
124         if(buf_in.length == N_in) {
125           break;
126         }
127       } else if(got_header && this.tcp_com.availableBytes() > 0) {
128         var data = this.tcp_com.read();
129         if(data.length) {
130           buf_in.push(...data);
131           if(buf_in.length >= N_in) {
132             buf_in = buf_in.slice(0, N_in);
133             break;
134           }
135         }
136       } else {
137         break;
138       }
139     }
140   }
141   this.tcp_com.close();
142   if(buf_in.length) {
143     return Buffer.from(buf_in).toString();
144   }
145 }
146
147 return false;
148 }
149
150 /**
151 * Parses the input from FreeCAD responses to identify errors and data.
152 * Splits the message by '|' and checks for error or data messages.
153 * @param {string} message - The message received from FreeCAD.
154 * @returns {Array} - An array of parsed message components.
155 */
156 inputPraser(message) {
157   var params = [];
158   var str = message.toString();
159
160   if(str.length) {
161     params = str.split('|');
162     if(params.length && params[0] == 'ERR') {

```

```

163         disp( '@FreeCADLink: Error = ' + params[1]) ;
164     }
165   }
166   return params;
167 }
168
169 /**
170 * Displays a message from FreeCAD in the JSLAB interface.
171 * Parses and displays messages specifically tagged as 'MSG' from FreeCAD.
172 * @param {string} message - The message received from FreeCAD.
173 */
174 showMessage(message) {
175   var params = this.inputPraser(message);
176   if(params.length && params[0] == 'MSG') {
177     this.jsl.env.disp( '@FreeCADLink: Message = ' + params[1]);
178   }
179 }
180
181 /**
182 * Closes the FreeCAD application gracefully.
183 * Sends a quit command and handles the termination of the TCP connection.
184 */
185 async quit() {
186   if(this.loaded) {
187     var response = await this.send('CMD|QUIT', 1000);
188     return this.inputPraser(response);
189   } else {
190     this.jsl.env.disp( '@FreeCADLink: '+language.string(182));
191   }
192   return false;
193 }
194
195 /**
196 * Opens a specified file in FreeCAD.
197 * Sends a command to open a file and handles responses to confirm file
198 * access.
199 * @param {string} filePath - The path to the file to be opened.
200 * @param {number} timeout - Timeout in milliseconds to wait for a response.
201 */
202 async open(filePath, timeout) {
203   if(this.loaded) {
204     if(exist(filePath)) {
205       var response = await this.send('CMD|OPEN| ' + filePath, timeout);
206       return this.inputPraser(response);
207     } else {
208       this.jsl.env.disp( '@FreeCADLink: '+language.string(183));
209     }
210   } else {
211     this.jsl.env.disp( '@FreeCADLink: '+language.string(182));
212   }
213   return false;
214 }
215
216 /**
217 * Imports a file into the current FreeCAD document.

```

```

217   * Sends an import command and handles responses to confirm the import
218   * operation.
219   * @param {string} filePath - The path of the file to be imported.
220   * @param {number} timeout - Timeout in milliseconds to wait for a response.
221   */
222   async importFile(filePath, timeout) {
223     if(this.loaded) {
224       if(exist(filePath)) {
225         var response = await this.send('CMD|IMPORT|' + filePath, timeout);
226         return this.inputPraser(response);
227       } else {
228         this.jsl.env.disp('@FreeCADLink: '+language.string(183));
229       }
230     } else {
231       this.jsl.env.disp('@FreeCADLink: '+language.string(182));
232     }
233     return false;
234   }
235 /**
236   * Creates a new document in FreeCAD, optionally specifying a filename.
237   * Sends a command to create a new document and handles the document
238   * creation response.
239   * @param {string} filename - Optional filename for the new document.
240   * @param {number} timeout - Timeout in milliseconds to wait for a response.
241   */
242   async newDocument(filename, timeout) {
243     if(this.loaded) {
244       var cmd = 'CMD|NEW';
245       if(!this.jsl.format.isDefined(filename)) {
246         cmd = cmd+"|"+filename;
247       }
248       var response = await this.send(cmd, timeout);
249       var params = this.inputPraser(response);
250       var name = "";
251       if(params.length && params[0] == 'DAT') {
252         name = params[1];
253       }
254       return name;
255     } else {
256       this.jsl.env.disp('@FreeCADLink: '+language.string(182));
257     }
258     return false;
259   }
260 /**
261   * Saves the current document in FreeCAD.
262   * Sends a save command and handles responses to confirm the save operation.
263   * @param {number} timeout - Timeout in milliseconds to wait for a response.
264   */
265   async save(timeout) {
266     if(this.loaded) {
267       var response = await this.send('CMD|SAVE', timeout);
268       return this.inputPraser(response);
269     } else {

```

```
270     this.jsl.env disp('@FreeCADLink: '+language.string(182));
271 }
272 return false;
273 }
274
275 /**
276 * Saves the current document in FreeCAD under a new filename.
277 * Sends a save as command and handles responses to confirm the operation.
278 * @param {string} filePath - The new file path for the document.
279 * @param {number} timeout - Timeout in milliseconds to wait for a response.
280 */
281 async saveAs(filePath, timeout) {
282     if(this.loaded) {
283         var response = await this.send('CMD|SAVEAS|' + filePath, timeout);
284         return this.inputPraser(response);
285     } else {
286         this.jsl.env disp('@FreeCADLink: '+language.string(182));
287     }
288     return false;
289 }
290
291 /**
292 * Closes the current document in FreeCAD.
293 * Sends a close command and handles responses to confirm the document
294 * closure.
295 * @param {number} timeout - Timeout in milliseconds to wait for a response.
296 */
297 async close(timeout) {
298     if(this.loaded) {
299         var response = await this.send('CMD|CLOSE', timeout);
300         return this.inputPraser(response);
301     } else {
302         this.jsl.env disp('@FreeCADLink: '+language.string(182));
303     }
304     return false;
305 }
306 /**
307 * Executes a command in FreeCAD and returns the evaluation result.
308 * Sends an evaluate command with the specified command string.
309 * @param {string} command - The command to be evaluated in FreeCAD.
310 * @param {number} timeout - Timeout in milliseconds to wait for a response.
311 */
312 async evaluate(command, timeout) {
313     if(this.loaded) {
314         var response = await this.send('EVAL|' + command, timeout);
315         return this.inputPraser(response);
316     } else {
317         this.jsl.env disp('@FreeCADLink: '+language.string(182));
318     }
319     return false;
320 }
321
322 /**
323 * Runs a script in FreeCAD with optional parameters.
```

```
324     * Sends a script command along with parameters and handles the script
325     * execution response.
326     * @param {string} script - The script to run.
327     * @param {string|array} param - Parameters to pass to the script.
328     * @param {number} timeout - Timeout in milliseconds to wait for a response.
329     */
330     async callScript(script, param, timeout) {
331         if(this.loaded) {
332             if(this.jsl.format.isUndefined(timeout)) {
333                 timeout = this.script_timeout;
334             }
335             if(this.jsl.format.isUndefined(param)){
336                 param = '';
337             } else if(isArray(param)){
338                 param = param.join(' | ');
339             }
340             var response = await this.send('SCRIPT|' + script + '|'+ param, timeout
341             );
342             return this.inputPraser(response);
343         } else {
344             this.jsl.env.disp('@FreeCADLink: '+language.string(182));
345         }
346         return false;
347     }
348     /**
349     * Retrieves the area of the selected object in FreeCAD.
350     * Sends a measure area command and parses the response to extract the area
351     * value.
352     * @param {number} timeout - Timeout in milliseconds to wait for a response.
353     */
354     async getArea(timeout) {
355         if(this.loaded) {
356             var response = await this.send('MSR|A', timeout);
357             var params = this.inputPraser(response);
358             var area = "";
359             if(params.length && params[0] == 'DAT') {
360                 area = params[1];
361             }
362             return area;
363         } else {
364             this.jsl.env.disp('@FreeCADLink: '+language.string(182));
365         }
366         return false;
367     }
368     /**
369     * Retrieves the volume of the selected object in FreeCAD.
370     * Sends a measure volume command and parses the response to extract the
371     * volume value.
372     * @param {number} timeout - Timeout in milliseconds to wait for a response.
373     */
374     async getVolume(timeout) {
375         if(this.loaded) {
376             var response = await this.send('MSR|V', timeout);
```

```

375     var params = this.inputPraser(response);
376     var vol = "";
377     if(params.length && params[0] == 'DAT') {
378         vol = params[1];
379     }
380     return vol;
381 } else {
382     this.jsl.env disp('@FreeCADLink: '+language.string(182));
383 }
384 return false;
385 }
386 }
387
388 exports.PRDC_JSLAB_FREECAD_LINK = PRDC_JSLAB_FREECAD_LINK;

```

Listing 94 - freecad-link.js

```

1 /**
2  * @file JSLAB library geography map 3D submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for map 3D.
10 */
11 class PRDC_JSLAB_GEOGRAPHY_MAP_3D {
12
13 /**
14  * Creates a new 3D map instance.
15  * @param {Object} jsl - The JSLAB instance or environment reference.
16  */
17 constructor(jsl, token) {
18     this.jsl = jsl;
19     this.token = token;
20
21     // Default camera parameters
22     this.latitude = 44.8768331;
23     this.longitude = 20.112352;
24     this.height = 1000; // meters
25     this.heading = 0; // degrees
26     this.pitch = -30; // degrees
27
28     this.wid;
29     this.ready = false;
30 }
31
32 /**
33  * Opens a window with Cesium and initializes 3D map.
34  * @returns {Promise<void>}
35  */
36 async createWindow() {
37     var wid = this.jsl.windows.openWindow('cesium.html');
38     this.wid = wid;
39     this.win = this.jsl.windows.open_windows[wid];

```

```

40   await this.jsl.windows.open_windows[wid].ready;
41   var context = this.jsl.windows.open_windows[wid].context;
42   context.imports_ready = false;
43   while (!context.imports_ready) {
44     if (typeof context.Cesium != 'undefined') {
45       context.imports_ready = true;
46     }
47     await this.jsl.non_blocking.waitMSeconds(1);
48   }
49   context.map_3d_cont = context.document.getElementById('map-3d-cont');
50   context.map_3d_cont.style = 'position: absolute; top:0; left:0; right:0;
      bottom:0;';
51
52   // Initialize Cesium Viewer
53   context.Cesium.Ion.defaultAccessToken = this.token;
54   this.viewer = new context.Cesium.Viewer(context.map_3d_cont, {
55     terrainProvider: await context.Cesium.CesiumTerrainProvider.
        fromIonAssetId(1),
56     timeline: false,
57     sceneModePicker: false,
58     baseLayerPicker: false,
59     geocoder: false,
60     infoBox: false,
61     selectionIndicator: false
62   });
63
64   this.context = context;
65   this.Cesium = context.Cesium;
66   this.ready = true;
67 }
68
69 /**
70 * Sets the camera view to the specified latitude, longitude, and height.
71 * @param {number} lat - Latitude.
72 * @param {number} lon - Longitude.
73 * @param {number} height - Height in meters.
74 * @param {number} [heading=0] - Heading in degrees.
75 * @param {number} [pitch=-30] - Pitch in degrees.
76 */
77 setView(lat, lon, height, heading = 0, pitch = -30) {
78   this.latitude = lat;
79   this.longitude = lon;
80   this.height = height;
81   this.heading = heading;
82   this.pitch = pitch;
83
84   if (this.ready && this.viewer) {
85     this.viewer.camera.flyTo({
86       destination: this.context.Cesium.Cartesian3.fromDegrees(lon, lat,
          height),
87       orientation: {
88         heading: this.context.Cesium.Math.toRadians(heading),
89         pitch: this.context.Cesium.Math.toRadians(pitch),
90         roll: 0.0
91       }
92     })
93   }

```

```

92         });
93     }
94   }
95
96 /**
97 * Adds a new entity to the 3D map.
98 * @param {Object} data - The data representing the entity to add.
99 * @returns {PRDC_JSLAB_GEOGRAPHY_MAP_3D_ENTITY} The newly created map
100    entity.
101 */
102 addEntity(data) {
103   return new PRDC_JSLAB_GEOGRAPHY_MAP_3D_ENTITY(this.jsl, this, data);
104 }
105 /**
106 * Removes all entities from the 3D map viewer.
107 */
108 removeAllEntities() {
109   this.viewer.entities.removeAll();
110 }
111
112 /**
113 * Animates the camera to fly to the specified entity.
114 * @param {Object} entity - The entity to fly to.
115 */
116 flyTo(entity) {
117   if(hasKey(entity, 'entity')) {
118     this.viewer.flyTo(entity.entity);
119   }
120 }
121 }
122
123 exports.PRDC_JSLAB_GEOGRAPHY_MAP_3D = PRDC_JSLAB_GEOGRAPHY_MAP_3D;
124
125 class PRDC_JSLAB_GEOGRAPHY_MAP_3D_ENTITY {
126
127 /**
128 * Creates a new 3D map entity and adds it to the viewer.
129 * @param {Object} jsl - The JSL environment object.
130 * @param {Object} map_3d - The 3D map instance where the entity will be
131   added.
132 * @param {Object} data - The data representing the entity.
133 */
134 constructor(jsl, map_3d, data) {
135   this.jsl = jsl;
136   this.map_3d = map_3d;
137   this.data = data;
138
139   this.entity = this.map_3d.viewer.entities.add(data);
140 }
141
142 /**
143 * Animates the camera to fly to this entity.
144 */
145 flyTo() {

```

```

145     this.map_3d.viewer.flyTo(this.entity);
146   }
147 }
```

Listing 95 - geography-map-3d.js

```

1  /**
2   * @file JSLAB library geography map submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for map.
10  */
11 class PRDC_JSLAB_GEOGRAPHY_MAP {
12
13 /**
14  * Creates a new map instance.
15  * @param {Object} jsl - The JSLAB instance or environment reference.
16  * @param {string} [tileset='OpenStreetMap'] - The name of the tileset to
17  * use.
18  */
19 constructor(jsl, tileset = 'OpenStreetMap') {
20   this.jsl = jsl;
21
22   // Default map parameters
23   this.lat = 44.8768331;
24   this.lon = 20.112352;
25   this.zoom = 12;
26   this.tileset = tileset;
27   this.tileset_url;
28
29   this.tilesets = {
30     // OpenStreetMap Standard
31     'OpenStreetMap': 'https://{}{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
32
33     // Stamen Design
34     'Stamen Toner': 'https://stamen-tiles.a.ssl.fastly.net/toner/{z}/{x}/{y}.png',
35     'Stamen Watercolor': 'https://stamen-tiles.a.ssl.fastly.net/watercolor/{z}/{x}/{y}.jpg',
36     'Stamen Terrain': 'https://stamen-tiles.a.ssl.fastly.net/terrain/{z}/{x}/{y}.jpg',
37
38     // CartoDB (CARTO)
39     'Carto Positron': 'https://{}{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png',
40     'Carto DarkMatter': 'https://{}{s}.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}{r}.png',
41
42     // OpenTopoMap
43     'OpenTopoMap': 'https://{}{s}.tile.opentopomap.org/{z}/{x}/{y}.png',
44
45     // Esri
```

```

45   'Esri WorldStreetMap': 'https://server.arcgisonline.com/ArcGIS/rest/
46     services/World_Street_Map/MapServer/tile/{z}/{y}/{x}' ,
47   'Esri WorldImagery': 'https://server.arcgisonline.com/ArcGIS/rest/
48     services/World_Imagery/MapServer/tile/{z}/{y}/{x}' ,
49   'Esri DarkGrayCanvas': 'https://server.arcgisonline.com/ArcGIS/rest/
50     services/Dark_Gray_Base/MapServer/tile/{z}/{y}/{x}' ,
51
52   // Wikimedia Maps
53   'Wikimedia Standard': 'https://maps.wikimedia.org/osm-intl/{z}/{x}/{y}.
54     png' ,
55   'Wikimedia Cycle Map': 'https://maps.wikimedia.org/osm-intl-cycle/{z}/{x}
56     /{y}.png' ,
57   'Wikimedia Transport Map': 'https://maps.wikimedia.org/osm-intl-
58     transport/{z}/{x}/{y}.png' ,
59 }
60
61   this.wid;
62   this.ready = false;
63 }
64
65 /**
66 * Opens a window with Leaflet and initializes the map.
67 * @returns {Promise<void>}
68 */
69 async createWindow() {
70   var wid = this.jsl.windows.openWindow('leaflet.html');
71   this.wid = wid;
72   this.win = this.jsl.windows.open_windows[wid];
73   await this.jsl.windows.open_windows[wid].ready;
74   var context = this.jsl.windows.open_windows[wid].context;
75   context.imports_ready = false;
76   while (!context.imports_ready) {
77     if(typeof context.L != 'undefined') {
78       context.imports_ready = true;
79     }
80     await this.jsl.non_blocking.waitMSeconds(1);
81   }
82   this.context = context;
83   context.map_cont = context.document.getElementById('map-cont');
84   context.map_cont.style = 'position: absolute; top:0; left:0; right:0; bottom
85     :0; ';
86   this.leaflet_obj = context.L.map('map-cont',
87     { attributionControl: false })
88   ).setView([this.lat, this.lon], this.zoom);
89   context.L.tileLayer(this.tileset_url).addTo(this.leaflet_obj);
90   this.ready = true;
91 }
92 /**

```

```

93  * Sets the center of the map to the specified latitude and longitude.
94  * This will update the internal state and move the Leaflet map if it is
95  * ready.
96  * @param {number} lat - Latitude.
97  * @param {number} lon - Longitude.
98  */
99  setCenter(lat, lon) {
100    this.lat = lat;
101    this.lon = lon;
102    if(this.ready && this.leaflet_obj) {
103      this.leaflet_obj.setView([lat, lon], this.zoom);
104    }
105  }
106 /**
107  * Sets the zoom level of the map.
108  * This will update the internal state and adjust the Leaflet map if it is
109  * ready.
110  * @param {number} zoom - The zoom level.
111  */
112 setZoom(zoom) {
113   this.zoom = zoom;
114   if(this.ready && this.leaflet_obj) {
115     this.leaflet_obj.setZoom(zoom);
116   }
117 }
118 /**
119  * Adds a marker to the map at the specified latitude and longitude.
120  * @param {number} lat - Latitude.
121  * @param {number} lon - Longitude.
122  * @returns {PRDC_JSLAB_GEOGRAPHY_MAP_MARKER|null} - The marker instance or
123  * null if map is not ready.
124  */
125 addMarker(lat, lon) {
126   return new PRDC_JSLAB_GEOGRAPHY_MAP_MARKER(this.jsl, this, lat, lon);
127 }
128
129 exports.PRDC_JSLAB_GEOGRAPHY_MAP = PRDC_JSLAB_GEOGRAPHY_MAP;
130
131 class PRDC_JSLAB_GEOGRAPHY_MAP_MARKER {
132
133 /**
134  * Creates a new marker and adds it to the map.
135  * @param {Object} jsl - The JSLAB instance or environment reference.
136  * @param {PRDC_JSLAB_GEOGRAPHY_MAP} map - The map instance.
137  * @param {number} lat - Latitude.
138  * @param {number} lon - Longitude.
139  */
140 constructor(jsl, map, lat, lon) {
141   this.jsl = jsl;
142   this.map = map;
143
144   this.leaflet_obj = this.map.context.L.marker([lat, lon]).addTo(this.map).

```

```
        leaflet_obj);
145  this.lat = lat;
146  this.lon = lon;
147 }
148
149 /**
150 * Sets a custom icon for the marker.
151 * @param {Object} iconOptions - Leaflet icon options: { iconUrl: '...', iconSize: [...] , iconAnchor: [...] , etc. }
152 */
153 setIcon(iconOptions) {
154  if(this.leaflet_obj && this.map.ready) {
155    let customIcon = this.map.context.L.icon(iconOptions);
156    this.leaflet_obj.setIcon(customIcon);
157    this.leaflet_obj.setRotationOrigin('center center');
158  }
159 }
160
161 /**
162 * Sets a custom icon for the marker.
163 * @param {Object} iconOptions - Leaflet divIcon options: { iconUrl: '...', iconSize: [...] , iconAnchor: [...] , etc. }
164 */
165 setDivIcon(iconOptions) {
166  if(this.leaflet_obj && this.map.ready) {
167    let customIcon = this.map.context.L.divIcon(iconOptions);
168    this.leaflet_obj.setIcon(customIcon);
169    this.leaflet_obj.setRotationOrigin('center center');
170  }
171 }
172
173 /**
174 * Sets the position of the marker.
175 * @param {number} lat - New latitude.
176 * @param {number} lon - New longitude.
177 */
178 setPosition(lat, lon) {
179  this.lat = lat;
180  this.lon = lon;
181  if(this.leaflet_obj && this.map.ready) {
182    this.leaflet_obj.setLatLng([lat, lon]);
183  }
184 }
185
186 /**
187 * Sets the rotation (in degrees) of the marker.
188 * Leaflet markers can be rotated via CSS transform if using a DivIcon or a
189 * custom class.
190 * This example uses a simple inline style transform if the marker's icon
191 * supports it.
192 * @param {number} rotation - Rotation angle in degrees.
193 */
194 setRotation(rotation) {
195  if(this.leaflet_obj && this.map.ready) {
196    this.leaflet_obj.setRotationAngle(rotation);
```

```
195      }
196    }
197 }
```

Listing 96 - geography-map.js

```
1  /**
2   * @file JSLAB library geography submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8 var { PRDC_JSLAB_GEOGRAPHY_MAP } = require('./geography-map');
9 var { PRDC_JSLAB_GEOGRAPHY_MAP_3D } = require('./geography-map-3d');
10
11 /**
12  * Class for JSLAB geography submodule.
13  */
14 class PRDC_JSLAB_LIB_GEOGRAPHY {
15
16 /**
17  * Initializes the geography submodule.
18  * @param {Object} js1 Reference to the main JSLAB object.
19  */
20 constructor(js1) {
21   var obj = this;
22   this.js1 = js1;
23 }
24
25 /**
26  * Initializes and returns a new 2D web map instance.
27  * @param {...*} args - Arguments for configuring the web map.
28  * @returns {Promise<PRDC_JSLAB_GEOGRAPHY_MAP>} The initialized 2D web map
29  * instance.
30  */
31 async webmap(...args) {
32   var map = new PRDC_JSLAB_GEOGRAPHY_MAP(this.js1, ...args);
33   await map.createWindow();
34   return map;
35 }
36
37 /**
38  * Initializes and returns a new 3D geoglobe instance.
39  * @param {...*} args - Arguments for configuring the 3D geoglobe.
40  * @returns {Promise<PRDC_JSLAB_GEOGRAPHY_MAP_3D>} The initialized 3D
41  * geoglobe instance.
42  */
43 async geoglobe(...args) {
44   var map_3d = new PRDC_JSLAB_GEOGRAPHY_MAP_3D(this.js1, ...args);
45   await map_3d.createWindow();
46   return map_3d;
47 }
48
49 /**
50  * Calculates the bounding box and center from an array of tile coordinates.
```

```

49   * @param {Array<Object>} tile_coordinates - An array of objects with tile
50     coordinates, each having properties 'x', 'y', and 'z' for tile X and Y
51     coordinates and zoom level, respectively.
52   * @returns {Object} An object containing the bounds as an array of '['
53     'min_lat, min_lng]' and '[max_lat, max_lng]', and the center as '['
54     'latitude, longitude]'.
55   */
56 calculateTilesBoundingBox(tile_coordinates) {
57   var min_lat = Number.MAX_VALUE,
58     max_lat = -Number.MAX_VALUE,
59     min_lng = Number.MAX_VALUE,
60     max_lng = -Number.MAX_VALUE;
61
62   tile_coordinates.forEach(function(coord) {
63     var n = Math.pow(2, coord.z);
64     var tileBounds = {
65       min_lat: (2 * Math.atan(Math.exp(Math.PI - (2 * Math.PI * coord.y)
66         / n)) - Math.PI / 2) * (180 / Math.PI),
67       max_lat: (2 * Math.atan(Math.exp(Math.PI - (2 * Math.PI * (coord.y
68         + 1)) / n)) - Math.PI / 2) * (180 / Math.PI),
69       min_lng: ((coord.x) / n) * 360 - 180,
70       max_lng: ((coord.x + 1) / n) * 360 - 180
71     };
72
73     min_lat = Math.min(min_lat, tileBounds.min_lat);
74     max_lat = Math.max(max_lat, tileBounds.max_lat);
75     min_lng = Math.min(min_lng, tileBounds.min_lng);
76     max_lng = Math.max(max_lng, tileBounds.max_lng);
77   });
78
79   var center = [(min_lat + max_lat) / 2, (min_lng + max_lng) / 2];
80   var bounds = [[min_lat, min_lng], [max_lat, max_lng]];
81
82   return { bounds, center };
83 }
84
85 /**
86  * Calculates a new latitude and longitude based on a starting point,
87  * distance, and bearing using the Haversine formula.
88  * @param {number} lat - The latitude of the starting point.
89  * @param {number} lon - The longitude of the starting point.
90  * @param {number} distance - The distance from the starting point in meters
91  *
92  * @param {number} bearing - The bearing in degrees from north.
93  * @returns {Array<number>} An array containing the latitude and longitude
94  * of the calculated point.
95  */
96 offsetLatLon(lat, lon, distance, bearing) {
97   var dist_rad = distance / 6371000;
98   var bearing_rad = (bearing * Math.PI) / 180;
99   var lat_rad = (lat * Math.PI) / 180;
100  var lng_rad = (lon * Math.PI) / 180;
101  var new_lat_rad = Math.asin(Math.sin(lat_rad) * Math.cos(dist_rad) +
102    Math.cos(lat_rad) * Math.sin(dist_rad) * Math.cos(bearing_rad));
103  var new_lng_rad = lng_rad + Math.atan2(Math.sin(bearing_rad) * Math.sin(
104    bearing_rad) * Math.cos(lat_rad), Math.cos(bearing_rad));
105
106  return [new_lat_rad * 180 / Math.PI, new_lng_rad * 180 / Math.PI];
107}

```

```

      dist_rad) * Math.cos(lat_rad),
      Math.cos(dist_rad) - Math.sin(lat_rad) * Math.sin(new_lat_rad));
    return [(new_lat_rad * 180) / Math.PI, (new_lng_rad * 180) / Math.PI];
}

/**
 * Calculates the distance between two points on Earth using the Haversine
 * formula.
 * @param {number} lat1 - The latitude of the first point.
 * @param {number} lon1 - The longitude of the first point.
 * @param {number} lat2 - The latitude of the second point.
 * @param {number} lon2 - The longitude of the second point.
 * @returns {number} The distance between the two points in meters.
 */
latLonDistance(lat1, lon1, lat2, lon2) {
  var dLat = (lat2-lat1) * Math.PI / 180;
  var dLon = (lon2-lon1) * Math.PI / 180;
  return 6371000 * 2 * Math.asin(Math.sqrt(Math.sin(dLat/2) * Math.sin(dLat
    /2) +
    Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
    Math.sin(dLon/2) * Math.sin(dLon/2))); // [m]
}

/**
 * Calculates the distance between two points on Earth including altitude
 * difference using the Haversine formula.
 * @param {number} lat1 - The latitude of the first point.
 * @param {number} lon1 - The longitude of the first point.
 * @param {number} alt1 - The altitude of the first point in meters.
 * @param {number} lat2 - The latitude of the second point.
 * @param {number} lon2 - The longitude of the second point.
 * @param {number} alt2 - The altitude of the second point in meters.
 * @returns {number} The 3D distance between the two points in meters.
 */
latLonAltDistance(lat1, lon1, alt1, lat2, lon2, alt2) {
  var L = latLonDistance(lat1, lon1, lat2, lon2);
  return Math.sqrt(Math.pow(L, 2)+Math.pow(alt1-alt2, 2));
}

/**
 * Checks if the latitude and longitude values have been updated.
 * @param {Object} latlon - An object containing the current and last values
 * of latitude and longitude.
 * @returns {boolean} True if the latitude and/or longitude values have been
 * updated; false otherwise.
 */
checkNewLatLon(latlon) {
  if(this.jsl.format.isDefined(latlon.value)) {
    return false;
  } else if(this.jsl.format.isDefined(latlon.last_value)) {
    return true;
  } else {
    return (latlon.value[0] != latlon.last_value[0]) || (latlon.value[1] !=
      latlon.last_value[1]);
  }
}

```

```

143 }
144
145 /**
146 * Converts latitude, longitude, and altitude to Cartesian coordinates.
147 * @param {number|number[]} lat - Latitude in degrees or array of latitudes.
148 * @param {number|number[]} lon - Longitude in degrees or array of
149 * longitudes.
150 * @param {number|number[]} alt - Altitude in meters or array of altitudes.
151 * @returns {Cesium.Cartesian3|Cesium.Cartesian3[]} Cartesian coordinate(s).
152 */
153 latLonAlt2cartesian(lat, lon, alt) {
154   var isArray = Array.isArray(lat) && Array.isArray(lon) && Array.isArray(
155     alt);
156
157   if(isArray) {
158     return lat.map((latitude, index) =>
159       this.jsl.env.Cesium.Cartesian3.fromDegrees(lon[index], latitude, alt[
160         index]));
161   } else {
162     return this.jsl.env.Cesium.Cartesian3.fromDegrees(lon, lat, alt);
163   }
164 }
165 exports.PRDC_JSLAB_LIB_GEOGRAPHY = PRDC_JSLAB_LIB_GEOGRAPHY;

```

Listing 97 - geography.js

```

1 /**
2  * @file JSLAB library geometry Space Search submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7 */
8
9 /**
10 * Class for N-dimensional Space Search.
11 */
12 class PRDC_JSLAB_GEOMETRY_SPACE_SERACH {
13
14 /**
15 * Displays the size of elements based on bounds and subdivisions.
16 * @param {Array} bounds - Array of [min, max] for each dimension.
17 * @param {Array} subdivisionPerDepth - Subdivision factors per depth and
18 * dimension.
19 */
20 dispElementSize(bounds, subdivisionPerDepth) {
21   var numDimensions = bounds.length;
22   // Calculate the initial differences between bounds for each dimension
23   var dq = bounds.map((bound, index) => (bound[1] - bound[0]) /
24     subdivisionPerDepth[0][index]);
25
26   // Copy dq to dqmin for further subdivision calculations
27   var dqmin = [...dq];

```

```

26
27 // Iterate through subdivisionPerDepth starting from the second depth
28 // level
29 for (var i = 1; i < subdivisionPerDepth.length; i++) {
30   for (var j = 0; j < numDimensions; j++) {
31     dqmin[j] /= subdivisionPerDepth[i][j];
32   }
33
34 // Display the results for initial and minimal element dimensions
35 disp(` Dimenzije inicijalnih elemenata: ${dq.map(val => val.toFixed(2)) .
36   join('x'))}`);
37 disp(` Dimenzije najmanjih elemenata: ${dqmin.map(val => val.toFixed(2)) .
38   join('x'))}`);
39
40 /**
41 * Splits the search space into smaller intervals for parallel processing.
42 * @param {Array.<Array.<number>>} x_lim - The limits of the search space,
43 *   where each sub-array represents [start, end] for a dimension.
44 * @param {Array.<Array.<number>>} k - The parameters to optimize.
45 * @param {number} N_proc - The number of processors to divide the work
46 *   among.
47 * @returns {Array.<Array.<number>>} An array containing the split search
48 *   spaces and the adjusted parameters.
49 */
50 splitSearchSpace(x_lim, k, N_proc) {
51   const [start, end] = x_lim[0];
52   const interval = (end - start) / N_proc;
53
54   var k_out = [...k];
55   k_out[0][0] = k_out[0][0] / N_proc;
56
57   return [Array.from({ length: N_proc }, (_, i) => [
58     [start + i * interval, start + (i + 1) * interval],
59     ...x_lim.slice(1)
60   ]), k_out];
61 }
62
63 /**
64 * Executes a provided function in parallel across the split search spaces.
65 * @param {Array.<Array.<number>>} x_lim - The limits of the search space,
66 *   where each sub-array represents [start, end] for a dimension.
67 * @param {Array.<Array.<number>>} k - The parameters to optimize.
68 * @param {Object} context - The execution context containing necessary
69 *   configurations and states.
70 * @param {Function} setup - The setup function to initialize the parallel
71 *   environment.
72 * @param {Function} fun - The function to execute in parallel on each split
73 *   of the search space.
74 * @returns {Promise<Array.<Array.<number>>>} A promise that resolves to
75 *   arrays of input and output results from the parallel execution.
76 */
77 async runParallel(x_lim, k, context, setup, fun) {
78   var N_proc = parallel.getProcessorsNum();
79
80   const [spaces, k_out] = splitSearchSpace(x_lim, k, N_proc);
81
82   const promises = spaces.map(space => {
83     const [start, end] = space[0];
84     const [k_start, k_end] = k_out[0];
85
86     return new Promise((resolve, reject) => {
87       const parallelContext = { ...context, k: k_start, k_end };
88
89       parallel.parallelize(
90         { start, end },
91         { k: k_start, k_end },
92         { setup, fun },
93         { context: parallelContext }
94       ).then(result => {
95         resolve(result);
96       }).catch(error => {
97         reject(error);
98       });
99     });
100
101   const results = await Promise.all(promises);
102
103   return results;
104 }

```

```

70
71     var funStr = fun.toString();
72     var funBody = funStr.slice(funStr.indexOf("{") + 1, funStr.lastIndexOf("}") );
73
74     var funArgs = funStr.slice(funStr.indexOf("(") + 1, funStr.indexOf(")") );
75
76     var [x_lim_parallel, k_parallel] = this.splitSearchSpace(x_lim, k, N_proc);
77
78     var res = await parallel.parfor(0, N_proc-1, 1, N_proc,
79         Object.assign(context, {x_lim_parallel, k_parallel,
80             funArgs, funBody}), setup, function(i) {
81         var new_fun = new Function(funArgs, funBody);
82         return crawler.run(x_lim_parallel[i], k_parallel, new_fun);
83     });
84     var Nin = res.map(pair => pair[0]).flat();
85     var Nout = res.map(pair => pair[1]).flat();
86     return [Nin, Nout];
87 }
88
89 /**
90 * Executes the space search algorithm.
91 * @param {number[][]} bounds - Array of [min, max] for each dimension.
92 * @param {number[][]} subdivisionPerDepth - Subdivision factors for each
93 *     depth and dimension.
94 * @param {function} conditionFunction - Function that determines if a point
95 *     satisfies the condition.
96 * @returns {Array<Array<number>>} - Arrays of points inside and outside the
97 *     condition.
98 */
99 run(bounds, subdivisionPerDepth, conditionFunction) {
100    var numDimensions = bounds.length;
101    var maxDepth = subdivisionPerDepth.length;
102
103    // Initial starting point
104    var startCoordinates = bounds.map(([min]) => min);
105
106    // Calculate initial step sizes (dq)
107    var initialStepSizes = bounds.map(
108        ([min, max], idx) => (max - min) / subdivisionPerDepth[0][idx]
109    );
110
111    // Generate step sizes for each depth
112    var stepSizesPerDepth = [initialStepSizes];
113    for(var depth = 1; depth < maxDepth; depth++) {
114        var previousStepSizes = stepSizesPerDepth[depth - 1];
115        var newStepSizes = previousStepSizes.map(
116            (size, idx) => size / subdivisionPerDepth[depth][idx]
117        );
118        stepSizesPerDepth.push(newStepSizes);
119    }
120
121    // Compute the values at the boundary points
122    var cornerShifts = this.generateCornerShifts(numDimensions);
123    var boundaryPoints = cornerShifts.map(shift => {

```

```

120     return shift.map((s, idx) => startCoordinates[idx] + s * (bounds[idx][1]
121         - bounds[idx][0]));
122   });
123   var boundaryValues = boundaryPoints.map(point => (conditionFunction(point)
124     ? 1 : 0));
125
126   // Call makeNodesND
127   var [Nin, Nout] = this.makeNodesND(
128     startCoordinates,
129     boundaryValues,
130     0,
131     stepSizesPerDepth,
132     subdivisionPerDepth,
133     conditionFunction
134   );
135
136   return [Nin, Nout];
137 }
138
139 /**
140  * Recursively creates nodes in N dimensions based on subdivisions.
141  * @param {number[]} startCoordinates - Starting coordinates for the current
142  *   grid.
143  * @param {number[]} boundaryValues - Values at the boundary points of the
144  *   current hypercube.
145  * @param {number} currentDepth - Current depth of the recursion.
146  * @param {number[][]} stepSizesPerDepth - Step sizes for each depth.
147  * @param {number[][]} subdivisionPerDepth - Subdivision factors for each
148  *   depth.
149  * @param {function} conditionFunction - User-defined condition function.
150  * @returns {Array<Array<number>>} - Arrays of points inside and outside the
151  *   condition.
152  */
153 makeNodesND(
154   startCoordinates,
155   boundaryValues,
156   currentDepth,
157   stepSizesPerDepth,
158   subdivisionPerDepth,
159   conditionFunction
160 ) {
161   var numDimensions = startCoordinates.length;
162   var maxDepth = subdivisionPerDepth.length; // maximum depth
163   var Nin = [];
164   var Nout = [];
165
166   // Generate coordinate arrays for each dimension
167   var stepSizes = stepSizesPerDepth[currentDepth];
168   var numSteps = subdivisionPerDepth[currentDepth];
169   var coordsArray = [];
170   for(var idx = 0; idx < numDimensions; idx++) {
171     var steps = numSteps[idx];
172     var stepSize = stepSizes[idx];
173     var start = startCoordinates[idx];
174     var arr = Array.from(
175       start
176       .map((val) => val + stepSize / (steps + 1))
177       .concat(start)
178       .map((val) => val - stepSize / (steps + 1))
179     ).map((val) => val - start);
180     coordsArray.push(arr);
181   }
182
183   if (currentDepth < maxDepth) {
184     var newStartCoordinates = coordsArray.map((arr) => arr.map((val) => val));
185     var newBoundaryValues = boundaryValues.map((val) => val);
186     var newStepSizesPerDepth = stepSizesPerDepth.map((arr) => arr);
187     var newSubdivisionPerDepth = subdivisionPerDepth.map((arr) => arr);
188     var newConditionFunction = conditionFunction;
189
190     var [newNin, newNout] = makeNodesND(
191       newStartCoordinates,
192       newBoundaryValues,
193       currentDepth + 1,
194       newStepSizesPerDepth,
195       newSubdivisionPerDepth,
196       newConditionFunction
197     );
198
199     Nin = Nin.concat(newNin);
200     Nout = Nout.concat(newNout);
201   }
202
203   return [Nin, Nout];
204 }

```

```

169      { length: steps + 1 },
170      (_ , i) => start + i * stepSize
171    );
172    coordsArray.push(arr);
173  }
174
175  // Initialize N-dimensional grid
176  var gridShape = coordsArray.map(A => A.length);
177  var N = createFilledArray(...gridShape, null);
178
179  // Mark whether a node needs to be calculated
180  var nf = createFilledArray(...gridShape, 1);
181
182  // Set nf to 0 for corner points (boundary points)
183  var cornerIndices = this.getCornerIndices(gridShape);
184  cornerIndices.forEach(indices => {
185    setValueAt(nf, indices, 0);
186  });
187
188  // Set the values at the corner points and add them to Nin or Nout
189  cornerIndices.forEach((indices, idx) => {
190    var coords = indices.map((index, dim) => coordsArray[dim][index]);
191    var value = boundaryValues[idx];
192    setValueAt(N, indices, [...coords, value]);
193
194  // On first level, add corner points to Nin or Nout
195  if(currentDepth == 0) {
196    if(value) {
197      Nin.push(coords);
198    } else {
199      Nout.push(coords);
200    }
201  }
202});
203
204 // Now compute the values at all nodes where nf is 1
205 var indicesList = this.generateIndicesList(gridShape, nf);
206 for(var indices of indicesList) {
207   var coords = indices.map((index, dim) => coordsArray[dim][index]);
208   var e = conditionFunction(coords) ? 1 : 0;
209   setValueAt(N, indices, [...coords, e]);
210
211   if(e) {
212     Nin.push(coords);
213   } else {
214     Nout.push(coords);
215   }
216 }
217
218 // Check for further subdivision
219 if(currentDepth < maxDepth - 1) {
220   var innerCubeIndicesList = this.generateInnerCubeIndicesList(gridShape);
221   for(var indices of innerCubeIndicesList) {
222     var cubeCorners = this.getCubeCorners(N, indices);
223     var boundaryValues_sub = cubeCorners.map(corner => corner[corner.

```

```

length - 1]) ;

224
225   var allSame = boundaryValues_sub.every(val => val ===
226     boundaryValues_sub[0]);
227   if (!allSame) {
228     var startCoords_sub = cubeCorners[0].slice(0, numDimensions);
229     var [Nin_sub, Nout_sub] = this.makeNodesND(
230       startCoords_sub,
231       boundaryValues_sub,
232       currentDepth + 1,
233       stepSizesPerDepth,
234       subdivisionPerDepth,
235       conditionFunction
236     );
237     Nin = Nin.concat(Nin_sub);
238     Nout = Nout.concat(Nout_sub);
239   }
240 }
241
242 return [Nin, Nout];
243 }

244 /**
245 * Generates all corner shifts for a hypercube in N dimensions.
246 * @param {number} numDimensions - Number of dimensions.
247 * @returns {number[][]} - Array of shifts for each corner.
248 */
249 generateCornerShifts(numDimensions) {
250   var shifts = [];
251   var totalCorners = 1 << numDimensions;
252   for(var i = 0; i < totalCorners; i++) {
253     var shift = [];
254     for(var j = 0; j < numDimensions; j++) {
255       shift.push((i >> j) & 1);
256     }
257     shifts.push(shift);
258   }
259   return shifts;
260 }

261 /**
262 * Retrieves the indices of corner points in the grid.
263 * @param {number[]} gridShape - Shape of the grid.
264 * @returns {number[][]} - List of corner indices.
265 */
266 getCornerIndices(gridShape) {
267   var numDimensions = gridShape.length;
268   var cornerShifts = this.generateCornerShifts(numDimensions);
269   return cornerShifts.map(shift => shift.map((s, idx) => s * (gridShape[idx]
270     - 1)));
271 }
272
273 /**
274 * Generates a list of indices for nodes that need to be calculated.
275 */

```

```

276  * @param {number[]} gridShape - Shape of the grid.
277  * @param {Array} nf - N-dimensional array indicating nodes to compute.
278  * @returns {number[][]} - List of node indices where nf is 1.
279  */
280 generateIndicesList(gridShape, nf) {
281   var indicesList = [];
282
283   var generateIndices = (indices, dim) => {
284     if(dim === gridShape.length) {
285       if(getValueAt(nf, indices) === 1) {
286         indicesList.push([... indices]);
287       }
288       return;
289     }
290     for(var i = 0; i < gridShape[dim]; i++) {
291       indices[dim] = i;
292       generateIndices(indices, dim + 1);
293     }
294   };
295   generateIndices(new Array(gridShape.length).fill(0), 0);
296   return indicesList;
297 }
298 /**
299 * Generates a list of starting indices for inner hypercubes.
300 * @param {number[]} gridShape - Shape of the grid.
301 * @returns {number[][]} - List of inner cube starting indices.
302 */
303 generateInnerCubeIndicesList(gridShape) {
304   var indicesList = [];
305
306   var generateIndices = (indices, dim) => {
307     if(dim === gridShape.length) {
308       indicesList.push([... indices]);
309       return;
310     }
311     if(gridShape[dim] <= 1) {
312       // No cubes along this dimension
313       return;
314     }
315     for(var i = 0; i < gridShape[dim] - 1; i++) {
316       indices[dim] = i;
317       generateIndices(indices, dim + 1);
318     }
319   };
320   generateIndices(new Array(gridShape.length).fill(0), 0);
321   return indicesList;
322 }
323 /**
324 * Retrieves the corner coordinates and values of a hypercube.
325 * @param {Array} N - N-dimensional grid.
326 * @param {number[]} indices - Starting indices of the hypercube.
327 * @returns {Array} - Array of corner coordinates and values.
328 */
329
330

```

```

331     getCubeCorners(N, indices) {
332         var numDimensions = indices.length;
333         var cornerShifts = this.generateCornerShifts(numDimensions);
334         var cubeCorners = cornerShifts.map(shift => {
335             var cornerIndices = indices.map((idx, dim) => idx + shift[dim]);
336             return getValueAt(N, cornerIndices);
337         });
338         return cubeCorners;
339     }
340 }
341
342 exports.PRDC_JSLAB_GEOMETRY_SPACE_SERACH = PRDC_JSLAB_GEOMETRY_SPACE_SERACH;

```

Listing 98 - geometry-spacesearch.js

```

1  /**
2  * @file JSLAB library geometry submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 var { PRDC_JSLAB_GEOMETRY_SPACE_SERACH } = require('./geometry-spacesearch');
9
10 /**
11 * Class for JSLAB geometry submodule.
12 */
13 class PRDC_JSLAB_LIB_GEOMETRY {
14
15 /**
16 * Initializes a new instance of the geometry submodule, providing access to
17 * geometry manipulation utilities.
18 * @param {Object} js1 Reference to the main JSLAB object.
19 */
20 constructor(js1) {
21     var obj = this;
22     this.js1 = js1;
23 }
24 /**
25 * Creates an instance of PRDC_JSLAB_LIB_OPTIM_SPACE_SERACH.
26 */
27 spaceSearch(...args) {
28     return new PRDC_JSLAB_GEOMETRY_SPACE_SERACH(...args);
29 }
30
31 /**
32 * Finds the nearest points in points2 for each point in points1.
33 * @param {Array[]} points1 - Array of points.
34 * @param {Array[]} points2 - Reference points.
35 * @returns {number[]} Array of indices for nearest points.
36 */
37 findNearestPoints(points1, points2) {
38     var L;
39     var id = this.js1.array.createFilledArray(points1.length, -1);
40     for(var i = 0; i < points1.length; i++) {

```

```

41   var L_min;
42   for(var j = 0; j < points2.length; j++) {
43     if(points1[0].length == 3) {
44       L = Math.sqrt(Math.pow(points1[i][0] - points2[j][0], 2) +
45                     Math.pow(points1[i][1] - points2[j][1], 2) +
46                     Math.pow(points1[i][2] - points2[j][2], 2));
47     } else {
48       L = Math.sqrt(Math.pow(points1[i][0] - points2[j][0], 2) +
49                     Math.pow(points1[i][1] - points2[j][1], 2));
50     }
51     if(j == 0 || L < L_min){
52       L_min = L;
53       id[i] = j;
54     }
55   }
56   return id;
57 }
58 }
59 /**
60 * Returns the shortest distance from point P to the line defined by (A, i)
61 * and the closest point (P1) on that line.
62 * @param {number[]} P - point [Px, Py, Pz]
63 * @param {number[]} A - a point on the line
64 * @param {number[]} i - a unit direction vector of the line
65 * @returns {{ d: number, P1: number[] }}
66 *   d - shortest distance
67 *   P1 - point on the line with the smallest distance to P
68 */
69 pointLineDistance(P, A, i) {
70   // P1 = A + dot(P - A, i) * i
71   const PA = minus(P, A);
72   const distAlongI = dot(PA, i);
73   const P1 = plus(A, scale(i, distAlongI));
74
75   // Distance = || P - P1 ||
76   const d = norm(minus(P, P1));
77
78   return { d, P1 };
79 }
80 /**
81 * Returns the intersection points of a circle (center O, radius r)
82 * in a plane with a line passing through point P with direction i.
83 * @param {number[]} P - point on the line
84 * @param {number[]} i - direction vector of the line (unit)
85 * @param {number[]} O - center of the circle
86 * @param {number} r - circle radius
87 * @returns {{ P1: number[] | null, P2: number[] | null, flag: number }}
88 *   flag = 0 -> intersection (two points)
89 *   flag = 1 -> tangent (one point)
90 *   flag = 2 -> no intersection
91 */
92 lineCircleIntersection(P, i, O, r) {
93   let P1 = null;
94   let P2 = null;

```

```

96   let flag = 0;
97
98   // Distance from O to line & the closest point A
99   const { d, P1: A } = this.pointLineDistance(O, P, i);
100
101  if(d < r) {
102    // Two intersection points
103    const h = Math.sqrt(r * r - d * d); // half of the chord length
104    P1 = plus(A, scale(i, h));
105    P2 = plus(A, scale(i, -h));
106    flag = 0; // intersection
107  } else if(Math.abs(d - r) <= EPS) {
108    // Tangent
109    P1 = A;
110    flag = 1;
111  } else {
112    // No intersection
113    flag = 2;
114  }
115
116  return { P1, P2, flag };
117}
118
119 /**
120 * Returns the line (point P, direction i) that is the intersection
121 * of two planes, or indicates if they are the same or parallel.
122 * @param {number[]} P1 - a point in plane 1
123 * @param {number[]} n1 - normal to plane 1
124 * @param {number[]} P2 - a point in plane 2
125 * @param {number[]} n2 - normal to plane 2
126 * @returns {{ P: number[] | null, i: number[] | null, flag: number }}
127 *   flag = 0 -> planes intersect
128 *   flag = 1 -> planes are the same
129 *   flag = 2 -> planes are parallel (no intersection)
130 */
131 planesIntersection(P1, n1, P2, n2) {
132   let P = null;
133   let i = null;
134   let flag = 0;
135
136   const V = cross3D(n1, n2, 1);
137
138   const V_norm = norm(V);
139   if(V_norm > EPS) {
140     // planes intersect
141     i = scale(V, 1.0 / V_norm); // unit direction
142
143     // Solve for a point on the intersection line:
144     // We want to solve the system:
145     // dot(n1, X) = dot(n1, P1)
146     // dot(n2, X) = dot(n2, P2)
147     //
148     // We'll attempt x=0, y=0, z=0 approach or check sub-determinants.
149     const A = [
150       [n1[0], n1[1], n1[2]],

```

```

151      [ n2[0] ,  n2[1] ,  n2[2] ] ,
152    ];
153    const B = [ dot(n1, P1) ,  dot(n2, P2) ];
154
155    // We try ignoring one coordinate at a time (k=1 to 3):
156    let solved = false;
157    for(let k = 0; k < 3; k++) {
158      // Indices [0,1,2], skip k => j
159      const j = [0, 1, 2].filter(idx => idx !== k);
160
161      // Build a 2x2 submatrix from A, using columns j[0], j[1]
162      const subA = [
163        [A[0][j[0]], A[0][j[1]]] ,
164        [A[1][j[0]], A[1][j[1]]] ,
165      ];
166      const detSubA = subA[0][0] * subA[1][1] - subA[0][1] * subA[1][0];
167
168      if(Math.abs(detSubA) > EPS) {
169        // We can solve
170        // subA * C = B
171        // C is 2x1 => we solve by 2x2 inverse
172        const invDet = 1.0 / detSubA;
173        const C0 = invDet * ( B[0]*subA[1][1] - B[1]*subA[0][1] );
174        const C1 = invDet * (-B[0]*subA[1][0] + B[1]*subA[0][0]);
175
176        // Build full solution X
177        const X = [0, 0, 0];
178        X[j[0]] = C0;
179        X[j[1]] = C1;
180        P = X;
181        solved = true;
182        break;
183      }
184    }
185    if(!solved) {
186      // fallback: planes might be very close, etc.
187      P = [0, 0, 0];
188    }
189  } else {
190    // Check if they are the same plane
191    // We test if P2 satisfies plane 1 => dot(n1, P1-P2)=0 (within EPS).
192    const diff = minus(P1, P2);
193    if(Math.abs(dot(n1, diff)) <= EPS) {
194      // same plane
195      flag = 1;
196    } else {
197      // parallel planes
198      flag = 2;
199    }
200  }
201
202  return { P, i, flag };
203}
204
205 /**

```

```

206  * Checks if point P lies on the line segment A-B.
207  * Returns 1 if on segment , 0 otherwise.
208  * @param {number[]} P
209  * @param {number[]} A
210  * @param {number[]} B
211  * @returns {number} 1 (on segment) , 0 (not on segment)
212  */
213 isPointOnLine(P, A, B) {
214   // i = (B - A) / ||B - A||
215   const AB = minus(B, A);
216   const i = scale(AB, 1.0 / norm(AB));
217
218   // dot(A - P, i) and dot(B - P, i)
219   const d1 = dot(minus(A, P), i);
220   const d2 = dot(minus(B, P), i);
221
222   // If both dot products have the same sign , P is outside the segment
223   if((d1 > 0 && d2 > 0) || (d1 < 0 && d2 < 0)) {
224     return 0;
225   }
226   return 1;
227 }
228 /**
229 * Returns the overlapping segment (if any) between two segments P1-P2 and
230 * P3-P4,
231 * or indicates no overlap.
232 * @param {number[]} P1
233 * @param {number[]} P2
234 * @param {number[]} P3
235 * @param {number[]} P4
236 * @returns {{ A: number[] | null , B: number[] | null , flag: number , id1: number
237   | null , id2: number | null }}}
238 */
239 linesOverlap(P1, P2, P3, P4) {
240   let A = null;
241   let B = null;
242   let id1 = null;
243   let id2 = null;
244   let flag = 0;
245
246   // i = (P2 - P1) / ||P2 - P1||
247   const v = minus(P2, P1);
248   const len = norm(v);
249   const i = scale(v, 1.0 / len);
250
251   const P = [P1, P2, P3, P4];
252
253   const tArr = [];
254   tArr.push([0, 1, 1]);
255   tArr.push([dot(minus(P2, P1), i), 2, 1]);
256   tArr.push([dot(minus(P3, P1), i), 3, 2]);
257   tArr.push([dot(minus(P4, P1), i), 4, 2]);
258
259   // Sort rows by the first column

```

```

259     tArr.sort((a, b) => a[0] - b[0]);
260
261     if (Math.abs(tArr[0][2] - tArr[1][2]) <= EPS) {
262         // no overlap
263         flag = 1;
264     } else {
265         id1 = tArr[1][1];
266         id2 = tArr[2][1];
267         A = P[id1 - 1];
268         B = P[id2 - 1];
269         flag = 0;
270     }
271
272     return { A, B, flag, id1, id2 };
273 }
274
275 /**
276 * Finds the minimal 3D distance between all pairs of points in two arrays.
277 *
278 * @param {number[][]} P1i - Array of 3D points (e.g. [[x1, y1, z1], [x2, y2,
279 * , z2], ...]).
280 * @param {number[][]} P2i - Another array of 3D points.
281 * @returns {{ L: number, P1: number[], P2: number[] }}
282 * L - The minimal distance found.
283 * P1 - The point in P1i corresponding to the minimal distance.
284 * P2 - The point in P2i corresponding to the minimal distance.
285 */
286 minPointsDistance3D(P1i, P2i) {
287     var P1ia = P1i.flat();
288     var P2ia = P2i.flat();
289
290     let L = Infinity;
291     let I = -1;
292     let J = -1;
293
294     for (let i = 0; i < P1ia.length; i += 3) {
295         for (let j = 0; j < P2ia.length; j += 3) {
296             const dx = P1ia[i] - P2ia[j];
297             const dy = P1ia[i+1] - P2ia[j+1];
298             const dz = P1ia[i+2] - P2ia[j+2];
299
300             const dist = Math.sqrt(dx*dx + dy*dy + dz*dz);
301
302             if (dist < L) {
303                 L = dist;
304                 I = i;
305                 J = j;
306             }
307         }
308     }
309     const closestP1 = [P1ia[I], P1ia[I + 1], P1ia[I + 2]];
310     const closestP2 = [P2ia[J], P2ia[J + 1], P2ia[J + 2]];
311
312     return {

```

```

313     L,
314     P1: closestP1 ,
315     P2: closestP2
316   };
317 }
318
319 /**
320 * Creates a new triangle instance.
321 * @param {Array} p1 - First vertex.
322 * @param {Array} p2 - Second vertex.
323 * @param {Array} p3 - Third vertex.
324 * @returns {PRDC_JSLAB_TRIANGLE} Triangle instance.
325 */
326 triangle(p1, p2, p3) {
327   return new PRDC_JSLAB_TRIANGLE(this.jsl, p1, p2, p3);
328 }
329
330 /**
331 * Performs Delaunay triangulation on a set of points.
332 * @param {Array[]} points - Array of points.
333 * @returns {PRDC_JSLAB_TRIANGLE[]} Array of triangles.
334 */
335 delaunayTriangulation(points) {
336   var min_x = Math.min(...points.map(p => p[0]));
337   var min_y = Math.min(...points.map(p => p[1]));
338   var max_x = Math.max(...points.map(p => p[0]));
339   var max_y = Math.max(...points.map(p => p[1]));
340
341   var dx = max_x - min_x, dy = max_y - min_y;
342   var delta_max = Math.max(dx, dy) * 2;
343   var p1 = [min_x - delta_max, min_y - delta_max, 0];
344   var p2 = [min_x + delta_max * 2, min_y - delta_max, 0];
345   var p3 = [min_x + dx / 2, min_y + delta_max * 2, 0];
346
347   var triangles = [this.triangle(p1, p2, p3)];
348
349   for(var point of points) {
350     var bad_triangles = triangles.filter(tri => tri.circumcircleContains(
351       point));
352
353     var edges = [];
354     for(var tri of bad_triangles) {
355       edges.push(...tri.edges);
356     }
357
358     var unique_edges = edges.filter((edge, index, arr) =>
359       arr.filter(e => (e[0] === edge[1] && e[1] === edge[0]) ||
360             (e[0] === edge[0] && e[1] === edge[1])).length === 1
361   );
362
363   triangles = triangles.filter(tri => !bad_triangles.includes(tri));
364
365   for(var edge of unique_edges) {
366     triangles.push(this.triangle(edge[0], edge[1], point));
367   }

```

```

367 }
368
369 var final_triangles = [];
370 for(var tri of triangles){
371   if(tri.p1 === p1 || tri.p2 === p1 || tri.p3 === p1) continue;
372   if(tri.p1 === p2 || tri.p2 === p2 || tri.p3 === p2) continue;
373   if(tri.p1 === p3 || tri.p2 === p3 || tri.p3 === p3) continue;
374   final_triangles.push(tri);
375 }
376 return final_triangles;
377 }
378
379 /**
380 * Generates a rotation matrix to rotate from vector a to vector b.
381 * @param {number[]} a - The initial unit vector.
382 * @param {number[]} b - The target unit vector.
383 * @returns {number[][]} - The resulting rotation matrix.
384 */
385 getRotationMatrix(a, b) {
386   // a and b are unit vectors
387   var v = this.jsl.array.cross3D(a, b, 1);
388   var s = this.jsl.env.math.norm(v);
389   var c = this.jsl.array.dotVector(a, b);

390   if(c === -1) { // Vectors are opposite
391     // Find a vector orthogonal to 'a'
392     var orthogonal = [0, 0, 0];
393     if(this.jsl.env.math.abs(a[0]) < this.jsl.env.math.abs(a[1]) &&
394       this.jsl.env.math.abs(a[0]) < this.jsl.env.math.abs(a[2])) {
395       orthogonal = [0, -a[2], a[1]];
396     } else if(this.jsl.env.math.abs(a[1]) < this.jsl.env.math.abs(a[2])) {
397       orthogonal = [-a[2], 0, a[0]];
398     } else {
399       orthogonal = [-a[1], a[0], 0];
400     }
401     orthogonal = this.jsl.array.normalizeVector(orthogonal);
402
403     // Compute rotation matrix using Householder reflection
404     var o = orthogonal;
405     return [
406       [-1 + 2 * o[0] * o[0], 2 * o[0] * o[1], 2 * o[0] * o[2],
407        2 * o[1] * o[0], -1 + 2 * o[1] * o[1], 2 * o[1] * o[2],
408        2 * o[2] * o[0], 2 * o[2] * o[1], -1 + 2 * o[2] * o[2]],
409     ];
410   } else if(s === 0) { // Vectors are the same
411     // Return identity matrix
412     return [1, 0, 0, 0, 1, 0, 0, 0, 1];
413   } else {
414     // Compute skew-symmetric cross-product matrix of v
415     var vx = this.jsl.array.skewVector(v);

416     // Compute R = I + vx + vx * vx * ((1 - c) / (s * s))
417     var I = [1, 0, 0, 0, 1, 0, 0, 0, 1];
418
419     var vx2 = this.jsl.array.multiply(vx, vx, 3, 3, 3);
420
421     var vx3 = this.jsl.array.multiply(vx2, vx, 3, 3, 3);
422
423     var vx4 = this.jsl.array.multiply(vx3, vx, 3, 3, 3);
424
425     var vx5 = this.jsl.array.multiply(vx4, vx, 3, 3, 3);
426
427     var vx6 = this.jsl.array.multiply(vx5, vx, 3, 3, 3);
428
429     var vx7 = this.jsl.array.multiply(vx6, vx, 3, 3, 3);
430
431     var vx8 = this.jsl.array.multiply(vx7, vx, 3, 3, 3);
432
433     var vx9 = this.jsl.array.multiply(vx8, vx, 3, 3, 3);
434
435     var vx10 = this.jsl.array.multiply(vx9, vx, 3, 3, 3);
436
437     var vx11 = this.jsl.array.multiply(vx10, vx, 3, 3, 3);
438
439     var vx12 = this.jsl.array.multiply(vx11, vx, 3, 3, 3);
440
441     var vx13 = this.jsl.array.multiply(vx12, vx, 3, 3, 3);
442
443     var vx14 = this.jsl.array.multiply(vx13, vx, 3, 3, 3);
444
445     var vx15 = this.jsl.array.multiply(vx14, vx, 3, 3, 3);
446
447     var vx16 = this.jsl.array.multiply(vx15, vx, 3, 3, 3);
448
449     var vx17 = this.jsl.array.multiply(vx16, vx, 3, 3, 3);
450
451     var vx18 = this.jsl.array.multiply(vx17, vx, 3, 3, 3);
452
453     var vx19 = this.jsl.array.multiply(vx18, vx, 3, 3, 3);
454
455     var vx20 = this.jsl.array.multiply(vx19, vx, 3, 3, 3);
456
457     var vx21 = this.jsl.array.multiply(vx20, vx, 3, 3, 3);
458
459     var vx22 = this.jsl.array.multiply(vx21, vx, 3, 3, 3);
460
461     var vx23 = this.jsl.array.multiply(vx22, vx, 3, 3, 3);
462
463     var vx24 = this.jsl.array.multiply(vx23, vx, 3, 3, 3);
464
465     var vx25 = this.jsl.array.multiply(vx24, vx, 3, 3, 3);
466
467     var vx26 = this.jsl.array.multiply(vx25, vx, 3, 3, 3);
468
469     var vx27 = this.jsl.array.multiply(vx26, vx, 3, 3, 3);
470
471     var vx28 = this.jsl.array.multiply(vx27, vx, 3, 3, 3);
472
473     var vx29 = this.jsl.array.multiply(vx28, vx, 3, 3, 3);
474
475     var vx30 = this.jsl.array.multiply(vx29, vx, 3, 3, 3);
476
477     var vx31 = this.jsl.array.multiply(vx30, vx, 3, 3, 3);
478
479     var vx32 = this.jsl.array.multiply(vx31, vx, 3, 3, 3);
480
481     var vx33 = this.jsl.array.multiply(vx32, vx, 3, 3, 3);
482
483     var vx34 = this.jsl.array.multiply(vx33, vx, 3, 3, 3);
484
485     var vx35 = this.jsl.array.multiply(vx34, vx, 3, 3, 3);
486
487     var vx36 = this.jsl.array.multiply(vx35, vx, 3, 3, 3);
488
489     var vx37 = this.jsl.array.multiply(vx36, vx, 3, 3, 3);
490
491     var vx38 = this.jsl.array.multiply(vx37, vx, 3, 3, 3);
492
493     var vx39 = this.jsl.array.multiply(vx38, vx, 3, 3, 3);
494
495     var vx40 = this.jsl.array.multiply(vx39, vx, 3, 3, 3);
496
497     var vx41 = this.jsl.array.multiply(vx40, vx, 3, 3, 3);
498
499     var vx42 = this.jsl.array.multiply(vx41, vx, 3, 3, 3);
500
501     var vx43 = this.jsl.array.multiply(vx42, vx, 3, 3, 3);
502
503     var vx44 = this.jsl.array.multiply(vx43, vx, 3, 3, 3);
504
505     var vx45 = this.jsl.array.multiply(vx44, vx, 3, 3, 3);
506
507     var vx46 = this.jsl.array.multiply(vx45, vx, 3, 3, 3);
508
509     var vx47 = this.jsl.array.multiply(vx46, vx, 3, 3, 3);
510
511     var vx48 = this.jsl.array.multiply(vx47, vx, 3, 3, 3);
512
513     var vx49 = this.jsl.array.multiply(vx48, vx, 3, 3, 3);
514
515     var vx50 = this.jsl.array.multiply(vx49, vx, 3, 3, 3);
516
517     var vx51 = this.jsl.array.multiply(vx50, vx, 3, 3, 3);
518
519     var vx52 = this.jsl.array.multiply(vx51, vx, 3, 3, 3);
520
521     var vx53 = this.jsl.array.multiply(vx52, vx, 3, 3, 3);
522
523     var vx54 = this.jsl.array.multiply(vx53, vx, 3, 3, 3);
524
525     var vx55 = this.jsl.array.multiply(vx54, vx, 3, 3, 3);
526
527     var vx56 = this.jsl.array.multiply(vx55, vx, 3, 3, 3);
528
529     var vx57 = this.jsl.array.multiply(vx56, vx, 3, 3, 3);
530
531     var vx58 = this.jsl.array.multiply(vx57, vx, 3, 3, 3);
532
533     var vx59 = this.jsl.array.multiply(vx58, vx, 3, 3, 3);
534
535     var vx60 = this.jsl.array.multiply(vx59, vx, 3, 3, 3);
536
537     var vx61 = this.jsl.array.multiply(vx60, vx, 3, 3, 3);
538
539     var vx62 = this.jsl.array.multiply(vx61, vx, 3, 3, 3);
540
541     var vx63 = this.jsl.array.multiply(vx62, vx, 3, 3, 3);
542
543     var vx64 = this.jsl.array.multiply(vx63, vx, 3, 3, 3);
544
545     var vx65 = this.jsl.array.multiply(vx64, vx, 3, 3, 3);
546
547     var vx66 = this.jsl.array.multiply(vx65, vx, 3, 3, 3);
548
549     var vx67 = this.jsl.array.multiply(vx66, vx, 3, 3, 3);
550
551     var vx68 = this.jsl.array.multiply(vx67, vx, 3, 3, 3);
552
553     var vx69 = this.jsl.array.multiply(vx68, vx, 3, 3, 3);
554
555     var vx70 = this.jsl.array.multiply(vx69, vx, 3, 3, 3);
556
557     var vx71 = this.jsl.array.multiply(vx70, vx, 3, 3, 3);
558
559     var vx72 = this.jsl.array.multiply(vx71, vx, 3, 3, 3);
560
561     var vx73 = this.jsl.array.multiply(vx72, vx, 3, 3, 3);
562
563     var vx74 = this.jsl.array.multiply(vx73, vx, 3, 3, 3);
564
565     var vx75 = this.jsl.array.multiply(vx74, vx, 3, 3, 3);
566
567     var vx76 = this.jsl.array.multiply(vx75, vx, 3, 3, 3);
568
569     var vx77 = this.jsl.array.multiply(vx76, vx, 3, 3, 3);
570
571     var vx78 = this.jsl.array.multiply(vx77, vx, 3, 3, 3);
572
573     var vx79 = this.jsl.array.multiply(vx78, vx, 3, 3, 3);
574
575     var vx80 = this.jsl.array.multiply(vx79, vx, 3, 3, 3);
576
577     var vx81 = this.jsl.array.multiply(vx80, vx, 3, 3, 3);
578
579     var vx82 = this.jsl.array.multiply(vx81, vx, 3, 3, 3);
580
581     var vx83 = this.jsl.array.multiply(vx82, vx, 3, 3, 3);
582
583     var vx84 = this.jsl.array.multiply(vx83, vx, 3, 3, 3);
584
585     var vx85 = this.jsl.array.multiply(vx84, vx, 3, 3, 3);
586
587     var vx86 = this.jsl.array.multiply(vx85, vx, 3, 3, 3);
588
589     var vx87 = this.jsl.array.multiply(vx86, vx, 3, 3, 3);
590
591     var vx88 = this.jsl.array.multiply(vx87, vx, 3, 3, 3);
592
593     var vx89 = this.jsl.array.multiply(vx88, vx, 3, 3, 3);
594
595     var vx90 = this.jsl.array.multiply(vx89, vx, 3, 3, 3);
596
597     var vx91 = this.jsl.array.multiply(vx90, vx, 3, 3, 3);
598
599     var vx92 = this.jsl.array.multiply(vx91, vx, 3, 3, 3);
600
601     var vx93 = this.jsl.array.multiply(vx92, vx, 3, 3, 3);
602
603     var vx94 = this.jsl.array.multiply(vx93, vx, 3, 3, 3);
604
605     var vx95 = this.jsl.array.multiply(vx94, vx, 3, 3, 3);
606
607     var vx96 = this.jsl.array.multiply(vx95, vx, 3, 3, 3);
608
609     var vx97 = this.jsl.array.multiply(vx96, vx, 3, 3, 3);
610
611     var vx98 = this.jsl.array.multiply(vx97, vx, 3, 3, 3);
612
613     var vx99 = this.jsl.array.multiply(vx98, vx, 3, 3, 3);
614
615     var vx100 = this.jsl.array.multiply(vx99, vx, 3, 3, 3);
616
617     var vx101 = this.jsl.array.multiply(vx100, vx, 3, 3, 3);
618
619     var vx102 = this.jsl.array.multiply(vx101, vx, 3, 3, 3);
620
621     var vx103 = this.jsl.array.multiply(vx102, vx, 3, 3, 3);
622
623     var vx104 = this.jsl.array.multiply(vx103, vx, 3, 3, 3);
624
625     var vx105 = this.jsl.array.multiply(vx104, vx, 3, 3, 3);
626
627     var vx106 = this.jsl.array.multiply(vx105, vx, 3, 3, 3);
628
629     var vx107 = this.jsl.array.multiply(vx106, vx, 3, 3, 3);
630
631     var vx108 = this.jsl.array.multiply(vx107, vx, 3, 3, 3);
632
633     var vx109 = this.jsl.array.multiply(vx108, vx, 3, 3, 3);
634
635     var vx110 = this.jsl.array.multiply(vx109, vx, 3, 3, 3);
636
637     var vx111 = this.jsl.array.multiply(vx110, vx, 3, 3, 3);
638
639     var vx112 = this.jsl.array.multiply(vx111, vx, 3, 3, 3);
640
641     var vx113 = this.jsl.array.multiply(vx112, vx, 3, 3, 3);
642
643     var vx114 = this.jsl.array.multiply(vx113, vx, 3, 3, 3);
644
645     var vx115 = this.jsl.array.multiply(vx114, vx, 3, 3, 3);
646
647     var vx116 = this.jsl.array.multiply(vx115, vx, 3, 3, 3);
648
649     var vx117 = this.jsl.array.multiply(vx116, vx, 3, 3, 3);
650
651     var vx118 = this.jsl.array.multiply(vx117, vx, 3, 3, 3);
652
653     var vx119 = this.jsl.array.multiply(vx118, vx, 3, 3, 3);
654
655     var vx120 = this.jsl.array.multiply(vx119, vx, 3, 3, 3);
656
657     var vx121 = this.jsl.array.multiply(vx120, vx, 3, 3, 3);
658
659     var vx122 = this.jsl.array.multiply(vx121, vx, 3, 3, 3);
660
661     var vx123 = this.jsl.array.multiply(vx122, vx, 3, 3, 3);
662
663     var vx124 = this.jsl.array.multiply(vx123, vx, 3, 3, 3);
664
665     var vx125 = this.jsl.array.multiply(vx124, vx, 3, 3, 3);
666
667     var vx126 = this.jsl.array.multiply(vx125, vx, 3, 3, 3);
668
669     var vx127 = this.jsl.array.multiply(vx126, vx, 3, 3, 3);
670
671     var vx128 = this.jsl.array.multiply(vx127, vx, 3, 3, 3);
672
673     var vx129 = this.jsl.array.multiply(vx128, vx, 3, 3, 3);
674
675     var vx130 = this.jsl.array.multiply(vx129, vx, 3, 3, 3);
676
677     var vx131 = this.jsl.array.multiply(vx130, vx, 3, 3, 3);
678
679     var vx132 = this.jsl.array.multiply(vx131, vx, 3, 3, 3);
680
681     var vx133 = this.jsl.array.multiply(vx132, vx, 3, 3, 3);
682
683     var vx134 = this.jsl.array.multiply(vx133, vx, 3, 3, 3);
684
685     var vx135 = this.jsl.array.multiply(vx134, vx, 3, 3, 3);
686
687     var vx136 = this.jsl.array.multiply(vx135, vx, 3, 3, 3);
688
689     var vx137 = this.jsl.array.multiply(vx136, vx, 3, 3, 3);
690
691     var vx138 = this.jsl.array.multiply(vx137, vx, 3, 3, 3);
692
693     var vx139 = this.jsl.array.multiply(vx138, vx, 3, 3, 3);
694
695     var vx140 = this.jsl.array.multiply(vx139, vx, 3, 3, 3);
696
697     var vx141 = this.jsl.array.multiply(vx140, vx, 3, 3, 3);
698
699     var vx142 = this.jsl.array.multiply(vx141, vx, 3, 3, 3);
700
701     var vx143 = this.jsl.array.multiply(vx142, vx, 3, 3, 3);
702
703     var vx144 = this.jsl.array.multiply(vx143, vx, 3, 3, 3);
704
705     var vx145 = this.jsl.array.multiply(vx144, vx, 3, 3, 3);
706
707     var vx146 = this.jsl.array.multiply(vx145, vx, 3, 3, 3);
708
709     var vx147 = this.jsl.array.multiply(vx146, vx, 3, 3, 3);
710
711     var vx148 = this.jsl.array.multiply(vx147, vx, 3, 3, 3);
712
713     var vx149 = this.jsl.array.multiply(vx148, vx, 3, 3, 3);
714
715     var vx150 = this.jsl.array.multiply(vx149, vx, 3, 3, 3);
716
717     var vx151 = this.jsl.array.multiply(vx150, vx, 3, 3, 3);
718
719     var vx152 = this.jsl.array.multiply(vx151, vx, 3, 3, 3);
720
721     var vx153 = this.jsl.array.multiply(vx152, vx, 3, 3, 3);
722
723     var vx154 = this.jsl.array.multiply(vx153, vx, 3, 3, 3);
724
725     var vx155 = this.jsl.array.multiply(vx154, vx, 3, 3, 3);
726
727     var vx156 = this.jsl.array.multiply(vx155, vx, 3, 3, 3);
728
729     var vx157 = this.jsl.array.multiply(vx156, vx, 3, 3, 3);
730
731     var vx158 = this.jsl.array.multiply(vx157, vx, 3, 3, 3);
732
733     var vx159 = this.jsl.array.multiply(vx158, vx, 3, 3, 3);
734
735     var vx160 = this.jsl.array.multiply(vx159, vx, 3, 3, 3);
736
737     var vx161 = this.jsl.array.multiply(vx160, vx, 3, 3, 3);
738
739     var vx162 = this.jsl.array.multiply(vx161, vx, 3, 3, 3);
740
741     var vx163 = this.jsl.array.multiply(vx162, vx, 3, 3, 3);
742
743     var vx164 = this.jsl.array.multiply(vx163, vx, 3, 3, 3);
744
745     var vx165 = this.jsl.array.multiply(vx164, vx, 3, 3, 3);
746
747     var vx166 = this.jsl.array.multiply(vx165, vx, 3, 3, 3);
748
749     var vx167 = this.jsl.array.multiply(vx166, vx, 3, 3, 3);
750
751     var vx168 = this.jsl.array.multiply(vx167, vx, 3, 3, 3);
752
753     var vx169 = this.jsl.array.multiply(vx168, vx, 3, 3, 3);
754
755     var vx170 = this.jsl.array.multiply(vx169, vx, 3, 3, 3);
756
757     var vx171 = this.jsl.array.multiply(vx170, vx, 3, 3, 3);
758
759     var vx172 = this.jsl.array.multiply(vx171, vx, 3, 3, 3);
760
761     var vx173 = this.jsl.array.multiply(vx172, vx, 3, 3, 3);
762
763     var vx174 = this.jsl.array.multiply(vx173, vx, 3, 3, 3);
764
765     var vx175 = this.jsl.array.multiply(vx174, vx, 3, 3, 3);
766
767     var vx176 = this.jsl.array.multiply(vx175, vx, 3, 3, 3);
768
769     var vx177 = this.jsl.array.multiply(vx176, vx, 3, 3, 3);
770
771     var vx178 = this.jsl.array.multiply(vx177, vx, 3, 3, 3);
772
773     var vx179 = this.jsl.array.multiply(vx178, vx, 3, 3, 3);
774
775     var vx180 = this.jsl.array.multiply(vx179, vx, 3, 3, 3);
776
777     var vx181 = this.jsl.array.multiply(vx180, vx, 3, 3, 3);
778
779     var vx182 = this.jsl.array.multiply(vx181, vx, 3, 3, 3);
780
781     var vx183 = this.jsl.array.multiply(vx182, vx, 3, 3, 3);
782
783     var vx184 = this.jsl.array.multiply(vx183, vx, 3, 3, 3);
784
785     var vx185 = this.jsl.array.multiply(vx184, vx, 3, 3, 3);
786
787     var vx186 = this.jsl.array.multiply(vx185, vx, 3, 3, 3);
788
789     var vx187 = this.jsl.array.multiply(vx186, vx, 3, 3, 3);
790
791     var vx188 = this.jsl.array.multiply(vx187, vx, 3, 3, 3);
792
793     var vx189 = this.jsl.array.multiply(vx188, vx, 3, 3, 3);
794
795     var vx190 = this.jsl.array.multiply(vx189, vx, 3, 3, 3);
796
797     var vx191 = this.jsl.array.multiply(vx190, vx, 3, 3, 3);
798
799     var vx192 = this.jsl.array.multiply(vx191, vx, 3, 3, 3);
800
801     var vx193 = this.jsl.array.multiply(vx192, vx, 3, 3, 3);
802
803     var vx194 = this.jsl.array.multiply(vx193, vx, 3, 3, 3);
804
805     var vx195 = this.jsl.array.multiply(vx194, vx, 3, 3, 3);
806
807     var vx196 = this.jsl.array.multiply(vx195, vx, 3, 3, 3);
808
809     var vx197 = this.jsl.array.multiply(vx196, vx, 3, 3, 3);
810
811     var vx198 = this.jsl.array.multiply(vx197, vx, 3, 3, 3);
812
813     var vx199 = this.jsl.array.multiply(vx198, vx, 3, 3, 3);
814
815     var vx200 = this.jsl.array.multiply(vx199, vx, 3, 3, 3);
816
817     var vx201 = this.jsl.array.multiply(vx200, vx, 3, 3, 3);
818
819     var vx202 = this.jsl.array.multiply(vx201, vx, 3, 3, 3);
820
821     var vx203 = this.jsl.array.multiply(vx202, vx, 3, 3, 3);
822
823     var vx204 = this.jsl.array.multiply(vx203, vx, 3, 3, 3);
824
825     var vx205 = this.jsl.array.multiply(vx204, vx, 3, 3, 3);
826
827     var vx206 = this.jsl.array.multiply(vx205, vx, 3, 3, 3);
828
829     var vx207 = this.jsl.array.multiply(vx206, vx, 3, 3, 3);
830
831     var vx208 = this.jsl.array.multiply(vx207, vx, 3, 3, 3);
832
833     var vx209 = this.jsl.array.multiply(vx208, vx, 3, 3, 3);
834
835     var vx210 = this.jsl.array.multiply(vx209, vx, 3, 3, 3);
836
837     var vx211 = this.jsl.array.multiply(vx210, vx, 3, 3, 3);
838
839     var vx212 = this.jsl.array.multiply(vx211, vx, 3, 3, 3);
840
841     var vx213 = this.jsl.array.multiply(vx212, vx, 3, 3, 3);
842
843     var vx214 = this.jsl.array.multiply(vx213, vx, 3, 3, 3);
844
845     var vx215 = this.jsl.array.multiply(vx214, vx, 3, 3, 3);
846
847     var vx216 = this.jsl.array.multiply(vx215, vx, 3, 3, 3);
848
849     var vx217 = this.jsl.array.multiply(vx216, vx, 3, 3, 3);
850
851     var vx218 = this.jsl.array.multiply(vx217, vx, 3, 3, 3);
852
853     var vx219 = this.jsl.array.multiply(vx218, vx, 3, 3, 3);
854
855     var vx220 = this.jsl.array.multiply(vx219, vx, 3, 3, 3);
856
857     var vx221 = this.jsl.array.multiply(vx220, vx, 3, 3, 3);
858
859     var vx222 = this.jsl.array.multiply(vx221, vx, 3, 3, 3);
860
861     var vx223 = this.jsl.array.multiply(vx222, vx, 3, 3, 3);
862
863     var vx224 = this.jsl.array.multiply(vx223, vx, 3, 3, 3);
864
865     var vx225 = this.jsl.array.multiply(vx224, vx, 3, 3, 3);
866
867     var vx226 = this.jsl.array.multiply(vx225, vx, 3, 3, 3);
868
869     var vx227 = this.jsl.array.multiply(vx226, vx, 3, 3, 3);
870
871     var vx228 = this.jsl.array.multiply(vx227, vx, 3, 3, 3);
872
873     var vx229 = this.jsl.array.multiply(vx228, vx, 3, 3, 3);
874
875     var vx230 = this.jsl.array.multiply(vx229, vx, 3, 3, 3);
876
877     var vx231 = this.jsl.array.multiply(vx230, vx, 3, 3, 3);
878
879     var vx232 = this.jsl.array.multiply(vx231, vx, 3, 3, 3);
880
881     var vx233 = this.jsl.array.multiply(vx232, vx, 3, 3, 3);
882
883     var vx234 = this.jsl.array.multiply(vx233, vx, 3, 3, 3);
884
885     var vx235 = this.jsl.array.multiply(vx234, vx, 3, 3, 3);
886
887     var vx236 = this.jsl.array.multiply(vx235, vx, 3, 3, 3);
888
889     var vx237 = this.jsl.array.multiply(vx236, vx, 3, 3, 3);
890
891     var vx238 = this.jsl.array.multiply(vx237, vx, 3, 3, 3);
892
893     var vx239 = this.jsl.array.multiply(vx238, vx, 3, 3, 3);
894
895     var vx240 = this.jsl.array.multiply(vx239, vx, 3, 3, 3);
896
897     var vx241 = this.jsl.array.multiply(vx240, vx, 3, 3, 3);
898
899     var vx242 = this.jsl.array.multiply(vx241, vx, 3, 3, 3);
900
901     var vx243 = this.jsl.array.multiply(vx242, vx, 3, 3, 3);
902
903     var vx244 = this.jsl.array.multiply(vx243, vx, 3, 3, 3);
904
905     var vx245 = this.jsl.array.multiply(vx244, vx, 3, 3, 3);
906
907     var vx246 = this.jsl.array.multiply(vx245, vx, 3, 3, 3);
908
909     var vx247 = this.jsl.array.multiply(vx246, vx, 3, 3, 3);
910
911     var vx248 = this.jsl.array.multiply(vx247, vx, 3, 3, 3);
912
913     var vx249 = this.jsl.array.multiply(vx248, vx, 3, 3, 3);
914
915     var vx250 = this.jsl.array.multiply(vx249, vx, 3, 3, 3);
916
917     var vx251 = this.jsl.array.multiply(vx250, vx, 3, 3, 3);
918
919     var vx252 = this.jsl.array.multiply(vx251, vx, 3, 3, 3);
920
921     var vx253 = this.jsl.array.multiply(vx252, vx, 3, 3, 3);
922
923     var vx254 = this.jsl.array.multiply(vx253, vx, 3, 3, 3);
924
925     var vx255 = this.jsl.array.multiply(vx254, vx, 3, 3, 3);
926
927     var vx256 = this.jsl.array.multiply(vx255, vx, 3, 3, 3);
928
929     var vx257 = this.jsl.array.multiply(vx256, vx, 3, 3, 3);
930
931     var vx258 = this.jsl.array.multiply(vx257, vx, 3, 3, 3);
932
933     var vx259 = this.jsl.array.multiply(vx258, vx, 3, 3, 3);
934
935     var vx260 = this.jsl.array.multiply(vx259, vx, 3, 3, 3);
936
937     var vx261 = this.jsl.array.multiply(vx260, vx, 3, 3, 3);
938
939     var vx262 = this.jsl.array.multiply(vx261, vx, 3, 3, 3);
940
941     var vx263 = this.jsl.array.multiply(vx262, vx, 3, 3, 3);
942
943     var vx264 = this.jsl.array.multiply(vx263, vx, 3, 3, 3);
944
945     var vx265 = this.jsl.array.multiply(vx264, vx, 3, 3, 3);
946
947     var vx266 = this.jsl.array.multiply(vx265, vx, 3, 3, 3);
948
949     var vx267 = this.jsl.array.multiply(vx266, vx, 3, 3, 3);
950
951     var vx268 = this.jsl.array.multiply(vx267, vx, 3, 3, 3);
952
953     var vx269 = this.jsl.array.multiply(vx268, vx, 3, 3, 3);
954
955     var vx270 = this.jsl.array.multiply(vx269, vx, 3, 3, 3);
956
957     var vx271 = this.jsl.array.multiply(vx270, vx, 3, 3, 3);
958
959     var vx272 = this.jsl.array.multiply(vx271, vx, 3, 3, 3);
960
961     var vx273 = this.jsl.array.multiply(vx272, vx, 3, 3, 3);
962
963     var vx274 = this.jsl.array.multiply(vx273, vx, 3, 3, 3);
964
965     var vx275 = this.jsl.array.multiply(vx274, vx, 3, 3, 3);
966
967     var vx276 = this.jsl.array.multiply(vx275, vx, 3, 3, 3);
968
969     var vx277 = this.jsl.array.multiply(vx276, vx, 3, 3, 3);
970
971     var vx278 = this.jsl.array.multiply(vx277, vx, 3, 3, 3);
972
973     var vx279 = this.jsl.array.multiply(vx278, vx, 3, 3, 3);
974
975     var vx280 = this.jsl.array.multiply(vx279, vx, 3, 3, 3);
976
977     var vx281 = this.jsl.array.multiply(vx280, vx, 3, 3, 3);
978
979     var vx282 = this.jsl.array.multiply(vx281, vx, 3, 3, 3);
980
981     var vx283 = this.jsl.array.multiply(vx282, vx, 3, 3, 3);
982
983     var vx284 = this.jsl.array.multiply(vx283, vx, 3, 3, 3);
984
985     var vx285 = this.jsl.array.multiply(vx284, vx, 3, 3, 3);
986
987     var vx286 = this.jsl.array.multiply(vx285, vx, 3, 3, 3);
988
989     var vx287 = this.jsl.array.multiply(vx286, vx, 3, 3, 3);
990
991     var vx288 = this.jsl.array.multiply(vx287, vx, 3, 3, 3);
992
9
```

```

422
423     var factor = (1 - c) / (s * s);
424
425     var R = this.jsl.array.plus(this.jsl.array.plus(I, vx), this.jsl.array.
426         scale(vx2, factor));
427
428     return R;
429 }
430
431 /**
432 * Transforms coordinates by scaling, rotating, and translating them.
433 * @param {number[][]} coordinates - Array of coordinate points.
434 * @param {number} scale_factor - Factor by which to scale the coordinates.
435 * @param {number[][]} rotation_matrix - Matrix used to rotate the
436   coordinates.
437 * @param {number[]} translation - Vector used to translate the coordinates.
438 * @returns {number[][]} - The transformed coordinates.
439 */
440 transform(coordinates, scale_factor, rotation_matrix, translation) {
441   var obj = this;
442   return coordinates.map(function(coordinate) {
443     var transformed = obj.jsl.array.plus(translation, obj.jsl.array.multiply
444       (rotation_matrix, obj.jsl.array.scale(coordinate, scale_factor), 3,
445        3, 1));
446     return transformed;
447   });
448 }
449 /**
450 * Creates 3D vectors for plotting based on provided parameters.
451 * @param {number[]} xi - X coordinates of vector origins.
452 * @param {number[]} yi - Y coordinates of vector origins.
453 * @param {number[]} zi - Z coordinates of vector origins.
454 * @param {number[]} ui - X components of vectors.
455 * @param {number[]} vi - Y components of vectors.
456 * @param {number[]} wi - Z components of vectors.
457 * @param {number} scale - Scale factor for the vectors.
458 * @param {number} angle_factor - Angle factor for arrowheads.
459 * @param {Object} opts - Additional plotting options.
460 * @returns {Object} - An object containing line and head trace data for
461   plotting.
462 */
463 createVectors3D(xi, yi, zi, ui, vi, wi, scale = 0.3, angle_factor = 0.4,
464   opts) {
465   if(!Array.isArray(xi)) xi = [xi];
466   if(!Array.isArray(yi)) yi = [yi];
467   if(!Array.isArray(zi)) zi = [zi];
468   if(!Array.isArray(ui)) ui = [ui];
469   if(!Array.isArray(vi)) vi = [vi];
470   if(!Array.isArray(wi)) wi = [wi];
471
472   // Define the unit arrow once
473   var arrowhead_length = scale * 1;
474   var arrowhead_width = arrowhead_length * this.jsl.env.math.tan(

```

```

angle_factor);

471
472 // Shaft points (unit arrow along x-axis)
473 var shaft_start = [0, 0, 0];
474 var shaft_end = [1 - arrowhead_length, 0, 0];
475
476 // Arrowhead base points
477 var arrow_tip = [1, 0, 0]; // Tip at the end of the shaft
478 var arrow_left = [1 - arrowhead_length, arrowhead_width, 0];
479 var arrow_right = [1 - arrowhead_length, -arrowhead_width, 0];
480
481 // All points of the unit arrow
482 var unit_arrow_points = [shaft_start, shaft_end, arrow_tip, arrow_left,
483   arrow_right];
484
485 var vectors = {
486   line: {
487     x: [],
488     y: [],
489     z: [],
490     type: 'scatter3d', color: '#00f', mode: 'lines', showLegend: false
491   },
492   head: {
493     x: [],
494     y: [],
495     z: [],
496     i: [],
497     j: [],
498     k: [],
499     type: 'mesh3d', color: '#00f', opacity: 1, flatShading: true,
500     showScale: false, showLegend: false, lighting: {ambient: 1}
501   }
502 };
503
504 if(typeof opts === 'object') {
505   Object.assign(vectors.line, opts);
506   Object.assign(vectors.head, opts);
507   if(opts.hasOwnProperty('id')) {
508     vectors.line.id = opts.id + '-line';
509     vectors.head.id = opts.id + '-head';
510   }
511 }
512
513 var vertex_index = 0;
514 var x_axis = [1, 0, 0];
515
516 for(var i = 0; i < xi.length; i++) {
517   var x0 = xi[i];
518   var y0 = yi[i];
519   var z0 = zi[i];
520   var u = ui[i];
521   var v = vi[i];
522   var w = wi[i];
523
524   var vector = [u, v, w];

```

```

524
525 // Calculate the length of the vector
526 var length = this.jsl.env.math.norm(vector);
527 if(length === 0) continue; // Skip zero-length vectors
528
529 // Normalize the direction vector
530 var dir = this.jsl.array.normalizeVector(vector);
531
532 // Compute rotation matrix to rotate from x-axis to dir
533 var R = this.getRotationMatrix(x_axis, dir);
534
535 // Scale factor is the length of the vector
536 var scale_factor = length;
537
538 // Translation vector is the starting point (x0, y0, z0)
539 var translation = [x0, y0, z0];
540
541 // Transform the unit arrow points
542 var transformed_points = this.transform(unit_arrow_points, scale_factor,
543 R, translation);
544
545 // Extract points
546 var shaft_start_rot = transformed_points[0];
547 var shaft_end_rot = transformed_points[1];
548 var arrow_tip_rot = transformed_points[2];
549 var arrow_left_rot = transformed_points[3];
550 var arrow_right_rot = transformed_points[4];
551
552 // Add shaft line to lines arrays
553 vectors.line.x.push(shaft_start_rot[0], shaft_end_rot[0], null);
554 vectors.line.y.push(shaft_start_rot[1], shaft_end_rot[1], null);
555 vectors.line.z.push(shaft_start_rot[2], shaft_end_rot[2], null);
556
557 // Add arrowhead edges to lines arrays
558 vectors.line.x.push(
559   arrow_tip_rot[0], arrow_left_rot[0], null,
560   arrow_left_rot[0], arrow_right_rot[0], null,
561   arrow_right_rot[0], arrow_tip_rot[0], null
562 );
563 vectors.line.y.push(
564   arrow_tip_rot[1], arrow_left_rot[1], null,
565   arrow_left_rot[1], arrow_right_rot[1], null,
566   arrow_right_rot[1], arrow_tip_rot[1], null
567 );
568 vectors.line.z.push(
569   arrow_tip_rot[2], arrow_left_rot[2], null,
570   arrow_left_rot[2], arrow_right_rot[2], null,
571   arrow_right_rot[2], arrow_tip_rot[2], null
572 );
573
574 // Add arrowhead vertices to mesh arrays
575 vectors.head.x.push(arrow_tip_rot[0], arrow_left_rot[0], arrow_right_rot
576 [0]);
577 vectors.head.y.push(arrow_tip_rot[1], arrow_left_rot[1], arrow_right_rot
578 [1]);

```

```

576     vectors.head.z.push(arrow_tip_rot[2], arrow_left_rot[2], arrow_right_rot
577     [2]);
578
579     // Define the face for the current arrowhead
580     vectors.head.i.push(vertex_index);
581     vectors.head.j.push(vertex_index + 1);
582     vectors.head.k.push(vertex_index + 2);
583
584     // Update vertex index for the next arrow
585     vertex_index += 3;
586   }
587   return vectors;
588 }
589 /**
590 * Creates 3D disks for plotting based on provided parameters.
591 * @param {number[]} xi - X coordinates of disk centers.
592 * @param {number[]} yi - Y coordinates of disk centers.
593 * @param {number[]} zi - Z coordinates of disk centers.
594 * @param {number[]} ri - Radii of the disks.
595 * @param {number[]} ui - X components of normal vectors (for disk
596 orientation).
597 * @param {number[]} vi - Y components of normal vectors (for disk
598 orientation).
599 * @param {number[]} wi - Z components of normal vectors (for disk
600 orientation).
601 * @param {Object} opts - Additional plotting options.
602 * @returns {Object} - An object containing line and area trace data for
603 plotting.
604 */
605 createDisks3D(xi, yi, zi, ri, ui, vi, wi, opts = {}) {
606   if(!Array.isArray(xi)) xi = [xi];
607   if(!Array.isArray(yi)) yi = [yi];
608   if(!Array.isArray(zi)) zi = [zi];
609   if(!Array.isArray(ri)) ri = [ri];
610   if(!Array.isArray(ui)) ui = [ui];
611   if(!Array.isArray(vi)) vi = [vi];
612   if(!Array.isArray(wi)) wi = [wi];
613   if(ri.length === 1 && xi.length > 1) ri = new Array(xi.length).fill(ri[0])
614   ;
615
616   var segments = opts.segments || 32;
617   delete opts.segments;
618
619   var disks = {
620     line: {
621       x: [],
622       y: [],
623       z: [],
624       type: 'scatter3d', color: '#00f', mode: 'lines', showLegend: false
625     },
626     area: {
627       x: [],
628       y: [],
629       z: []
630     }
631   }
632
633   for(let i = 0; i < xi.length; i++) {
634     let r = ri[i];
635     let u = ui[i];
636     let v = vi[i];
637     let w = wi[i];
638
639     let center = [xi[i], yi[i], zi[i]];
640
641     let segmentsAngle = 2 * Math.PI / segments;
642     let angleStep = segmentsAngle / 2;
643
644     let startAngle = Math.atan2(v, u);
645     let endAngle = startAngle + segmentsAngle;
646
647     let start = [center[0] + r * Math.cos(startAngle), center[1] + r * Math.sin(startAngle), center[2]];
648     let end = [center[0] + r * Math.cos(endAngle), center[1] + r * Math.sin(endAngle), center[2]];
649
650     let areaStart = [center[0] + r * Math.cos(startAngle + angleStep), center[1] + r * Math.sin(startAngle + angleStep), center[2]];
651     let areaEnd = [center[0] + r * Math.cos(endAngle - angleStep), center[1] + r * Math.sin(endAngle - angleStep), center[2]];
652
653     disks.line.x.push(...[start, end].map(p => p[0]));
654     disks.line.y.push(...[start, end].map(p => p[1]));
655     disks.line.z.push(...[start, end].map(p => p[2]));
656
657     disks.area.x.push(...[areaStart, areaEnd].map(p => p[0]));
658     disks.area.y.push(...[areaStart, areaEnd].map(p => p[1]));
659     disks.area.z.push(...[areaStart, areaEnd].map(p => p[2]));
660   }
661
662   return disks;
663 }

```

```

625     i: [],
626     j: [],
627     k: [],
628     type: 'mesh3d', color: '#00f', opacity: 1, flatShading: true,
629     showScale: false, showLegend: false, lighting: {ambient: 1}
630   }
631 };
632
633 if(typeof opts == 'object') {
634   Object.assign(disks.line, opts);
635   Object.assign(disks.area, opts);
636   if(opts.hasOwnProperty('id')) {
637     disks.line.id = opts.id + '-line';
638     disks.area.id = opts.id + '-area';
639   }
640 }
641
642 var vertex_index = 0;
643 var z_axis = [0, 0, 1]; // Assuming disk normal initially aligns with x-
644   axis
645
646 // Create a unit circle in the XY plane for the disk
647 var points = [...this.circle(1, segments)];
648
649 for(var i = 0; i < xi.length; i++) {
650   var x0 = xi[i];
651   var y0 = yi[i];
652   var z0 = zi[i];
653   var radius = ri[i];
654   var normal = [ui[i], vi[i], wi[i]];
655
656   if(radius === 0) continue; // Skip zero-radius disks
657
658   // Normalize the normal vector to ensure it's a unit vector
659   var dir = this.jsl.array.normalizeVector(normal);
660
661   // Compute rotation matrix to rotate from x-axis to the normal vector
662   var R = this.getRotationMatrix(z_axis, dir);
663
664   // Translation vector is the disk center point
665   var translation = [x0, y0, z0];
666
667   var circle = [...points];
668
669   // Transform the circle points to the correct position and orientation
670   var transformed_points = this.transform(circle, radius, R, translation);
671
672   // Add circle outline to line data for edges
673   for(var j = 0; j < transformed_points.length; j++) {
674     var point = transformed_points[j];
675     disks.line.x.push(point[0]);
676     disks.line.y.push(point[1]);
677     disks.line.z.push(point[2]);
678     if(j == transformed_points.length - 1) {
679       disks.line.x.push(transformed_points[0][0], null); // Connect last

```

```

          to first point
679   disks.line.y.push(transformed_points[0][1], null);
680   disks.line.z.push(transformed_points[0][2], null);
681 }
682 }
683 disks.line.x.push(null);
684 disks.line.y.push(null);
685 disks.line.z.push(null);

686 // Add all points for mesh (area) data
687 for(var j = 0; j < transformed_points.length; j++) {
688   var point = transformed_points[j];
689   disks.area.x.push(point[0]);
690   disks.area.y.push(point[1]);
691   disks.area.z.push(point[2]);
692 }
693

694 // Triangulation for mesh
695 for(var j = 1; j < transformed_points.length - 1; j++) {
696   disks.area.i.push(vertex_index);
697   disks.area.j.push(vertex_index + j);
698   disks.area.k.push(vertex_index + j + 1);
699 }
700

701 vertex_index += transformed_points.length;
702 }
703 return disks;
704 }

705 /**
706 * Creates a rectangular planes in 3D space, oriented by a normal vector [u,
707 * v, w].
708 * @param {number[]} xi - X coordinates of planes centers.
709 * @param {number[]} yi - Y coordinates of planes centers.
710 * @param {number[]} zi - Z coordinates of planes centers.
711 * @param {number[]} width_i - Width of the rectangle.
712 * @param {number[]} height_i - Height of the rectangle.
713 * @param {number[]} ui - X component of the plane's normal vector.
714 * @param {number[]} vi - Y component of the plane's normal vector.
715 * @param {number[]} wi - Z component of the plane's normal vector.
716 * @param {Object} opts - Additional plotting options (color, opacity,
717 * etc.).
718 * @returns {Object} - An object containing line and area trace data
719 * for plotting.
720 */
721 createPlanes3D(xi, yi, zi, width_i, height_i, ui, vi, wi, opts) {
722   if(!Array.isArray(xi)) xi = [xi];
723   if(!Array.isArray(yi)) yi = [yi];
724   if(!Array.isArray(zi)) zi = [zi];
725   if(!Array.isArray(width_i)) width_i = [width_i];
726   if(!Array.isArray(height_i)) height_i = [height_i];
727   if(!Array.isArray(ui)) ui = [ui];
728   if(!Array.isArray(vi)) vi = [vi];
729   if(!Array.isArray(wi)) wi = [wi];
730   if(width_i.length === 1 && xi.length > 1) width_i = new Array(xi.length).fill(width_i[0]);
    
```

```

730   fill(width_i[0]);
731   if(height_i.length === 1 && xi.length > 1) height_i = new Array(xi.length)
732     .fill(height_i[0]);
733
734   const planes = {
735     line: {
736       x: [],
737       y: [],
738       z: [],
739       type: 'scatter3d',
740       color: '#00f',
741       mode: 'lines',
742       showLegend: false
743     },
744     area: {
745       x: [],
746       y: [],
747       z: [],
748       i: [],
749       j: [],
750       k: [],
751       type: 'mesh3d',
752       color: '#00f',
753       opacity: 1,
754       flatShading: true,
755       showScale: false,
756       showLegend: false,
757       lighting: { ambient: 1 }
758     }
759   };
760
761   // Merge any provided opts into our line & area objects
762   if(typeof opts === 'object') {
763     Object.assign(planes.line, opts);
764     Object.assign(planes.area, opts);
765     if(opts.hasOwnProperty('id')) {
766       planes.line.id = opts.id + '-line';
767       planes.area.id = opts.id + '-area';
768     }
769   }
770
771   // Reference axis (z-axis) that our rectangle initially lies in (XY-plane)
772
773   // We will rotate from z-axis to our normal.
774   const z_axis = [0, 0, 1];
775
776   for(var i = 0; i < ui.length; i++) {
777     // Create a symmetrical rectangle in the XY-plane, centered at (0,0,0)
778     const rect_coords = this.symRectangle(width_i[i], height_i[i], 0);
779
780     // Convert that flat array into point-triplets:
781     var rectangle_points = [];
782     for(let i = 0; i < rect_coords.length; i += 3) {
783       rectangle_points.push([rect_coords[i], rect_coords[i + 1], rect_coords[i + 2]]);
    
```

```

781 }
782
783 // Normal vector
784 const normal = [ ui[ i ] , vi[ i ] , wi[ i ] ];
785
786 // Normalize the plane normal
787 const dir = this.jsl.array.normalizeVector(normal);
788
789 // Compute rotation matrix to rotate a plane (lying in XY-plane) so its
790 // normal aligns with 'dir'
791 const R = this.getRotationMatrix(z_axis , dir);
792
793 // Rotate these points according to R
794 // No scaling or translation is applied here. If you want to shift it to
795 // [x0,y0,z0], just add that translation.
796 rectangle_points = this.transform(rectangle_points , 1.0 , R , [ xi[ i ] , yi[ i ]
797 // Build the line trace: push each consecutive segment plus a null to
798 // break the stroke
799 for(let j = 0; j < rectangle_points.length; j++) {
800     var [x, y, z] = rectangle_points[j];
801     planes.line.x.push(x);
802     planes.line.y.push(y);
803     planes.line.z.push(z);
804 }
805 // Insert null to break the line
806 planes.line.x.push(null);
807 planes.line.y.push(null);
808 planes.line.z.push(null);
809
810 // Build the mesh: we only need the first 4 unique corners for a
811 // rectangle
812 // (the 5th is a repeat of the 1st).
813 // Triangulate the rectangle as two triangles: (0,1,2) and (0,2,3)
814 const n = 4; // We only take indices 0..3
815 const base_index = 0;
816
817 for(let j = 0; j < n; j++) {
818     planes.area.x.push(rectangle_points[j][0]);
819     planes.area.y.push(rectangle_points[j][1]);
820     planes.area.z.push(rectangle_points[j][2]);
821 }
822 // Two triangles to form the quad
823 planes.area.i.push(base_index , base_index);
824 planes.area.j.push(base_index + 1 , base_index + 2);
825 planes.area.k.push(base_index + 2 , base_index + 3);
826 }
827
828 /**
829 * Creates a lines in 3D space.
830 * @param {number[]} x1i - X1 coordinates of lines.
831 * @param {number[]} y1i - Y1 coordinates of lines.

```

```

831  * @param {number[]} z1i - Z1 coordinates of lines.
832  * @param {number[]} x2i - X2 coordinates of lines.
833  * @param {number[]} y2i - Y2 coordinates of lines.
834  * @param {number[]} z2i - Z2 coordinates of lines.
835  * @returns {Object} - lines object.
836  */
837 createLines3D(x1i, y1i, z1i, x2i, y2i, z2i, opts) {
838   if(!Array.isArray(x1i)) x1i = [x1i];
839   if(!Array.isArray(y1i)) y1i = [y1i];
840   if(!Array.isArray(z1i)) z1i = [z1i];
841   if(!Array.isArray(x2i)) x2i = [x2i];
842   if(!Array.isArray(y2i)) y2i = [y2i];
843   if(!Array.isArray(z2i)) z2i = [z2i];
844
845   const lines = {
846     x: [],
847     y: [],
848     z: [],
849     type: 'scatter3d',
850     color: '#00f',
851     mode: 'lines',
852     showLegend: false
853   };
854
855   // Merge any provided opts into our line & area objects
856   if(typeof opts === 'object') {
857     Object.assign(lines, opts);
858   }
859
860   for(var i = 0; i < x1i.length; i++) {
861     lines.x.push(x1i[i], x2i[i], null);
862     lines.y.push(y1i[i], y2i[i], null);
863     lines.z.push(z1i[i], z2i[i], null);
864   }
865   return lines;
866 }
867 /**
868  * Creates a points in 3D space.
869  * @param {number[]} xi - X coordinates of points.
870  * @param {number[]} yi - Y coordinates of points.
871  * @param {number[]} zi - Z coordinates of points.
872  * @returns {Object} - points object.
873  */
874 createPoints3D(xi, yi, zi, opts) {
875   if(!Array.isArray(xi)) xi = [xi];
876   if(!Array.isArray(yi)) yi = [yi];
877   if(!Array.isArray(zi)) zi = [zi];
878
879   const points = {
880     x: [],
881     y: [],
882     z: [],
883     type: 'scatter3d',
884     color: '#00f',
885   };

```

```
886     mode: 'markers',
887     showLegend: false
888   };
889
890   // Merge any provided opts into our line & area objects
891   if(typeof opts === 'object') {
892     Object.assign(points, opts);
893   }
894
895   for(var i = 0; i < xi.length; i++) {
896     points.x.push(xi[i], null);
897     points.y.push(yi[i], null);
898     points.z.push zi[i], null);
899   }
900   return points;
901 }
902
903 /**
904 * Creates a points in 3D space.
905 * @param {number[]} xi - X coordinates of points.
906 * @param {number[]} yi - Y coordinates of points.
907 * @param {number[]} zi - Z coordinates of points.
908 * @returns {Object} - points object.
909 */
910 createText3D(xi, yi, zi, texti, dxi, dyi, dzi, opts) {
911   if(!Array.isArray(xi)) xi = [xi];
912   if(!Array.isArray(yi)) yi = [yi];
913   if(!Array.isArray(zi)) zi = [zi];
914   if(!Array.isArray(texti)) texti = [texti];
915   if(!Array.isArray(dxi)) dxi = [dxi];
916   if(!Array.isArray(dyi)) dyi = [dyi];
917   if(!Array.isArray(dzi)) dzi = [dzi];
918   if(dxi.length === 1 && xi.length > 1) dxi = new Array(xi.length).fill(dxi[0]);
919   if(dyi.length === 1 && yi.length > 1) dyi = new Array(yi.length).fill(dyi[0]);
920   if(dzi.length === 1 && zi.length > 1) dzi = new Array(zi.length).fill(dzi[0]);
921
922   const texts = {
923     x: [],
924     y: [],
925     z: [],
926     text: [],
927     type: 'scatter3d',
928     textposition: 'center middle',
929     textfont: {
930       size: 18,
931       color: '#f00',
932     },
933     mode: 'text',
934     showLegend: false
935   };
936
937   // Merge any provided opts into our line & area objects
```

```

938     if(typeof opts === 'object') {
939         Object.assign(texts, opts);
940     }
941
942     for(var i = 0; i < xi.length; i++) {
943         texts.x.push(plus(xi[i], dxi[i]));
944         texts.y.push(plus(yi[i], dyi[i]));
945         texts.z.push(plus zi[i], dzi[i]));
946         texts.text.push(texti[i]);
947     }
948     return texts;
949 }
950
951 /**
952 * Creates a symmetrical rectangle in 3D space.
953 * @param {number} W - Width of the rectangle.
954 * @param {number} H - Height of the rectangle.
955 * @param {number} [Z=0] - Z-coordinate for the rectangle plane.
956 * @returns {number[]} - Array of vertex coordinates for the rectangle.
957 */
958 symRectangle(W, H, Z = 0) {
959     return [W/2, H/2, Z,
960             -W/2, H/2, Z,
961             -W/2, -H/2, Z,
962             W/2, -H/2, Z,
963             W/2, H/2, Z];
964 }
965
966 /**
967 * Helper method to generate points for a circle in the XY plane.
968 * @param {number} radius - Radius of the circle.
969 * @param {number} segments - Number of segments for the circle.
970 * @returns {number[][]} - Array of points forming the circle.
971 */
972 circle(radius, segments) {
973     var points = [];
974     for(var i = 0; i < segments; i++) {
975         var angle = 2 * Math.PI * i / segments;
976         points.push([radius * Math.cos(angle), radius * Math.sin(angle), 0]);
977     }
978     return points;
979 }
980
981 /**
982 * Helper method to generate points for a disk in the XY plane.
983 * @param {number} radius - Radius of the disk.
984 * @param {number} segments_a - Number of angular segments for the disk.
985 * @param {number} segments_r - Number of radial segments for the disk.
986 * @returns {number[][]} - Array of points forming the disk.
987 */
988 disk(radius, segments_a, segments_r) {
989     var points = [];
990     points.push([0, 0, 0]);
991     for(var j = 1; j <= segments_r; j++) {
992         var r = radius * j / segments_r;

```

```

993     for (var i = 1; i <= segments_a; i++) {
994         var angle = 2 * Math.PI * i / segments_a;
995         points.push([r * Math.cos(angle), r * Math.sin(angle), 0]);
996     }
997 }
998 return points;
999 }

1000 /**
1001 * Generates the boundary of a 3D shape based on points and a shrink factor.
1002 * @param {number[][]} points - Array of points defining the shape.
1003 * @param {number} [shrink=0.5] - Factor by which to shrink the boundary.
1004 * @returns {Array} - An array containing boundary facets and the volume.
1005 */
1006 boundary3D(points, shrink = 0.5) {
1007     var shp = new this.jsl.env.AlphaShape3D();
1008     shp.newShape(points);

1009     var Acrit = shp.getCriticalAlpha('one-region');
1010     var spec = shp.getAlphaSpectrum();

1011     var idx = spec.indexOf(Acrit);
1012     var subspec = spec.slice(idx);

1013     var idx = Math.max(Math.ceil((1 - shrink) * subspec.length) - 1, 0);
1014     var alphaval = subspec[idx];

1015     shp.setAlpha(alphaval);
1016     var V = shp.getVolume();
1017     var bf = shp.getBoundaryFacets();
1018     shp = null;
1019     return [bf, V];
1020 }

1021 /**
1022 * Writes geometry data to an OFF file.
1023 * @param {string} filename - The path to the OFF file.
1024 * @param {number[][]} vertices - Array of vertex coordinates.
1025 * @param {number[][]} faces - Array of face indices.
1026 */
1027 writeOff(filename, vertices, faces) {
1028     var shp = new this.jsl.env.AlphaShape3D();
1029     shp.writeOff(filename, vertices, faces);
1030     shp = null;
1031 }

1032 /**
1033 * Reads an OFF file and returns the vertices and faces.
1034 * @param {string} filename - The path to the OFF file.
1035 * @returns {{vertices: number[][], faces: number[][]}} - An object
1036             containing vertices and faces arrays.
1037 * @throws Will throw an error if the file cannot be read or is not a valid
1038             OFF file.
1039 */
1040 readOff(filename) {
1041
1042
1043
1044
1045

```

```

1046 var data = this.jsl.env.readFileSync(filename, 'utf8');
1047 var tokens = data.split(/\s+/).filter(token => token.length > 0);
1048 if(tokens[0] !== 'OFF') {
1049   this.jsl.env.error('@readOff: '+language.string(193));
1050 }
1051 tokens.shift();
1052 if(tokens.length < 3) {
1053   this.jsl.env.error('@readOff: '+language.string(194));
1054 }
1055
1056 var j = 0;
1057 tokens = tokens.map(x => parseFloat(x));
1058
1059 var nvert = tokens[j++];
1060 var nface = tokens[j++];
1061 if(!isNaN(tokens[j])) {
1062   j++;
1063 }
1064
1065 // Read vertex coordinates
1066 var vertices = createArray(nvert);
1067 var k = 0;
1068 for(var i = 0; i < nvert; i++) {
1069   if(j + 2 >= tokens.length) {
1070     this.jsl.env.error('@readOff: '+language.string(195));
1071     break;
1072   }
1073   vertices[k++] = [tokens[j++], tokens[j++], tokens[j++]];
1074 }
1075
1076 // Read face data
1077 var faces = createArray(nface);
1078 var k = 0;
1079 for(let i = 0; i < nface; i++) {
1080   if(j >= tokens.length) {
1081     this.jsl.env.error('@readOff: '+language.string(196));
1082     break;
1083   }
1084   var vertices_per_face = tokens[j++];
1085   var face = [];
1086   for(let v = 0; v < vertices_per_face; v++) {
1087     face.push(tokens[j++]);
1088   }
1089   faces[k++] = face;
1090 }
1091
1092 return [vertices, faces];
1093 }
1094 }
1095
1096 exports.PRDC_JSLAB_LIB_GEOMETRY = PRDC_JSLAB_LIB_GEOMETRY;
1097
1098 /**
1099 * Class for JSLAB triangle.
1100 */

```

```

1101 class PRDC_JSLAB_TRIANGLE {
1102
1103 #jsl;
1104
1105 /**
1106 * Creates a new triangle instance.
1107 * @constructor
1108 * @param {Object} jsl - Reference to the main JSLAB object.
1109 * @param {Array} p1 - First vertex.
1110 * @param {Array} p2 - Second vertex.
1111 * @param {Array} p3 - Third vertex.
1112 */
1113 constructor(jsl, v1, v2, v3) {
1114   this.#jsl = jsl;
1115   this._set(v1, v2, v3);
1116 }
1117
1118 /**
1119 * Sets the triangle vertices and edges.
1120 * @param {Array} v1 - First vertex.
1121 * @param {Array} v2 - Second vertex.
1122 * @param {Array} v3 - Third vertex.
1123 */
1124 _set(v1, v2, v3) {
1125   if(Array.isArray(v1[0])) {
1126     v3 = v1[2];
1127     v2 = v1[1];
1128     v1 = v1[0];
1129   }
1130
1131   this.v1 = v1 || [];
1132   this.v2 = v2 || [];
1133   this.v3 = v3 || [];
1134   this.edges = [
1135     [v1, v2], [v2, v3], [v3, v1]
1136   ];
1137 }
1138
1139 /**
1140 * Checks if the triangle's circumcircle contains a point.
1141 * @param {Array} point - The point to test.
1142 * @returns {boolean} True if inside the circumcircle.
1143 */
1144 circumcircleContains(point) {
1145   var ax = this.v1[0], ay = this.v1[1];
1146   var bx = this.v2[0], by = this.v2[1];
1147   var cx = this.v3[0], cy = this.v3[1];
1148   var dx = point[0], dy = point[1];
1149
1150   var ax_ = ax - dx, ay_ = ay - dy;
1151   var bx_ = bx - dx, by_ = by - dy;
1152   var cx_ = cx - dx, cy_ = cy - dy;
1153
1154   var det = (ax_ * ax_ + ay_ * ay_) * (bx_ * cy_ - cx_ * by_) -
1155           (bx_ * bx_ + by_ * by_) * (ax_ * cy_ - cx_ * ay_);

```

```

1156           (cx_ * cx_ + cy_ * cy_) * (ax_ * by_ - bx_ * ay_);
1157
1158     return det > 0;
1159   }
1160
1161 /**
1162 * Determines if a point is inside the triangle.
1163 * @param {Array} point - The point to check.
1164 * @returns {boolean} True if the point is inside.
1165 */
1166 contains(point) {
1167   var sign = (v1, v2, v3) =>
1168     (v1[0] - v3[0]) * (v2[1] - v3[1]) - (v2[0] - v3[0]) * (v1[1] - v3[1]);
1169
1170   var d1 = sign(point, this.v1, this.v2);
1171   var d2 = sign(point, this.v2, this.v3);
1172   var d3 = sign(point, this.v3, this.v1);
1173
1174   return (d1 >= 0 && d2 >= 0 && d3 >= 0)
1175     || (d1 <= 0 && d2 <= 0 && d3 <= 0);
1176 }
1177
1178 /**
1179 * Computes the interpolated value at a point using barycentric coordinates.
1180 * @param {Array} point - The query point.
1181 * @returns {number} Interpolated value.
1182 */
1183 valueAt(point) {
1184   const denom = ((this.v2[1] - this.v3[1]) * (this.v1[0] - this.v3[0]) +
1185                 (this.v3[0] - this.v2[0]) * (this.v1[1] - this.v3[1]));
1186
1187   // Compute barycentric coordinates
1188   const alpha = ((this.v2[1] - this.v3[1]) * (point[0] - this.v3[0]) +
1189                  (this.v3[0] - this.v2[0]) * (point[1] - this.v3[1])) /
1190     denom;
1191   const beta = ((this.v3[1] - this.v1[1]) * (point[0] - this.v3[0]) +
1192                 (this.v1[0] - this.v3[0]) * (point[1] - this.v3[1])) / denom
1193     ;
1194   const gamma = 1 - alpha - beta;
1195
1196   return alpha * this.v1[2] + beta * this.v2[2] + gamma * this.v3[2];
1197 }
1198 }
```

Listing 99 - geometry.js

```

1 /**
2 * @file JSLAB library gui submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB gui submodule.
10 */

```

```

11 class PRDC_JSLAB_LIB_GUI {
12
13     /**
14      * Initializes the gui submodule.
15      * @param {Object} js1 Reference to the main JSLAB object.
16      */
17     constructor(js1) {
18         var obj = this;
19         this.js1 = js1;
20
21         this.escape_html_map = {
22             '&': '&amp;',
23             '<': '&lt;',
24             '>': '&gt;',
25             '"' : '&quot;',
26             "''": '&#39;',
27             '/' : '&#x2F;',
28             '' '': '&#x60;',
29             '=' : '&#x3D;';
30         };
31     }
32
33     /**
34      * Escapes reserved HTML characters in a string to prevent injection or XSS.
35      * @param {string} string The raw text whose special characters should be
36      * converted to HTML entities.
37      */
38     escapeHTML(string) {
39         return String(string).replace(/[&<>"'=\/]/g, s => this.escape_html_map[s
40             ]);
41     }
42
43     /**
44      * Reports whether an element participates in layout flow (i.e., is not
45      * 'display:none').
46      * @param {HTMLElement|null} el The element to test for visibility in normal
47      * document flow.
48      * @returns {boolean} 'true' if the element has a non-null 'offsetParent';
49      * otherwise 'false'.
50      */
51     isVisible(el) {
52         return !!el && el.offsetParent !== null;
53     }
54
55     /**
56      * Reads an input-like elements value, escapes it for HTML, and returns
57      * the result.
58      * @param {string|HTMLElement} el_or_sel The element itself or a selector/ID
59      * string used to locate it.
60      * @returns {string} The escaped string value, or an empty string if the
61      * element is not found.
62      */
63     getElVal(el_or_sel) {
64         return this.escapeHTML(this._readVal(this._resolveEl(el_or_sel)));
65     }

```

```

58     }
59
60    /**
61     * Reads an input-like elements value, converts it to a number, and
62     * returns the result.
63     * @param {string|HTMLElement} el_or_sel The element itself or a selector/ID
64     * string used to locate it.
65     * @returns {number} The numeric value or 'NaN' if conversion fails.
66     */
67   getElValNum(el_or_sel) {
68     return Number(this._readVal(this._resolveEl(el_or_sel)));
69   }
70
71   /**
72     * Attaches 'input' and 'change' listeners to a slider and forwards events
73     * to a callback.
74     * @param {string|HTMLElement} el The slider element or a selector for
75     * delegated listening.
76     * @param {Function} fun The function to invoke each time the slider emits 'input'
77     * or 'change'.
78     */
79   onSliderInput(el, fun) {
80     var handler = function (e) { fun.call(this, e); };
81
82     if(typeof el === 'string') {
83       var doc = this._resolveDoc(el);
84       this._delegate(doc, 'input', el, () => true, handler);
85       this._delegate(doc, 'change', el, () => true, handler);
86     } else {
87       var n = el;
88       n.addEventListener('input', handler);
89       n.addEventListener('change', handler);
90     }
91   }
92
93   /**
94     * Fires a callback when an input loses focus or the user presses Enter,
95     * with optional delegation.
96     * @param {string|HTMLElement} el Target element or CSS selector for
97     * delegation.
98     * @param {Function} fun Callback executed with 'this' bound to the element
99     * that triggered the event.
100    * @param {Function} [validator] Optional predicate that must return 'true'
101      for the callback to fire.
102    */
103   onInput(el, fun, validator) {
104     if(typeof el === 'string') {
105       var doc = this._resolveDoc(el);
106       this._delegate(doc, 'focusout', el, () => true, fun, true);
107       this._delegate(doc, 'keyup', el, e => e.keyCode === this._ENTER, fun);
108     } else {
109       var nodes = this._resolveEls(el);
110       nodes.forEach(n => {
111         n.addEventListener('blur', e => fun.call(n, e));
112         n.addEventListener('keyup', e => {
113           if(e.keyCode === this._ENTER) fun.call(n, e);
114         });
115       });
116     }
117   }
118
119   /**
120     * Returns the first element matching the selector or the element itself if
121     * no selector was provided.
122     * @param {string|HTMLElement} el The element or selector to resolve.
123     * @returns {Element} The resolved element.
124     */
125   _resolveEl(el) {
126     if(typeof el === 'string') {
127       return document.querySelector(el);
128     } else {
129       return el;
130     }
131   }
132
133   /**
134     * Returns the first element matching the selector or the element itself if
135     * no selector was provided.
136     * @param {string|HTMLElement} el The element or selector to resolve.
137     * @returns {Element} The resolved element.
138     */
139   _resolveEls(el) {
140     if(typeof el === 'string') {
141       return document.querySelectorAll(el);
142     } else {
143       return el;
144     }
145   }
146
147   /**
148     * Converts a string to a number, returning NaN if the conversion fails.
149     * @param {string} str The string to convert.
150     * @returns {number} The converted number or NaN.
151     */
152   _readVal(str) {
153     return Number(str);
154   }
155
156   /**
157     * Resolves a selector or element to a Document object.
158     * @param {string|HTMLElement} el The selector or element to resolve.
159     * @returns {Document} The resolved document.
160     */
161   _resolveDoc(el) {
162     if(typeof el === 'string') {
163       return document;
164     } else {
165       return el.ownerDocument || el;
166     }
167   }
168
169   /**
170     * Delegates an event listener to a target element or document.
171     * @param {Document|Element} doc The target element or document.
172     * @param {string} type The event type.
173     * @param {Element} el The element to delegate to.
174     * @param {Function} fun The callback function.
175     * @param {boolean} capture Whether the event should be captured.
176     */
177   _delegate(doc, type, el, fun, capture) {
178     doc.addEventListener(type, fun, capture);
179     el.addEventListener(type, fun, capture);
180   }
181
182   /**
183     * Checks if a predicate returns true for a given argument.
184     * @param {Function} pred The predicate function.
185     * @param {*} arg The argument to check.
186     * @returns {boolean} Whether the predicate returned true.
187     */
188   _isTrue(pred, arg) {
189     return pred(arg);
190   }
191
192   /**
193     * Checks if a predicate returns true for all arguments.
194     * @param {Function} pred The predicate function.
195     * @param {...*} args The arguments to check.
196     * @returns {boolean} Whether the predicate returned true for all arguments.
197     */
198   _allTrue(pred, ...args) {
199     return args.every(arg => pred(arg));
200   }
201
202   /**
203     * Checks if a predicate returns true for any argument.
204     * @param {Function} pred The predicate function.
205     * @param {...*} args The arguments to check.
206     * @returns {boolean} Whether the predicate returned true for any argument.
207     */
208   _anyTrue(pred, ...args) {
209     return args.some(arg => pred(arg));
210   }
211
212   /**
213     * Checks if a predicate returns true for the first argument.
214     * @param {Function} pred The predicate function.
215     * @param {*} arg The argument to check.
216     * @returns {boolean} Whether the predicate returned true for the first argument.
217     */
218   _firstTrue(pred, arg) {
219     return pred(arg);
220   }
221
222   /**
223     * Checks if a predicate returns true for the last argument.
224     * @param {Function} pred The predicate function.
225     * @param {*} arg The argument to check.
226     * @returns {boolean} Whether the predicate returned true for the last argument.
227     */
228   _lastTrue(pred, arg) {
229     return pred(arg);
230   }
231
232   /**
233     * Checks if a predicate returns true for the middle argument.
234     * @param {Function} pred The predicate function.
235     * @param {*} arg The argument to check.
236     * @returns {boolean} Whether the predicate returned true for the middle argument.
237     */
238   _middleTrue(pred, arg) {
239     return pred(arg);
240   }
241
242   /**
243     * Checks if a predicate returns true for the second argument.
244     * @param {Function} pred The predicate function.
245     * @param {*} arg The argument to check.
246     * @returns {boolean} Whether the predicate returned true for the second argument.
247     */
248   _secondTrue(pred, arg) {
249     return pred(arg);
250   }
251
252   /**
253     * Checks if a predicate returns true for the third argument.
254     * @param {Function} pred The predicate function.
255     * @param {*} arg The argument to check.
256     * @returns {boolean} Whether the predicate returned true for the third argument.
257     */
258   _thirdTrue(pred, arg) {
259     return pred(arg);
260   }
261
262   /**
263     * Checks if a predicate returns true for the fourth argument.
264     * @param {Function} pred The predicate function.
265     * @param {*} arg The argument to check.
266     * @returns {boolean} Whether the predicate returned true for the fourth argument.
267     */
268   _fourthTrue(pred, arg) {
269     return pred(arg);
270   }
271
272   /**
273     * Checks if a predicate returns true for the fifth argument.
274     * @param {Function} pred The predicate function.
275     * @param {*} arg The argument to check.
276     * @returns {boolean} Whether the predicate returned true for the fifth argument.
277     */
278   _fifthTrue(pred, arg) {
279     return pred(arg);
280   }
281
282   /**
283     * Checks if a predicate returns true for the sixth argument.
284     * @param {Function} pred The predicate function.
285     * @param {*} arg The argument to check.
286     * @returns {boolean} Whether the predicate returned true for the sixth argument.
287     */
288   _sixthTrue(pred, arg) {
289     return pred(arg);
290   }
291
292   /**
293     * Checks if a predicate returns true for the seventh argument.
294     * @param {Function} pred The predicate function.
295     * @param {*} arg The argument to check.
296     * @returns {boolean} Whether the predicate returned true for the seventh argument.
297     */
298   _seventhTrue(pred, arg) {
299     return pred(arg);
300   }
301
302   /**
303     * Checks if a predicate returns true for the eighth argument.
304     * @param {Function} pred The predicate function.
305     * @param {*} arg The argument to check.
306     * @returns {boolean} Whether the predicate returned true for the eighth argument.
307     */
308   _eighthTrue(pred, arg) {
309     return pred(arg);
310   }
311
312   /**
313     * Checks if a predicate returns true for the ninth argument.
314     * @param {Function} pred The predicate function.
315     * @param {*} arg The argument to check.
316     * @returns {boolean} Whether the predicate returned true for the ninth argument.
317     */
318   _ninthTrue(pred, arg) {
319     return pred(arg);
320   }
321
322   /**
323     * Checks if a predicate returns true for the tenth argument.
324     * @param {Function} pred The predicate function.
325     * @param {*} arg The argument to check.
326     * @returns {boolean} Whether the predicate returned true for the tenth argument.
327     */
328   _tenthTrue(pred, arg) {
329     return pred(arg);
330   }
331
332   /**
333     * Checks if a predicate returns true for the eleventh argument.
334     * @param {Function} pred The predicate function.
335     * @param {*} arg The argument to check.
336     * @returns {boolean} Whether the predicate returned true for the eleventh argument.
337     */
338   _eleventhTrue(pred, arg) {
339     return pred(arg);
340   }
341
342   /**
343     * Checks if a predicate returns true for the twelfth argument.
344     * @param {Function} pred The predicate function.
345     * @param {*} arg The argument to check.
346     * @returns {boolean} Whether the predicate returned true for the twelfth argument.
347     */
348   _twelfthTrue(pred, arg) {
349     return pred(arg);
350   }
351
352   /**
353     * Checks if a predicate returns true for the thirteenth argument.
354     * @param {Function} pred The predicate function.
355     * @param {*} arg The argument to check.
356     * @returns {boolean} Whether the predicate returned true for the thirteenth argument.
357     */
358   _thirteenthTrue(pred, arg) {
359     return pred(arg);
360   }
361
362   /**
363     * Checks if a predicate returns true for the fourteenth argument.
364     * @param {Function} pred The predicate function.
365     * @param {*} arg The argument to check.
366     * @returns {boolean} Whether the predicate returned true for the fourteenth argument.
367     */
368   _fourteenthTrue(pred, arg) {
369     return pred(arg);
370   }
371
372   /**
373     * Checks if a predicate returns true for the fifteenth argument.
374     * @param {Function} pred The predicate function.
375     * @param {*} arg The argument to check.
376     * @returns {boolean} Whether the predicate returned true for the fifteenth argument.
377     */
378   _fifteenthTrue(pred, arg) {
379     return pred(arg);
380   }
381
382   /**
383     * Checks if a predicate returns true for the sixteenth argument.
384     * @param {Function} pred The predicate function.
385     * @param {*} arg The argument to check.
386     * @returns {boolean} Whether the predicate returned true for the sixteenth argument.
387     */
388   _sixteenthTrue(pred, arg) {
389     return pred(arg);
390   }
391
392   /**
393     * Checks if a predicate returns true for the seventeenth argument.
394     * @param {Function} pred The predicate function.
395     * @param {*} arg The argument to check.
396     * @returns {boolean} Whether the predicate returned true for the seventeenth argument.
397     */
398   _seventeenthTrue(pred, arg) {
399     return pred(arg);
400   }
401
402   /**
403     * Checks if a predicate returns true for the eighteenth argument.
404     * @param {Function} pred The predicate function.
405     * @param {*} arg The argument to check.
406     * @returns {boolean} Whether the predicate returned true for the eighteenth argument.
407     */
408   _eighteenthTrue(pred, arg) {
409     return pred(arg);
410   }
411
412   /**
413     * Checks if a predicate returns true for the nineteenth argument.
414     * @param {Function} pred The predicate function.
415     * @param {*} arg The argument to check.
416     * @returns {boolean} Whether the predicate returned true for the nineteenth argument.
417     */
418   _nineteenthTrue(pred, arg) {
419     return pred(arg);
420   }
421
422   /**
423     * Checks if a predicate returns true for the twentieth argument.
424     * @param {Function} pred The predicate function.
425     * @param {*} arg The argument to check.
426     * @returns {boolean} Whether the predicate returned true for the twentieth argument.
427     */
428   _twentiethTrue(pred, arg) {
429     return pred(arg);
430   }
431
432   /**
433     * Checks if a predicate returns true for the twenty-first argument.
434     * @param {Function} pred The predicate function.
435     * @param {*} arg The argument to check.
436     * @returns {boolean} Whether the predicate returned true for the twenty-first argument.
437     */
438   _twentyFirstTrue(pred, arg) {
439     return pred(arg);
440   }
441
442   /**
443     * Checks if a predicate returns true for the twenty-second argument.
444     * @param {Function} pred The predicate function.
445     * @param {*} arg The argument to check.
446     * @returns {boolean} Whether the predicate returned true for the twenty-second argument.
447     */
448   _twentySecondTrue(pred, arg) {
449     return pred(arg);
450   }
451
452   /**
453     * Checks if a predicate returns true for the twenty-third argument.
454     * @param {Function} pred The predicate function.
455     * @param {*} arg The argument to check.
456     * @returns {boolean} Whether the predicate returned true for the twenty-third argument.
457     */
458   _twentyThirdTrue(pred, arg) {
459     return pred(arg);
460   }
461
462   /**
463     * Checks if a predicate returns true for the twenty-fourth argument.
464     * @param {Function} pred The predicate function.
465     * @param {*} arg The argument to check.
466     * @returns {boolean} Whether the predicate returned true for the twenty-fourth argument.
467     */
468   _twentyFourthTrue(pred, arg) {
469     return pred(arg);
470   }
471
472   /**
473     * Checks if a predicate returns true for the twenty-fifth argument.
474     * @param {Function} pred The predicate function.
475     * @param {*} arg The argument to check.
476     * @returns {boolean} Whether the predicate returned true for the twenty-fifth argument.
477     */
478   _twentyFifthTrue(pred, arg) {
479     return pred(arg);
480   }
481
482   /**
483     * Checks if a predicate returns true for the twenty-sixth argument.
484     * @param {Function} pred The predicate function.
485     * @param {*} arg The argument to check.
486     * @returns {boolean} Whether the predicate returned true for the twenty-sixth argument.
487     */
488   _twentySixthTrue(pred, arg) {
489     return pred(arg);
490   }
491
492   /**
493     * Checks if a predicate returns true for the twenty-seventh argument.
494     * @param {Function} pred The predicate function.
495     * @param {*} arg The argument to check.
496     * @returns {boolean} Whether the predicate returned true for the twenty-seventh argument.
497     */
498   _twentySeventhTrue(pred, arg) {
499     return pred(arg);
500   }
501
502   /**
503     * Checks if a predicate returns true for the twenty-eighth argument.
504     * @param {Function} pred The predicate function.
505     * @param {*} arg The argument to check.
506     * @returns {boolean} Whether the predicate returned true for the twenty-eighth argument.
507     */
508   _twentyEighthTrue(pred, arg) {
509     return pred(arg);
510   }
511
512   /**
513     * Checks if a predicate returns true for the twenty-ninth argument.
514     * @param {Function} pred The predicate function.
515     * @param {*} arg The argument to check.
516     * @returns {boolean} Whether the predicate returned true for the twenty-ninth argument.
517     */
518   _twentyNinthTrue(pred, arg) {
519     return pred(arg);
520   }
521
522   /**
523     * Checks if a predicate returns true for the thirty argument.
524     * @param {Function} pred The predicate function.
525     * @param {*} arg The argument to check.
526     * @returns {boolean} Whether the predicate returned true for the thirty argument.
527     */
528   _thirtyTrue(pred, arg) {
529     return pred(arg);
530   }
531
532   /**
533     * Checks if a predicate returns true for the thirty-first argument.
534     * @param {Function} pred The predicate function.
535     * @param {*} arg The argument to check.
536     * @returns {boolean} Whether the predicate returned true for the thirty-first argument.
537     */
538   _thirtyFirstTrue(pred, arg) {
539     return pred(arg);
540   }
541
542   /**
543     * Checks if a predicate returns true for the thirty-second argument.
544     * @param {Function} pred The predicate function.
545     * @param {*} arg The argument to check.
546     * @returns {boolean} Whether the predicate returned true for the thirty-second argument.
547     */
548   _thirtySecondTrue(pred, arg) {
549     return pred(arg);
550   }
551
552   /**
553     * Checks if a predicate returns true for the thirty-third argument.
554     * @param {Function} pred The predicate function.
555     * @param {*} arg The argument to check.
556     * @returns {boolean} Whether the predicate returned true for the thirty-third argument.
557     */
558   _thirtyThirdTrue(pred, arg) {
559     return pred(arg);
560   }
561
562   /**
563     * Checks if a predicate returns true for the thirty-fourth argument.
564     * @param {Function} pred The predicate function.
565     * @param {*} arg The argument to check.
566     * @returns {boolean} Whether the predicate returned true for the thirty-fourth argument.
567     */
568   _thirtyFourthTrue(pred, arg) {
569     return pred(arg);
570   }
571
572   /**
573     * Checks if a predicate returns true for the thirty-fifth argument.
574     * @param {Function} pred The predicate function.
575     * @param {*} arg The argument to check.
576     * @returns {boolean} Whether the predicate returned true for the thirty-fifth argument.
577     */
578   _thirtyFifthTrue(pred, arg) {
579     return pred(arg);
580   }
581
582   /**
583     * Checks if a predicate returns true for the thirty-sixth argument.
584     * @param {Function} pred The predicate function.
585     * @param {*} arg The argument to check.
586     * @returns {boolean} Whether the predicate returned true for the thirty-sixth argument.
587     */
588   _thirtySixthTrue(pred, arg) {
589     return pred(arg);
590   }
591
592   /**
593     * Checks if a predicate returns true for the thirty-seventh argument.
594     * @param {Function} pred The predicate function.
595     * @param {*} arg The argument to check.
596     * @returns {boolean} Whether the predicate returned true for the thirty-seventh argument.
597     */
598   _thirtySeventhTrue(pred, arg) {
599     return pred(arg);
600   }
601
602   /**
603     * Checks if a predicate returns true for the thirty-eighth argument.
604     * @param {Function} pred The predicate function.
605     * @param {*} arg The argument to check.
606     * @returns {boolean} Whether the predicate returned true for the thirty-eighth argument.
607     */
608   _thirtyEighthTrue(pred, arg) {
609     return pred(arg);
610   }
611
612   /**
613     * Checks if a predicate returns true for the thirty-ninth argument.
614     * @param {Function} pred The predicate function.
615     * @param {*} arg The argument to check.
616     * @returns {boolean} Whether the predicate returned true for the thirty-ninth argument.
617     */
618   _thirtyNinthTrue(pred, arg) {
619     return pred(arg);
620   }
621
622   /**
623     * Checks if a predicate returns true for the forty argument.
624     * @param {Function} pred The predicate function.
625     * @param {*} arg The argument to check.
626     * @returns {boolean} Whether the predicate returned true for the forty argument.
627     */
628   _fortyTrue(pred, arg) {
629     return pred(arg);
630   }
631
632   /**
633     * Checks if a predicate returns true for the forty-first argument.
634     * @param {Function} pred The predicate function.
635     * @param {*} arg The argument to check.
636     * @returns {boolean} Whether the predicate returned true for the forty-first argument.
637     */
638   _fortyFirstTrue(pred, arg) {
639     return pred(arg);
640   }
641
642   /**
643     * Checks if a predicate returns true for the forty-second argument.
644     * @param {Function} pred The predicate function.
645     * @param {*} arg The argument to check.
646     * @returns {boolean} Whether the predicate returned true for the forty-second argument.
647     */
648   _fortySecondTrue(pred, arg) {
649     return pred(arg);
650   }
651
652   /**
653     * Checks if a predicate returns true for the forty-third argument.
654     * @param {Function} pred The predicate function.
655     * @param {*} arg The argument to check.
656     * @returns {boolean} Whether the predicate returned true for the forty-third argument.
657     */
658   _fortyThirdTrue(pred, arg) {
659     return pred(arg);
660   }
661
662   /**
663     * Checks if a predicate returns true for the forty-fourth argument.
664     * @param {Function} pred The predicate function.
665     * @param {*} arg The argument to check.
666     * @returns {boolean} Whether the predicate returned true for the forty-fourth argument.
667     */
668   _fortyFourthTrue(pred, arg) {
669     return pred(arg);
670   }
671
672   /**
673     * Checks if a predicate returns true for the forty-fifth argument.
674     * @param {Function} pred The predicate function.
675     * @param {*} arg The argument to check.
676     * @returns {boolean} Whether the predicate returned true for the forty-fifth argument.
677     */
678   _fortyFifthTrue(pred, arg) {
679     return pred(arg);
680   }
681
682   /**
683     * Checks if a predicate returns true for the forty-sixth argument.
684     * @param {Function} pred The predicate function.
685     * @param {*} arg The argument to check.
686     * @returns {boolean} Whether the predicate returned true for the forty-sixth argument.
687     */
688   _fortySixthTrue(pred, arg) {
689     return pred(arg);
690   }
691
692   /**
693     * Checks if a predicate returns true for the forty-seventh argument.
694     * @param {Function} pred The predicate function.
695     * @param {*} arg The argument to check.
696     * @returns {boolean} Whether the predicate returned true for the forty-seventh argument.
697     */
698   _fortySeventhTrue(pred, arg) {
699     return pred(arg);
700   }
701
702   /**
703     * Checks if a predicate returns true for the forty-eighth argument.
704     * @param {Function} pred The predicate function.
705     * @param {*} arg The argument to check.
706     * @returns {boolean} Whether the predicate returned true for the forty-eighth argument.
707     */
708   _fortyEighthTrue(pred, arg) {
709     return pred(arg);
710   }
711
712   /**
713     * Checks if a predicate returns true for the forty-ninth argument.
714     * @param {Function} pred The predicate function.
715     * @param {*} arg The argument to check.
716     * @returns {boolean} Whether the predicate returned true for the forty-ninth argument.
717     */
718   _fortyNinthTrue(pred, arg) {
719     return pred(arg);
720   }
721
722   /**
723     * Checks if a predicate returns true for the fifty argument.
724     * @param {Function} pred The predicate function.
725     * @param {*} arg The argument to check.
726     * @returns {boolean} Whether the predicate returned true for the fifty argument.
727     */
728   _fiftyTrue(pred, arg) {
729     return pred(arg);
730   }
731
732   /**
733     * Checks if a predicate returns true for the fifty-first argument.
734     * @param {Function} pred The predicate function.
735     * @param {*} arg The argument to check.
736     * @returns {boolean} Whether the predicate returned true for the fifty-first argument.
737     */
738   _fiftyFirstTrue(pred, arg) {
739     return pred(arg);
740   }
741
742   /**
743     * Checks if a predicate returns true for the fifty-second argument.
744     * @param {Function} pred The predicate function.
745     * @param {*} arg The argument to check.
746     * @returns {boolean} Whether the predicate returned true for the fifty-second argument.
747     */
748   _fiftySecondTrue(pred, arg) {
749     return pred(arg);
750   }
751
752   /**
753     * Checks if a predicate returns true for the fifty-third argument.
754     * @param {Function} pred The predicate function.
755     * @param {*} arg The argument to check.
756     * @returns {boolean} Whether the predicate returned true for the fifty-third argument.
757     */
758   _fiftyThirdTrue(pred, arg) {
759     return pred(arg);
760   }
761
762   /**
763     * Checks if a predicate returns true for the fifty-fourth argument.
764     * @param {Function} pred The predicate function.
765     * @param {*} arg The argument to check.
766     * @returns {boolean} Whether the predicate returned true for the fifty-fourth argument.
767     */
768   _fiftyFourthTrue(pred, arg) {
769     return pred(arg);
770   }
771
772   /**
773     * Checks if a predicate returns true for the fifty-fifth argument.
774     * @param {Function} pred The predicate function.
775     * @param {*} arg The argument to check.
776     * @returns {boolean} Whether the predicate returned true for the fifty-fifth argument.
777     */
778   _fiftyFifthTrue(pred, arg) {
779     return pred(arg);
780   }
781
782   /**
783     * Checks if a predicate returns true for the fifty-sixth argument.
784     * @param {Function} pred The predicate function.
785     * @param {*} arg The argument to check.
786     * @returns {boolean} Whether the predicate returned true for the fifty-sixth argument.
787     */
788   _fiftySixthTrue(pred, arg) {
789     return pred(arg);
790   }
791
792   /**
793     * Checks if a predicate returns true for the fifty-seventh argument.
794     * @param {Function} pred The predicate function.
795     * @param {*} arg The argument to check.
796     * @returns {boolean} Whether the predicate returned true for the fifty-seventh argument.
797     */
798   _fiftySeventhTrue(pred, arg) {
799     return pred(arg);
800   }
801
802   /**
803     * Checks if a predicate returns true for the fifty-eighth argument.
804     * @param {Function} pred The predicate function.
805     * @param {*} arg The argument to check.
806     * @returns {boolean} Whether the predicate returned true for the fifty-eighth argument.
807     */
808   _fiftyEighthTrue(pred, arg) {
809     return pred(arg);
810   }
811
812   /**
813     * Checks if a predicate returns true for the fifty-ninth argument.
814     * @param {Function} pred The predicate function.
815     * @param {*} arg The argument to check.
816     * @returns {boolean} Whether the predicate returned true for the fifty-ninth argument.
817     */
818   _fiftyNinthTrue(pred, arg) {
819     return pred(arg);
820   }
821
822   /**
823     * Checks if a predicate returns true for the sixty argument.
824     * @param {Function} pred The predicate function.
825     * @param {*} arg The argument to check.
826     * @returns {boolean} Whether the predicate returned true for the sixty argument.
827     */
828   _sixtyTrue(pred, arg) {
829     return pred(arg);
830   }
831
832   /**
833     * Checks if a predicate returns true for the sixty-first argument.
834     * @param {Function} pred The predicate function.
835     * @param {*} arg The argument to check.
836     * @returns {boolean} Whether the predicate returned true for the sixty-first argument.
837     */
838   _sixtyFirstTrue(pred, arg) {
839     return pred(arg);
840   }
841
842   /**
843     * Checks if a predicate returns true for the sixty-second argument.
844     * @param {Function} pred The predicate function.
845     * @param {*} arg The argument to check.
846     * @returns {boolean} Whether the predicate returned true for the sixty-second argument.
847     */
848   _sixtySecondTrue(pred, arg) {
849     return pred(arg);
850   }
851
852   /**
853     * Checks if a predicate returns true for the sixty-third argument.
854     * @param {Function} pred The predicate function.
855     * @param {*} arg The argument to check.
856     * @returns {boolean} Whether the predicate returned true for the sixty-third argument.
857     */
858   _sixtyThirdTrue(pred, arg) {
859     return pred(arg);
860   }
861
862   /**
863     * Checks if a predicate returns true for the sixty-fourth argument.
864     * @param {Function} pred The predicate function.
865     * @param {*} arg The argument to check.
866     * @returns {boolean} Whether the predicate returned true for the sixty-fourth argument.
867     */
868   _sixtyFourthTrue(pred, arg) {
869     return pred(arg);
870   }
871
872   /**
873     * Checks if a predicate returns true for the sixty-fifth argument.
874     * @param {Function} pred The predicate function.
875     * @param {*} arg The argument to check.
876     * @returns {boolean} Whether the predicate returned true for the sixty-fifth argument.
877     */
878   _sixtyFifthTrue(pred, arg) {
879     return pred(arg);
880   }
881
882   /**
883     * Checks if a predicate returns true for the sixty-sixth argument.
884     * @param {Function} pred The predicate function.
885     * @param {*} arg The argument to check.
886     * @returns {boolean} Whether the predicate returned true for the sixty-sixth argument.
887     */
888   _sixtySixthTrue(pred, arg) {
889     return pred(arg);
890   }
891
892   /**
893     * Checks if a predicate returns true for the sixty-seventh argument.
894     * @param {Function} pred The predicate function.
895     * @param {*} arg The argument to check.
896     * @returns {boolean} Whether the predicate returned true for the sixty-seventh argument.
897     */
898   _sixtySeventhTrue(pred, arg) {
899     return pred(arg);
900   }
901
902   /**
903     * Checks if a predicate returns true for the sixty-eighth argument.
904     * @param {Function} pred The predicate function.
905     * @param {*} arg The argument to check.
906     * @returns {boolean} Whether the predicate returned true for the sixty-eighth argument.
907     */
908   _sixtyEighthTrue(pred, arg) {
909     return pred(arg);
910   }
911
912   /**
913     * Checks if a predicate returns true for the sixty-ninth argument.
914     * @param {Function} pred The predicate function.
915     * @param {*} arg The argument to check.
916     * @returns {boolean} Whether the predicate returned true for the sixty-ninth argument.
917     */
918   _sixtyNinthTrue(pred, arg) {
919     return pred(arg);
920   }
921
922   /**
923     * Checks if a predicate returns true for the seventy argument.
924     * @param {Function} pred The predicate function.
925     * @param {*} arg The argument to check.
926     * @returns {boolean} Whether the predicate returned true for the seventy argument.
927     */
928   _seventyTrue(pred, arg) {
929     return pred(arg);
930   }
931
932   /**
933     * Checks if a predicate returns true for the seventy-first argument.
934     * @param {Function} pred The predicate function.
935     * @param {*} arg The argument to check.
936     * @returns {boolean} Whether the predicate returned true for the seventy-first argument.
937     */
938   _seventyFirstTrue(pred, arg) {
939     return pred(arg);
940   }
941
942   /**
943     * Checks if a predicate returns true for the seventy-second argument.
944     * @param {Function} pred The predicate function.
945     * @param {*} arg The argument to check.
946     * @returns {boolean} Whether the predicate returned true for the seventy-second argument.
947     */
948   _seventySecondTrue(pred, arg) {
949     return pred(arg);
950   }
951
952   /**
953     * Checks if a predicate returns true for the seventy-third argument.
954     * @param {Function} pred The predicate function.
955     * @param {*} arg The argument to check.
956     * @returns {boolean} Whether the predicate returned true for the seventy-third argument.
957     */
958   _seventyThirdTrue(pred, arg) {
959     return pred(arg);
960   }
961
962   /**
963     * Checks if a predicate returns true for the seventy-fourth argument.
964     * @param {Function} pred The predicate function.
965     * @param {*} arg The argument to check.
966     * @returns {boolean} Whether the predicate returned true for the seventy-fourth argument.
967     */
968   _seventyFourthTrue(pred, arg) {
969     return pred(arg);
970   }
971
972   /**
973     * Checks if a predicate returns true for the seventy-fifth argument.
974     * @param {Function} pred The predicate function.
975     * @param {*} arg The argument to check.
976     * @returns {boolean} Whether the predicate returned true for the seventy-fifth argument.
977     */
978   _seventyFifthTrue(pred, arg) {
979     return pred(arg);
980   }
981
982   /**
983     * Checks if a predicate returns true for the seventy-sixth argument.
984     * @param {Function} pred The predicate function.
985     * @param {*} arg The argument to check.
986     * @returns {boolean} Whether the predicate returned true for the seventy-sixth argument.
987     */
988   _seventySixthTrue(pred, arg) {
989     return pred(arg);
990   }
991
992   /**
993     * Checks if a predicate returns true for the seventy-seventh argument.
994     * @param {Function} pred The predicate function.
995     * @param {*} arg The argument to check.
996     * @returns {boolean} Whether the predicate returned true for the seventy-seventh argument.
997     */
998   _seventySeventhTrue(pred, arg) {
999     return pred(arg);
1000  }
1001
1002  /**
1003    * Checks if a predicate returns true for the seventy-eighth argument.
1004    * @param {Function} pred The predicate function.
1005    * @param {*} arg The argument to check.
1006    * @returns {boolean} Whether the predicate returned true for the seventy-eighth argument.
1007    */
1008   _seventyEighthTrue(pred, arg) {
1009     return pred(arg);
1010  }
1011
1012  /**
1013    * Checks if a predicate returns true for the seventy-ninth argument.
1014    * @param {Function} pred The predicate function.
1015    * @param {*} arg The argument to check.
1016    * @returns {boolean} Whether the predicate returned true for the seventy-ninth argument.
1017    */
1018   _seventyNinthTrue(pred, arg) {
1019     return pred(arg);
1020  }
1021
1022  /**
1023    * Checks if a predicate returns true for the eighty argument.
1024    * @param {Function} pred The predicate function.
1025    * @param {*} arg The argument to check.
1026    * @returns {boolean} Whether the predicate returned true for the eighty argument.
1027    */
1028   _eightyTrue(pred, arg) {
1029     return pred(arg);
1030  }
1031
1032  /**
1033    * Checks if a predicate returns true for the eighty-first argument.
1034    * @param {Function} pred The predicate function.
1035    * @param {*} arg The argument to check.
1036    * @returns {boolean} Whether the predicate returned true for the eighty-first argument.
1037    */
1038   _eightyFirstTrue(pred, arg) {
1039     return pred(arg);
1040  }
1041
1042  /**
1043    * Checks if a predicate returns true for the eighty-second argument.
1044    * @param {Function} pred The predicate function.
1045    * @param {*} arg The argument to check.
1046    * @returns {boolean} Whether the predicate returned true for the eighty-second argument.
1047    */
1048   _eightySecondTrue(pred, arg) {
1049     return pred(arg);
1050  }
1051
1052  /**
1053    * Checks if a predicate returns true for the eighty-third argument.
1054    * @param {Function} pred The predicate function.
1055    * @param {*} arg The argument to check.
1056    * @returns {boolean} Whether the predicate returned true for the eighty-third argument.
1057    */
1058   _eightyThirdTrue(pred, arg) {
1059     return pred(arg);
1060  }
1061
1062  /**
1063    * Checks if a predicate returns true for the eighty-fourth argument.
1064    * @param {Function} pred The predicate function.
10
```

```

104             if (e.keyCode === this._ENTER) fun.call(n, e);
105         });
106     });
107 }
108 }
109
110 /**
111 * Detects actual value changes in an input and invokes a callback after
112 * validation.
113 * @param {string|HTMLElement} el Target element or selector for delegated
114 * listening.
115 * @param {Function} fun Callback executed when the value truly changes.
116 * @param {Function} [validator] Optional predicate to approve or reject the
117 * change.
118 */
119 onInputChange(el, fun, validator) {
120   var initLastVal = n =>
121     n.setAttribute('last-val', this.escapeHTML(this._readVal(n)));
122
123   var hasChanged = n =>
124     this.escapeHTML(this._readVal(n)) !== n.getAttribute('last-val');
125
126   var maybeFire = (n, e) => {
127     if (!hasChanged(n)) return;
128
129     if (typeof validator === 'function' && !validator.call(this, n, e)) {
130       n.value = n.getAttribute('last-val');
131       n.focus();
132       this.addClassMs(n, 'error', 400);
133       return;
134     }
135   };
136
137   initLastVal(n);
138   fun.call(n, e);
139 }
140
141 this._resolveEls(el).forEach(initLastVal);
142
143 if (typeof el === 'string') {
144   var always = () => true;
145   var doc = this._resolveDoc(el);
146   this._delegate(doc, 'focusout', el, always, e => maybeFire(e.target, e),
147                 true);
148   this._delegate(doc, 'keyup', el, e => e.keyCode === this._ENTER, e =>
149                 maybeFire(e.target, e));
150   this._delegate(doc, 'change', el, always, e => maybeFire(e.target, e));
151 } else {
152   var n = el;
153   ['blur', 'change'].forEach(t => n.addEventListener(t, e => maybeFire(n,
154     e)));
155   n.addEventListener('keyup', e => {
156     if (e.keyCode === this._ENTER) maybeFire(n, e);
157   });
158 }
159 }
```

```

153
154  /**
155   * Like 'onInputChange' but triggers only while the element remains focused
156   * (active input).
157   * @param {string|HTMLElement} el Target element or selector for delegated
158   * listening.
159   * @param {Function} fun Callback executed when the value changes during
160   * active editing.
161   * @param {Function} [validator] Optional predicate to approve or reject the
162   * change.
163   */
164 onActiveInputChange(el, fun, validator) {
165   var initLastVal = n =>
166     n.setAttribute('last-val', this.escapeHTML(this._readVal(n)));
167   var maybeFire = (n, e) => {
168     if (!hasChanged(n)) return;
169     if (typeof validator === 'function' && !validator.call(this, n, e)) {
170       n.value = n.getAttribute('last-val');
171       n.focus();
172       this.addClassMs(n, 'error', 400);
173       return;
174     }
175     initLastVal(n);
176     fun.call(n, e);
177   };
178   this._resolveEls(el).forEach(initLastVal);
179
180   if (typeof el === 'string') {
181     var doc = this._resolveDoc(el);
182     this._delegate(doc, 'focusout', el, () => true, e => maybeFire(e.target, e), true);
183     this._delegate(doc, 'keyup', el, e => e.keyCode === this._ENTER, e => maybeFire(e.target, e));
184   } else {
185     var n = el;
186     n.addEventListener('blur', e => maybeFire(n, e));
187     n.addEventListener('keyup', e => {
188       if (e.keyCode === this._ENTER) maybeFire(n, e);
189     });
190   }
191 }
192
193 /**
194  * Programmatically sets an inputs value and records it as the baseline
195  * for change tracking.
196  * @param {string|HTMLElement} el_or_sel The input element or selector
197  * identifying it.
198  * @param {string|number} val The value to assign and remember as the
199  * set      value.
200  */
201 setInputValue(el_or_sel, val) {

```

```

199     var n = this._resolveEl(el_or_sel);
200     if (!n) return;
201     n.value = val;
202     n.setAttribute('last-val', String(val));
203     n.setAttribute('set-val', String(val));
204 }
205
206 /**
207 * Updates an inputs value only if it is different from the current
208 * content, and records it.
209 * @param {string|HTMLElement} el_or_sel The input element or selector
210 * identifying it.
211 * @param {string|number} val The new value to write and store as the
212 * baseline.
213 */
214 updateInputValue(el_or_sel, val) {
215     var n = this._resolveEl(el_or_sel);
216     if (!n) return;
217     if (n.value != val) {
218         n.value = val;
219         n.setAttribute('last-val', String(val));
220         n.setAttribute('set-val', String(val));
221     }
222 }
223 /**
224 * Restores an input to the value previously stored with 'setInputValue'.
225 * @param {string|HTMLElement} el_or_sel The input element or selector
226 * identifying it.
227 */
228 resetInputValue(el_or_sel) {
229     var n = this._resolveEl(el_or_sel);
230     if (!n) return;
231     var v = n.getAttribute('set-val');
232     if (v !== null) n.value = v;
233 }
234 /**
235 * Adds or removes the 'changed' CSS class based on whether the current
236 * value differs from the stored baseline.
237 * @param {string|HTMLElement} el_or_sel The input element or selector
238 * identifying it.
239 */
240 showInputChanged(el_or_sel) {
241     var n = this._resolveEl(el_or_sel);
242     if (!n) return;
243     var set_val = n.getAttribute('set-val') ?? '';
244     n.classList.toggle('changed', String(n.value) !== set_val);
245 }
246 /**
247 * Validates that an input contains a numeric value and briefly highlights
248 * errors.
249 * @param {string|HTMLElement} el_or_sel The input element or selector
250 * identifying it.

```

```

246   * @returns {Array} Tuple of the parsed number and a validity flag .
247   */
248   validateInputNumber(el_or_sel) {
249     var n = this._resolveEl(el_or_sel);
250     if (!n) return [NaN, false];
251     var str = this._readVal(n).trim();
252     var num = Number(str);
253     if (str === '' || Number.isNaN(num)) {
254       n.focus();
255       this.addClassMs(n, 'error', 400);
256       return [num, false];
257     }
258     return [num, true];
259   }
260
261 /**
262  * Sets an inputs value and temporarily applies a 'warning' class for
263  * visual feedback.
264  * @param {string|HTMLElement} el_or_sel The input element or selector
265  * identifying it .
266  * @param {string|number} val The value to assign before flashing the
267  * warning.
268  */
269 setInputWithWarning(el_or_sel, val) {
270   var n = this._resolveEl(el_or_sel);
271   if (!n) return;
272   n.value = val;
273   this.addClassMs(n, 'warning', 400);
274
275 /**
276  * Adds a CSS class to an element for a specified duration , then removes it .
277  * @param {HTMLElement} node The element to which the class will be applied .
278  * @param {string} cls The CSS class name to toggle .
279  * @param {number} ms The number of milliseconds to keep the class before
280  * removal .
281  */
282 addClassMs(node, cls, ms) {
283   node.classList.add(cls);
284   setTimeout(() => node.classList.remove(cls), ms);
285
286 /**
287  * Observes an elements size changes and reports each new 'contentRect '
288  * to a callback .
289  * @param {string|HTMLElement} el_or_sel The element or selector to observe .
290  * @param {Function} cb Callback that receives the 'ResizeObserverEntry .
291  * contentRect ' whenever the element resizes .
292  * @returns {ResizeObserver|undefined} The observer instance , or 'undefined '
293  * if the element is not found .
294  */
295 onResize(el_or_sel, cb) {
296   var el = this._resolveEl(el_or_sel);
297   if (!el) return;
298   var ro = new ResizeObserver(ent => cb(ent[0].contentRect));

```

```

294     ro.observe(el);
295     return ro;
296   }
297
298 /**
299 * Converts various inputs into an array of HTMLElements.
300 * @param {string|HTMLElement|NodeList|Iterable<HTMLElement>} el_or_sel
301 Element, selector, or collection to resolve.
302 * @returns {HTMLElement[]} An array of resolved elements (possibly empty).
303 */
304 _resolveEls(el_or_sel) {
305   if (!el_or_sel) return [];
306   if (typeof el_or_sel === 'string') {
307     return Array.from(document.querySelectorAll(el_or_sel));
308   }
309   if (el_or_sel.nodeType === 1) {
310     return [el_or_sel];
311   }
312   if (typeof el_or_sel[Symbol.iterator] === 'function') {
313     return Array.from(el_or_sel).filter(n => n?.nodeType === 1);
314   }
315   return [];
316 }
317 /**
318 * Resolves and returns the first HTMLElement that matches the input
319 reference.
320 * @param {string|HTMLElement} el_or_sel Element or selector used to locate
321 one element.
322 * @returns {HTMLElement|null} The matched element or 'null' if none found.
323 */
324 _resolveEl(el_or_sel) {
325   return this._resolveEls(el_or_sel)[0] ?? null;
326 }
327 /**
328 * Retrieves the textual value from an input-type or content-editable
329 element.
330 * @param {HTMLElement} el The element whose value or text content should be
331 read.
332 * @returns {string} The raw string content of the element.
333 */
334 _readVal(el) {
335   return el.matches('[contenteditable]') ? el.textContent : el.value;
336 }
337 /**
338 * Determines the appropriate 'Document' context for a given element or
339 selector.
340 * @param {string|Node} el_or_sel Element, node list, or selector used to
341 infer a document.
342 * @returns {Document} The resolved document object.
343 */
344 _resolveDoc(el_or_sel) {
345   if (typeof el_or_sel !== 'string' && el_or_sel?.ownerDocument)

```

```

342         return el_or_sel.ownerDocument;
343     return document;
344 }
345 /**
346 * Registers a delegated event listener that triggers a handler when the
347 target matches a selector and passes an optional filter.
348 * @param {Document|ShadowRoot} doc The root node on which to listen for the
349 event.
350 * @param {string} type The event type to listen for (e.g., "keyup").
351 * @param {string} selector The CSS selector that qualifying targets must
352 match.
353 * @param {Function} filter Predicate that further filters qualifying events
354 .
355 * @param {Function} handler Function executed with 'this' bound to the
356 event target.
357 * @param {boolean} [useCapture=false] Whether to attach the listener in the
358 capture phase.
359 */
360 _delegate(doc, type, selector, filter, handler, useCapture = false) {
361     doc.addEventListener(
362         type,
363         e => {
364             const t = e.target;
365             if (t && t.matches(selector) && filter(e)) handler.call(t, e);
366         },
367         useCapture
368     );
369 }
370 }
371 }
372
373 exports.PRDC_JSLAB_LIB_GUI = PRDC_JSLAB_LIB_GUI;

```

Listing 100 - gui.js

```

1 /**
2 * @file JSLAB sandbox window init file
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 // Modules
9 // -----
10 const helper = require("../js/helper.js");
11 require("../js/init-config.js");
12 const { PRDC_JSLAB_LANGUAGE } = require('../js/language');
13
14 global.app_path = process.argv.find(e => e.startsWith('--app-path=')).split('=')
15   )[1].replace(/\.\.js\?\$/,'');
16
17 const { PRDC_JSLAB_LIB } = require('../js/sandbox/jslab');
18
19 // Global variables
20 global.language = new PRDC_JSLAB_LANGUAGE();
21 var jsl = new PRDC_JSLAB_LIB();

```

```

21
22 if(config.TEST) {
23     const { PRDC_JSLAB_TESTER } = require("../js/tester.js");
24     tester = new PRDC_JSLAB_TESTER('sandbox');
25     tester.runTests();
26 }

```

Listing 101 - init-sandbox.js

```

1 /**
2  * @file Init worker
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  * @version 0.0.1
7  */
8 "use strict";
9
10 global.fs = require('fs');
11 global.path = require('path');
12 global.app_path = process.argv.find(e => e.startsWith('--app-path=')).split('=')
13   )[1].replace(/\\"/g, '');
14 // Global variables
15 global.win = self;
16 global.worker_module;
17
18 /**
19  * Handle messages
20 */
21 self.addEventListener("message", function(e) {
22     if(e.data.type == 'configureWorker') {
23         var { PRDC_WORKER } = require(e.data.module_path);
24         global.worker_module = new PRDC_WORKER();
25     } else if(global.worker_module && e.data.hasOwnProperty('method')) {
26         global.worker_module[e.data.method](e.data);
27     }
28 });

```

Listing 102 - init-worker.js

```

1 /**
2  * @file JSLAB electron environment
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 if(!global.is_worker) {
9     var { ipcRenderer } = require('electron');
10 }
11
12 const { PRDC_JSLAB_FREECAD_LINK } = require('../freecad-link');
13 const { PRDC_JSLAB_OPENMODELICA_LINK } = require('../om-link');
14
15 const fs = require('fs');

```

```

16 const os = require('os');
17 const net = require('net');
18 const udp = require('dgram');
19 const cp = require("child_process");
20 const path = require("path");
21 const tcpPortUsed = require('tcp-port-used');
22 const { pathEqual } = require('path-equal');
23 const { Readable, Writable } = require('stream');
24 const { NativeModule } = require(app_path + '/build/Release/native_module');
25 const { AlphaShape3D } = require(app_path + '/build/Release/alpha_shape_3d');
26 const { extractFull } = require('node-7z');
27 const seedrandom = require('seedrandom');
28 const bin7zip = require('7zip-bin').path7za;
29 const PDFDocument = require('pdfkit');
30 const SVGtoPDF = require('svg-to-pdfkit');
31 const { PolynomialRegression } = require('ml-regression-polynomial');
32 const recast = require('recast');
33 const babel_parser = require('@babel/parser');
34 var SourceMapConsumer = require("source-map").SourceMapConsumer;
35 var { SerialPort } = require('serialport');
36 const jsdoc = require('jsdoc-api');

37 /**
38 * Class for JSLAB electron environment.
39 */
40 class PRDC_JSLAB_ENV {

41   /**
42    * Constructs a electron environment submodule object with access to JSLAB's
43    * electron environment functions.
44    * @constructor
45    * @param {Object} jsl - Reference to the main JSLAB object.
46    */
47   constructor(jsl) {
48     var obj = this;
49     this.jsl = jsl;
50
51     if(!global.is_worker) {
52       this.context = window;
53       this.debug = ipcRenderer.sendSync("sync-message", "get-debug-flag");
54       this.version = ipcRenderer.sendSync("sync-message", "get-app-version");
55       this.exe_path = ipcRenderer.sendSync("sync-message", "get-path", "exe");
56       this.platform = ipcRenderer.sendSync("sync-message", "get-platform");
57       this.speech = new SpeechSynthesisUtterance();
58       this.speech.voice = speechSynthesis.getVoices()[0];
59       this.navigator = navigator;
60       this.processors_number = navigator.hardwareConcurrency;
61       this.Cesium = Cesium;
62       this.online = this.navigator.onLine;
63       function onOnlineChange() {
64         obj.online = obj.navigator.onLine;
65       }
66       this.context.addEventListener('online', onOnlineChange);
67       this.context.addEventListener('offline', onOnlineChange);
68     } else {
69

```

```

70   this.context = global;
71   this.debug = global.debug;
72   this.version = global.version;
73   this.exe_path = undefined;
74   this.platform = global.platform;
75   this.processors_number = undefined;
76   this.Cesium = {};
77 }
78 this.native_module = new NativeModule();
79 this.AlphaShape3D = AlphaShape3D;
80 this.bin7zip = bin7zip;
81 this.seedRandom = seedrandom;
82 this.extractFull = extractFull;
83
84 this.process_pid = process.pid;
85 this.math = this.context.math;
86 this.fmin = this.context.fmin;
87 this.PDFDocument = PDFDocument;
88 this.SVGtoPDF = SVGtoPDF;
89 this.PolynomialRegression = PolynomialRegression;
90 this.os = os;
91 this.net = net;
92 this.udp = udp;
93 this.tcpPortUsed = tcpPortUsed;
94 this.recast = recast;
95 this.babel_parser = babel_parser;
96 this.SourceMapConsumer = SourceMapConsumer;
97 this.SourceMapConsumer.initialize({
98   "lib/mappings.wasm": app_path+/node_modules/source-map/lib/mappings.
99   wasm',
100 });
101 this.SerialPort = SerialPort;
102 this.jsdoc = jsdoc;
103
104 this.context.freecad_link = new PRDC_JSLAB_FREECAD_LINK(this.jsl);
105 this.context.om_link = new PRDC_JSLAB_OPENMODELICA_LINK(this.jsl);
106
107 // On IPC message
108 if(!global.is_worker) {
109   ipcRenderer.on("SandboxWindow", function(event, action, data) {
110     switch(action) {
111       case "eval-code":
112         obj.jsl.eval.evalCodeFromMain(...data);
113         break;
114       case "get-completions":
115         ipcRenderer.send("completions-" + data[0], obj.jsl.basic.
116         getCompletions(data));
117         break;
118       case "stop-loop":
119         obj.jsl.setStopLoop(data);
120         break;
121       case "run-last-script":
122         obj.jsl.eval.runLast();
123         break;
124       case "set-current-path":

```

```
123     obj.jsl.setPath(data);
124     break;
125   case "set-saved-paths":
126     obj.jsl.setSavedPaths(data);
127     break;
128   case "set-language":
129     language.set(data);
130     obj.jsl.figures._updateLanguage();
131     obj.jsl.windows._updateLanguage();
132     break;
133   }
134 });
135 }
136
137 // Functions
138 this.setImmediate = this.context.setImmediate;
139 this.clearImmediate = this.context.clearImmediate;
140 this.pathEqual = pathEqual;
141 this.Readable = Readable;
142 this.Writable = Writable;
143
144 // Which properties and methods to export to context
145 this.exports = [ 'debug', 'version', 'platform' ];
146 }
147
148 /**
149 * Opens a window based on the provided ID.
150 * @param {number} id - The ID of the window to open.
151 * @returns {((boolean|BrowserWindow))} - The opened window or false if the
152 * window does not exist.
153 */
154 openWindow(wid, file = "blank.html") {
155   if(!global.is_worker) {
156     var obj = this;
157     var sub_context = this.context.open(file, wid);
158     sub_context.addEventListener("error", function(err) {
159       if(err && err.hasOwnProperty('error')) {
160         obj.error(err.error.stack);
161       } else {
162         obj.error(err.message);
163       }
164     });
165   var loadCheckInterval = setInterval(function() {
166     try {
167       if(isFinite(sub_context.wid)) {
168         clearInterval(loadCheckInterval);
169       }
170     } catch {
171       sub_context.close();
172       obj.jsl.windows._closedWindow(wid);
173       clearInterval(loadCheckInterval);
174       obj.error('@openWindow: '+language.string(174));
175     }
176   }, 10);
177 }
```



```

232     };
233
234     sub_context.openDevTools = function() {
235       return obj.jsl.windows.open_windows[wid].openDevTools();
236     };
237
238     sub_context.printToPdf = function(options) {
239       return obj.jsl.windows.open_windows[wid].printToPdf(options);
240     };
241
242     sub_context.document.addEventListener("keydown", function(e) {
243       if(e.key === 'F12') {
244         sub_context.openDevTools();
245       } else if(e.ctrlKey && e.key.toLowerCase() === 'c') {
246         if(obj.getWinSelectionText(sub_context) === "") {
247           // No selected text
248           obj.jsl.setStopLoop(true);
249           e.stopPropagation();
250           e.preventDefault();
251         }
252       }
253     });
254     resolve(sub_context);
255   }, false);
256 });
257 }
258
259 /**
260 * Retrieves the selected text from a given window.
261 * @param {Window} context - The window context to get the selection text
262 *   from.
263 * @returns {string} - The selected text.
264 */
265 getWinSelectionText(context) {
266   var text = "";
267   if(context.getSelection) {
268     text = context.getSelection().toString();
269   } else if(context.document.selection && context.document.selection.type !==
270     "Control") {
271     text = context.document.selection.createRange().text;
272   }
273   return text;
274 }
275 /**
276 * Closes a window or all windows based on the provided ID.
277 * @param {number} id - The ID of the window to close, or "all" to close all
278 *   windows.
279 */
280 closeWindow(wid) {
281   if(!global.is_worker) {
282     var obj = this;
283     if(wid === "all") {
284       Object.keys(this.jsl.windows.open_windows).forEach(function(key) {

```

```
284     obj.jsl.windows.open_windows[key].context.close();
285     obj.jsl.windows._closedWindow(key);
286   });
287   return true;
288 } else if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
289   obj.jsl.windows.open_windows[key].context.close();
290   obj.jsl.windows._closedWindow(key);
291   return true;
292 }
293 }
294 return false;
295 }

296 /**
297 * Shows the specified window.
298 * @param {number} wid - The ID of the window to show.
299 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
300 */
301 showWindow(wid) {
302   if(!global.is_worker) {
303     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
304       ipcRenderer.sendSync("sync-message", "call-sub-win-method",
305         [wid, 'show']);
306       return true;
307     }
308   }
309   return false;
310 }

311 /**
312 * Hides the specified window.
313 * @param {number} wid - The ID of the window to hide.
314 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
315 */
316 hideWindow(wid) {
317   if(!global.is_worker) {
318     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
319       ipcRenderer.sendSync("sync-message", "call-sub-win-method",
320         [wid, 'hide']);
321       return true;
322     }
323   }
324   return false;
325 }

326 /**
327 * Brings the specified window to the foreground.
328 * @param {number} wid - The ID of the window to focus.
329 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
330 */
331 focusWindow(wid) {
332   if(!global.is_worker) {
333     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
334       ipcRenderer.sendSync("sync-message", "call-sub-win-method",
335         [wid, 'focus']);
```

```
339         return true;
340     }
341 }
342 return false;
343 }

344 /**
345 * Minimizes the specified window.
346 * @param {number} wid - The ID of the window to minimize.
347 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
348 */
349 minimizeWindow(wid) {
350     if(!global.is_worker) {
351         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
352             ipcRenderer.sendSync("sync-message", "call-sub-win-method",
353                 [wid, 'minimize']);
354             return true;
355         }
356     }
357     return false;
358 }

359 }

360 /**
361 * Centers the specified window on the screen.
362 * @param {number} wid - The ID of the window to center.
363 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
364 */
365 centerWindow(wid) {
366     if(!global.is_worker) {
367         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
368             ipcRenderer.sendSync("sync-message", "call-sub-win-method",
369                 [wid, 'center']);
370             return true;
371         }
372     }
373     return false;
374 }

375 }

376 /**
377 * Moves the specified window to the top.
378 * @param {number} wid - The ID of the window to move to top.
379 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
380 */
381 moveTopWindow(wid) {
382     if(!global.is_worker) {
383         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
384             ipcRenderer.sendSync("sync-message", "call-sub-win-method",
385                 [wid, 'moveTop']);
386             return true;
387         }
388     }
389     return false;
390 }

391 }

392 /**
393 *
```

```
394     * Sets the size of a specified window.  
395     * @param {number} wid - The ID of the window.  
396     * @param {number} width - The new width of the window.  
397     * @param {number} height - The new height of the window.  
398     * @returns {boolean|undefined} - Returns false if the window ID is invalid.  
399     */  
400     setWindowSize(wid, width, height) {  
401         if(!global.is_worker) {  
402             if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
403                 return ipcRenderer.sendSync("sync-message", "call-sub-win-method",  
404                     [wid, "setSize", width, 49 + height]);  
405             }  
406         }  
407         return false;  
408     }  
409  
410     /**  
411     * Sets the position of a specified window.  
412     * @param {number} wid - The ID of the window.  
413     * @param {number} left - The new left position of the window.  
414     * @param {number} top - The new top position of the window.  
415     * @returns {boolean|undefined} - Returns false if the window ID is invalid.  
416     */  
417     setWindowPos(wid, left, top) {  
418         if(!global.is_worker) {  
419             if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
420                 return ipcRenderer.sendSync("sync-message", "call-sub-win-method",  
421                     [wid, "setPosition", left, top]);  
422             }  
423         }  
424         return false;  
425     }  
426  
427     /**  
428     * Sets whether the specified window is resizable.  
429     * @param {number} wid - The ID of the window.  
430     * @param {boolean} state - The resizable state to set.  
431     * @returns {boolean|undefined} - Returns false if the window ID is invalid.  
432     */  
433     setWindowResizable(wid, state) {  
434         if(!global.is_worker) {  
435             if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
436                 return ipcRenderer.sendSync("sync-message", "call-sub-win-method",  
437                     [wid, "setResizable", state]);  
438             }  
439         }  
440         return false;  
441     }  
442  
443     /**  
444     * Sets whether the specified window is movable.  
445     * @param {number} wid - The ID of the window.  
446     * @param {boolean} state - The movable state to set.  
447     * @returns {boolean|undefined} - Returns false if the window ID is invalid.  
448     */
```

```

449 setWindowMovable(wid, state) {
450   if(!global.is_worker) {
451     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
452       return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
453         [wid, "setMovable", state]);
454     } else {
455       return false;
456     }
457   }
458 }
459 /**
460 * Sets the aspect ratio of the specified window.
461 * @param {number} wid - The ID of the window.
462 * @param {number} aspect_ratio - The aspect ratio to set (width/height).
463 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
464 */
465 setWindowAspectRatio(wid, aspect_ratio) {
466   if(!global.is_worker) {
467     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
468       return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
469         [wid, "setAspectRatio", aspect_ratio]);
470     }
471   }
472   return false;
473 }
474 /**
475 * Sets the opacity of the specified window.
476 * @param {number} wid - The ID of the window.
477 * @param {number} opacity - The opacity level to set (0.0 to 1.0).
478 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
479 */
480 setWindowOpacity(wid, opacity) {
481   if(!global.is_worker) {
482     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
483       return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
484         [wid, "setOpacity", opacity]);
485     }
486   }
487   return false;
488 }
489 /**
490 * Sets state of fullscreen of window if not running in a worker thread.
491 * @param {string} wid - The window ID.
492 * @param {string} state - fullscreen state.
493 * @returns {boolean|undefined} False if the window does not exist,
494 *          undefined if in a worker thread.
495 */
496 setWindowFullscreen(wid, state) {
497   if(!global.is_worker) {
498     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
499       return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
500         [wid, "setFullScreen", state]);
501     }
502   }

```

```
503         }
504     }
505     return false;
506 }
507
508 /**
509 * Sets the title of the specified window if not running in a worker thread.
510 * @param {string} wid - The window ID.
511 * @param {string} title - The new title for the window.
512 * @returns {boolean|undefined} False if the window does not exist,
513         undefined if in a worker thread.
514 */
515 setWindowTitle(wid, title) {
516     if(!global.is_worker) {
517         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
518             this.jsl.windows.open_windows[wid].context.document.title = title;
519         }
520     }
521     return false;
522 }
523 /**
524 * Retrieves the size of a specified window.
525 * @param {number} wid - The ID of the window.
526 * @returns {Array|boolean} - Returns an array [width, height] or false if
527         the window ID is invalid.
528 */
529 getSize(wid) {
530     if(!global.is_worker) {
531         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
532             return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
533                 [wid, "getSize"]);
534         }
535     }
536     return false;
537 }
538 /**
539 * Retrieves the position of a specified window.
540 * @param {number} wid - The ID of the window.
541 * @returns {Array|boolean} - Returns an array [left, top] or false if the
542         window ID is invalid.
543 */
544 getPosition(wid) {
545     if(!global.is_worker) {
546         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
547             return ipcRenderer.sendSync("sync-message", "call-sub-win-method",
548                 [wid, "getPosition"]);
549         }
550     }
551     return false;
552 }
553 /**
554 * Opens the developer tools for a specified window in the renderer process.
```



```
555 * Only available if not in a worker context.
556 * @param {string} wid - The window ID.
557 * @returns {boolean} True if the developer tools were opened; otherwise,
558 *             false.
559 */
560 openWindowDevTools(wid) {
561     if(!global.is_worker) {
562         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
563             return ipcRenderer.sendSync("sync-message", "open-sub-win-devtools",
564                                         wid);
565         }
566     }
567 }
568 /**
569 * Returns media source id for a specified window in the renderer process.
570 * Only available if not in a worker context.
571 * @param {string} wid - The window ID.
572 * @returns {String|boolean} Media source id if there is window; otherwise,
573 *             false.
574 */
575 getWindowMediaSourceId(wid) {
576     if(!global.is_worker) {
577         if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {
578             return ipcRenderer.sendSync("sync-message", "get-sub-win-source-id",
579                                         wid);
580         }
581     }
582 }
583 /**
584 * Opens the developer tools for the sandbox environment.
585 */
586 openSandboxDevTools() {
587     ipcRenderer.send("MainProcess", "show-sandbox-dev-tools");
588 }
589 /**
590 * Clears local storage.
591 */
592 clearStorage() {
593     if(!global.is_worker) {
594         ipcRenderer.send("MainWindow", "clear-storage");
595     }
596 }
597 }
598 /**
599 * Creates a directory at the specified path. If the directory already
600 * exists, no action is taken.
601 * @param {string} directory - The path where the directory will be created.
602 * @returns {boolean} True if the directory was created or already exists,
603 *             false if an error occurred.
604 */

```

```

604     makeDirectory(directory) {
605         try {
606             fs.mkdirSync(directory, { recursive: true });
607             return true;
608         } catch(err) {
609             this.jsl._console.log(err);
610             if(err.code === 'EEXIST') {
611                 return true;
612             } else {
613                 return false;
614             }
615         }
616     }
617
618     /**
619      * Checks if a directory exists at the specified path.
620      * @param {string} directory - The path to the directory.
621      * @returns {boolean} - True if the directory exists, false otherwise.
622      */
623     checkDirectory(directory) {
624         var lstat = fs.lstatSync(directory, {throwIfNoEntry: false});
625         if(lstat !== undefined && lstat.isDirectory()) {
626             return true;
627         } else {
628             return false;
629         }
630     }
631
632     /**
633      * Checks if a file exists at the specified path.
634      * @param {string} file_path - The path to the file.
635      * @returns {boolean} - True if the file exists, false otherwise.
636      */
637     checkFile(file_path) {
638         var lstat = fs.lstatSync(file_path, {throwIfNoEntry: false});
639         if(lstat !== undefined && lstat.isFile()) {
640             return true;
641         } else {
642             return false;
643         }
644     }
645
646     /**
647      * Checks if the provided path is an absolute path.
648      * @param {string} file_path The file path to check.
649      * @returns {boolean} True if the path is absolute, false otherwise.
650      */
651     pathIsAbsolute(file_path) {
652         if(typeof file_path === 'string') {
653             return path.isAbsolute(file_path);
654         } else {
655             return false;
656         }
657     }
658

```

```
659  /**
660   * Joins all given path segments together using the platform-specific
661   * separator as a delimiter.
662   * @param {...string} paths The path segments to join.
663   * @returns {string} The combined path.
664   */
665   pathJoin(...arg) {
666     return path.join(...arg);
667   }
668 /**
669  * Retrieves the platform-specific path separator.
670  * @returns {string} The path separator.
671  */
672   pathSep() {
673     return path.sep;
674   }
675 /**
676  * Extracts the basename of a path, possibly removing the specified
677  * extension.
678  * @param {...string} paths The path to extract the basename from, followed
679  * by an optional extension to remove.
680  * @returns {string} The basename of the path.
681  */
682   pathBaseName(...arg) {
683     return path.basename(...arg);
684   }
685 /**
686  * Retrieves the file name without its extension from the given file path.
687  * @param {string} file_path - The path to the file.
688  * @returns {string} - The file name without the extension.
689  */
690   pathFileName(file_path) {
691     return path.parse(file_path).name;
692   }
693 /**
694  * Extracts the directory of file.
695  * @param {String} path The filesystem path from which to extract the
696  * directory.
697  * @returns {String} The directory from the given path.
698  */
699   pathDirName(file_path) {
700     return path.dirname(file_path);
701   }
702 /**
703  * Retrieves the file extension from the given file path.
704  * @param {string} file_path - The path to the file.
705  * @returns {string} - The file extension.
706  */
707   pathExtName(file_path) {
708     return path.extname(file_path);
709   }
```

```

710     }
711
712     /**
713      * Resolves a sequence of path segments into an absolute path.
714      * @param {string} path_in - The path or sequence of paths to resolve.
715      * @returns {string} - The resolved absolute path.
716      */
717     pathResolve(path_in) {
718       return path.resolve(path_in);
719     }
720
721     /**
722      * Computes the relative path from one path to another.
723      * @param {string} from - The starting path.
724      * @param {string} to - The target path.
725      * @returns {string} - The relative path from the 'from' path to the 'to'
726      * path.
727      */
728     pathRelative(from, to) {
729       return path.relative(from, to);
730     }
731
732     /**
733      * Normalizes a given path, resolving '..' and '.' segments.
734      * @param {string} path - The path to normalize.
735      * @returns {string} - The normalized path.
736      */
737     pathNormalize(path_in) {
738       return path.normalize(path_in);
739     }
740
741     /**
742      * Parses a file path into its component parts.
743      * @param {string} path - The file path to parse.
744      * @returns {Object} An object containing properties like 'root', 'dir',
745      * 'base', 'ext', and 'name'.
746      */
747     pathParse(path_in) {
748       return path.parse(path_in);
749     }
750
751     /**
752      * Reads the content of a file synchronously.
753      * @param {string} file_path The path to the file.
754      * @returns {((Buffer|string|false))} The file content or false in case of an
755      * error.
756      */
757     readFileSync(... arg) {
758       try {
759         return fs.readFileSync(... arg);
760       } catch(err) {
761         this.jsl._console.log(err);
762         return false;
763       }
764     }

```

```
762
763  /**
764   * Copies a file synchronously with the given arguments.
765   * @param {...any} arg - Arguments to pass to fs.copyFileSync.
766   * @returns {boolean} - Returns true if the copy was successful, false
767   * otherwise.
768   */
769   copyFileSync (... arg) {
770     try {
771       return fs.copyFileSync (... arg);
772     } catch (err) {
773       this.jsl._console.log (err);
774       return false;
775     }
776   }
777 /**
778  * Writes data to a file synchronously.
779  * @param {string} file_path - The path to the file.
780  * @param {any} data - The data to write.
781  * @param {boolean} throw_flag - Whether to throw an error on failure.
782  * @returns {boolean} - Returns true if the write was successful, false
783  * otherwise.
784  */
785   writeFileSync (file_path , data , throw_flag) {
786     try {
787       return fs.writeFileSync (file_path , data);
788     } catch (err) {
789       this.jsl._console.log (err);
790       if (throw_flag) {
791         this.error ('@writeFileSync: '+err);
792       }
793       return false;
794     }
795   }
796 /**
797  * Removes a file or directory synchronously.
798  * @param {string} path - The path to remove.
799  * @param {boolean} [throw_flag=true] - Whether to throw an error on failure
800  * .
801  * @returns {boolean} - Returns true if the removal was successful, false
802  * otherwise.
803  */
804   rmSync (path_in , throw_flag = true) {
805     try {
806       return fs.rmSync (path_in , { recursive: true , force: true });
807     } catch (err) {
808       this.jsl._console.log (err);
809       if (throw_flag) {
810         this.error ('@rmSync: '+err);
811       }
812       return false;
813     }
814   }
```

```
813
814  /**
815   * Reads the contents of a directory synchronously.
816   * @param {string} folder The path to the directory.
817   * @returns {string[]|false} An array of filenames or false in case of an
818   * error.
819   */
820   readDir(... args) {
821     try {
822       return fs.readdirSync(... args);
823     } catch(err) {
824       this.jsl._console.log(err);
825       this.error(`@readDir: ${err}`);
826       return false;
827     }
828
829  /**
830   * Displays a dialog to open files , asynchronously returning the selected
831   * files' paths.
832   * @param {Object} options The options for the dialog.
833   * @returns {Promise<string[]>} A promise that resolves to the paths of
834   * selected files.
835   */
836   showOpenDialog(options) {
837     if(!global.is_worker) {
838       return ipcRenderer.invoke("dialog", "showOpenDialog", options);
839     }
840     return false;
841
842  /**
843   * Displays a dialog to open files , synchronously returning the selected
844   * files' paths.
845   * @param {Object} options The options for the dialog.
846   * @returns {string[]} The paths of selected files.
847   */
848   showOpenDialogSync(options) {
849     if(!global.is_worker) {
850       return ipcRenderer.sendSync("dialog", "showOpenDialogSync", options);
851     }
852     return false;
853
854  /**
855   * Displays a dialog to save file , asynchronously returning the selected
856   * files' paths.
857   * @param {Object} options The options for the dialog.
858   * @returns {Promise<string[]>} A promise that resolves to the paths of
859   * selected files.
860   */
861   showSaveDialog(options) {
862     if(!global.is_worker) {
863       return ipcRenderer.invoke("dialog", "showSaveDialog", options);
864     }
865   }
```

```

862     return false;
863 }
864
865 /**
866 * Displays a dialog to save file , synchronously returning the selected
867 files' paths .
868 * @param {Object} options The options for the dialog .
869 * @returns {string[]} The paths of selected files .
870 */
871 showSaveDialogSync(options) {
872     if(!global.is_worker) {
873         return ipcRenderer.sendSync("dialog", "showSaveDialogSync", options);
874     }
875     return false;
876 }
877
878 /**
879 * Displays a message box , synchronously returning the index of the clicked
880 button .
881 * @param {Object} options The options for the message box .
882 * @returns {number} The index of the clicked button .
883 */
884 showMessageBox(options) {
885     if(!global.is_worker) {
886         return ipcRenderer.sendSync("dialog", "showMessageBoxSync", options);
887     }
888     return false;
889 }
890 /**
891 * Checks if a loop stop flag has been set , indicating whether to halt
892 execution .
893 * @returns {boolean} True if the loop should stop , false otherwise .
894 */
895 checkStopLoop() {
896     if(!global.is_worker) {
897         return ipcRenderer.sendSync("sync-message", "check-stop-loop");
898     }
899     return false;
900 }
901 /**
902 * Resets the loop stop flag to allow continued execution .
903 * @returns {undefined} No return value .
904 */
905 resetStopLoop() {
906     if(!global.is_worker) {
907         return ipcRenderer.sendSync("sync-message", "reset-stop-loop");
908     }
909     return false;
910 }
911 /**
912 * Opens the editor window and optionally loads a script .
913 * @param {string} filename - The path to the script file to open .

```

```
914     * @param {number} lineno - Line number to highlight .
915     */
916    editor(filename , lineno) {
917      if(! global.is_worker) {
918        ipcRenderer.send("MainProcess" , "show-editor");
919        if(typeof filename == "string") {
920          ipcRenderer.send("EditorWindow" , "open-script" , [filename , lineno]);
921        }
922      }
923    }
924
925 /**
926  * Sends a message to display in the main window of the application .
927  * @param {...any} args - The messages to send for display in the main
928  *   window .
929  */
930  disp(...args) {
931    if(! global.is_worker) {
932      var obj = this;
933      args.forEach(function(msg) {
934        ipcRenderer.send("MainWindow" , "disp" , obj.jsl.format.safeStringify(
935          msg));
936      });
937    }
938 /**
939  * Sends a message to display in the main window of the application with
940  * monospaced font .
941  * @param {...any} args - The messages to send for display in the main
942  *   window with monospaced font .
943  */
944  dispMonospaced(...args) {
945    if(! global.is_worker) {
946      var obj = this;
947      args.forEach(function(msg) {
948        ipcRenderer.send("MainWindow" , "disp-monospaced" , obj.jsl.format.
949          safeStringify(msg));
950      });
951    }
952 /**
953  * Sends a message to display latex in the main window of the application .
954  * @param {string} expr The expression to be displayed .
955  */
956  dispLatex(expr) {
957    if(! global.is_worker) {
958      ipcRenderer.send("MainWindow" , "disp-latex" , expr);
959    }
960
961 /**
962  * Triggers the display of CMD help content in the main window .
963  */

```

```
964     cmd_help() {
965         if(!global.is_worker) {
966             ipcRenderer.send("MainWindow", "help");
967         }
968     }
969
970 /**
971 * Triggers the display of informational content in the main window.
972 */
973 info() {
974     if(!global.is_worker) {
975         ipcRenderer.send("MainWindow", "info");
976     }
977 }
978
979 /**
980 * Opens the settings interface in the main window.
981 */
982 settings() {
983     if(!global.is_worker) {
984         ipcRenderer.send("MainWindow", "settings");
985     }
986 }
987
988 /**
989 * Sends an error message to be displayed in the main window.
990 * @param {string} msg The error message to be displayed.
991 */
992 error(msg, throw_flag = true) {
993     if(!global.is_worker) {
994         this.jsl.stop_loop = true;
995         this.jsl.onStopLoop(false);
996         this.jsl.no_ans = true;
997         this.jsl.ignore_output = true;
998         this.jsl.onEvaluated();
999         if(throw_flag) {
1000             throw new Error(msg);
1001         } else {
1002             ipcRenderer.send("MainWindow", "error", this.jsl.format.safeStringify(
1003                 msg));
1004         }
1005     }
1006
1007 /**
1008 * Sends an internal error message to be displayed, indicating an error
1009 * within the application's internals.
1010 * @param {string} msg The internal error message.
1011 */
1012 errorInternal(msg) {
1013     if(!global.is_worker) {
1014         ipcRenderer.send("MainWindow", "internal-error", msg);
1015     }
1016 }
```

```

1017 /**
1018 * Sends a warning message to be displayed in the main window.
1019 * @param {string} msg The warning message.
1020 */
1021 warn(msg) {
1022     if(!global.is_worker) {
1023         ipcRenderer.send("MainWindow", "warn", msg);
1024     }
1025 }
1026
1027 /**
1028 * Clears the command window, removing all current content.
1029 */
1030clc() {
1031     if(!global.is_worker) {
1032         ipcRenderer.send("MainWindow", "clear");
1033     }
1034 }
1035
1036 /**
1037 * Lists the contents of the current directory and returns them.
1038 * @returns {string[]} The contents of the current directory.
1039 */
1040listFolderContents() {
1041     return fs.readdirSync(this.jsl.current_path);
1042 }
1043
1044 /**
1045 * Requests an update of the file browser to reflect current directory or
1046 * file changes.
1047 */
1048updateFileBrowser() {
1049     if(!global.is_worker) {
1050         ipcRenderer.send("MainWindow", "update-file-browser");
1051     }
1052
1053 /**
1054 * Displays the result of an operation in the workspace area of the main
1055 * window.
1056 * @param {string} data The data or result to display.
1057 */
1058showAns(data) {
1059     if(!global.is_worker) {
1060         ipcRenderer.send("MainWindow", "show-ans", data);
1061     }
1062
1063 /**
1064 * Requests an update of the workspace to reflect changes in variables or
1065 * state.
1066 */
1067updateWorkspace() {
1068     if(!global.is_worker) {
1069         ipcRenderer.send("MainWindow", "update-workspace");

```



```
1069     }
1070 }
1071
1072 /**
1073 * Sets the workspace with the provided data , replacing the current state .
1074 */
1075 setWorkspace() {
1076     if(!global.is_worker) {
1077         ipcRenderer.send("MainWindow" , "set-workspace" , this.jsl.getWorkspace())
1078         ;
1079     }
1080 }
1081 /**
1082 * Saves a new path to the application's memory for quick access .
1083 * @param {string} new_path The path to save .
1084 */
1085 savePath(new_path) {
1086     if(!global.is_worker) {
1087         ipcRenderer.send("MainWindow" , "save-path" , new_path);
1088     }
1089 }
1090 /**
1091 * Removes a previously saved path from the application's memory .
1092 * @param {string} saved_path The path to remove .
1093 */
1094 removePath(saved_path) {
1095     if(!global.is_worker) {
1096         ipcRenderer.send("MainWindow" , "remove-path" , saved_path);
1097     }
1098 }
1099
1100 /**
1101 * Sets the application's status message .
1102 * @param {string} state The current state of the application .
1103 * @param {string} txt The text message to display as status .
1104 */
1105 setStatus(state , txt) {
1106     if(!global.is_worker) {
1107         ipcRenderer.send("MainWindow" , "set-status" , [state , txt]);
1108     }
1109 }
1110
1111 /**
1112 * Updates the application's statistics , typically displayed in the status
1113 * bar or a similar area .
1114 * @param {object} stats The statistical data to set .
1115 */
1116 setStats(stats) {
1117     if(!global.is_worker) {
1118         ipcRenderer.send("MainWindow" , "set-stats" , stats);
1119     }
1120 }
1121
```

```

1122 /**
1123 * Notifies the main window that code evaluation has started.
1124 */
1125 codeEvaluating() {
1126   if(!global.is_worker) {
1127     ipcRenderer.send("MainWindow", "code-evaluating");
1128   }
1129 }
1130 /**
1131 * Notifies the main window that code evaluation has finished.
1132 */
1133 codeEvaluated() {
1134   if(!global.is_worker) {
1135     ipcRenderer.send("MainWindow", "code-evaluated");
1136     ipcRenderer.send("MainProcess", "code-evaluated");
1137   }
1138 }
1139 /**
1140 * Checks if a script resides within the current active directory or a saved
1141 * directory, and if not, prompts to find the script.
1142 * @param {string} script_path The path of the script to check.
1143 * @returns {boolean} Returns true if the script directory is unknown,
1144 * prompting for location; otherwise false.
1145 */
1146 checkScriptDir(script_path) {
1147   var script_dir = this.addPathSep(path.dirname(script_path));
1148   if(pathEqual(script_dir, this.jsl.current_path) ||
1149     this.jsl.saved_paths.includes(script_dir)) {
1150     return false;
1151   } else {
1152     ipcRenderer.send("MainWindow", "unknown-script-dir");
1153     return true;
1154   }
1155 }
1156 /**
1157 * Ensures a path string ends with a path separator, appending one if
1158 * necessary.
1159 * @param {string} path_str The path string to modify.
1160 * @returns {string} The modified path string with a trailing separator.
1161 */
1162 addPathSep(path_str) {
1163   if(path_str && path_str[path_str.length -1] != path.sep) {
1164     path_str += path.sep;
1165   }
1166   return path_str;
1167 }
1168 /**
1169 * Changes the current working directory to the specified path.
1170 * @param {string} new_path The path to set as the current working directory
1171 .
1172 */

```

```

1173 cd(new_path) {
1174   if(!global.is_worker) {
1175     ipcRenderer.send("MainWindow", "set-current-path", new_path,
1176       undefined, false);
1177   }
1178 }
1179 /**
1180 * Retrieves a default path based on a specified type, e.g., documents,
1181 * downloads.
1182 * @param {string} type The type of default path to retrieve.
1183 * @returns {string} The default path for the specified type.
1184 */
1185 getDefaultPath(type) {
1186   if(!global.is_worker) {
1187     return ipcRenderer.sendSync("sync-message", "get-path", type) + path.sep
1188     ;
1189   }
1190   return false;
1191 }
1192 /**
1193 * Opens the specified folder in the file manager.
1194 * @param {string} file_path The path of the folder to open.
1195 */
1196 openFolder(file_path) {
1197   if(!global.is_worker) {
1198     return ipcRenderer.send("MainProcess", "open-folder", file_path);
1199   }
1200   return false;
1201 }
1202 /**
1203 * Opens the specified directory in the file manager. This method is similar
1204 * to 'openFolder'.
1205 * @param {string} file_path The path of the directory to open.
1206 */
1207 openDir(file_path) {
1208   if(!global.is_worker) {
1209     return ipcRenderer.send("MainProcess", "open-dir", file_path);
1210   }
1211   return false;
1212 }
1213 /**
1214 * Shows the specified file in the folder using the file manager.
1215 * @param {string} file_path The path of the file to show.
1216 */
1217 /**
1218 * Shows the specified file in the folder using the file manager.
1219 * @param {string} file_path The path of the file to show.
1220 */
1221 /**
1222 * Shows the specified file in the folder using the file manager.
1223 */

```

```

1224
1225  /**
1226   * Shows the specified file in the directory using the file manager. This is
1227   * similar to 'showFileInFolder'.
1228   * @param {string} file_path The path of the file to highlight.
1229   */
1230 showFileInDir(file_path) {
1231   if(!global.is_worker) {
1232     return ipcRenderer.send("MainProcess", "show-file-in-dir", file_path);
1233   }
1234   return false;
1235 }
1236 /**
1237  * Retrieves desktop sources synchronously by sending an IPC message.
1238  * @returns {DesktopSource[]|undefined} An array of desktop sources if not
1239  * in a worker, otherwise undefined.
1240 */
1241 getDesktopSources() {
1242   if(!global.is_worker) {
1243     return ipcRenderer.sendSync("get-desktop-sources");
1244   }
1245   return false;
1246 }
1247 /**
1248  * Prints window to PDF format.
1249  * @param {string} wid - The window ID.
1250  * @param {Object} options - Options for printing
1251  * @returns {Buffer} Generated PDF data.
1252 */
1253 printWindowToPdf(wid, options) {
1254   if(!global.is_worker) {
1255     return ipcRenderer.invoke('print-sub-win-to-pdf', wid, options);
1256   }
1257   return false;
1258 }
1259 /**
1260  * Executes a system command asynchronously.
1261  * @param {...*} args Command arguments.
1262 */
1263 exec(...args){
1264   return cp.exec(...args);
1265 }
1266
1267 /**
1268  * Executes a system command synchronously and returns the result.
1269  * @param {...*} args Command arguments.
1270  * @returns {*} The result of the command execution.
1271 */
1272 execSync(...args){
1273   return cp.execSync(...args);
1274 }
1275
1276

```

```

1277  /**
1278   * Spawns child process synchronously.
1279   * @param {...*} args Command arguments.
1280   */
1281 spawnSync (...args) {
1282   return cp.spawnSync (...args);
1283 }
1284
1285 /**
1286  * Spawns child process assynchronously.
1287  * @param {...*} args Command arguments.
1288  */
1289 spawn (...args) {
1290   return cp.spawn (...args);
1291 }
1292
1293 /**
1294  * Resets the sandbox environment by sending a synchronous IPC message if
1295  * not in a worker.
1296  * @returns {void}
1297  */
1298 resetSandbox () {
1299   if (!global.is_worker) {
1300     ipcRenderer.sendSync ("sync-message", "reset-sandbox");
1301   }
1302
1303 /**
1304  * Resets the app.
1305  * @returns {void}
1306  */
1307 resetApp () {
1308   if (!global.is_worker) {
1309     ipcRenderer.sendSync ("sync-message", "reset-app");
1310   }
1311 }
1312 }
1313
1314 exports.PRDC_JSLAB_ENV = PRDC_JSLAB_ENV;

```

Listing 103 - jslab-env-electron.js

```

1 /**
2  * @file JSLAB library eval
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9  * Class for JSLAB library eval.
10 */
11 class PRDC_JSLAB_EVAL {
12
13 /**
14  * Constructs a eval submodule object with access to JSLAB's electron

```

```

    environment functions.

15  * @constructor
16  * @param {Object} jsl - Reference to the main JSLAB object .
17  */
18  constructor(jsl) {
19    var obj = this;
20    this.jsl = jsl;

21
22  // Errors handling
23  this.jsl.context.addEventListener('unhandledrejection', function(e) {
24    if(e && e.reason) {
25      if(obj.jsl.ready) {
26        if(e.reason.hasOwnProperty('stack')) {
27          obj.rewriteError('@jslab: '+e.reason.stack.toString());
28        } else if(e.reason.message) {
29          obj.jsl.env.error('@jslab: '+e.reason.message, false);
30        }
31      } else {
32        console.log(e);
33        obj.jsl.env.errorInternal('@jslab [FATAL INTERNAL]: ' + e.message);
34      }
35      e.preventDefault();
36    }
37  });

38
39  this.jsl.context.addEventListener('error', function(e) {
40    if(obj.jsl.ready) {
41      if(e && e.error && 'stack' in e.error) {
42        obj.rewriteError('@jslab: '+e.error.stack.toString());
43      } else if(e && e.error && e.error.message) {
44        obj.jsl.env.error('@jslab: '+e.error.message, false);
45      }
46    } else {
47      console.log(e);
48      obj.jsl.env.errorInternal('@jslab [FATAL INTERNAL]: ' + e.message);
49    }
50    e.preventDefault();
51  });
52}

53
54 /**
55 * Evaluates a block of code as a string within the JSLAB environment .
56 * @param {String} code The code block to evaluate .
57 * @param {Boolean} [show_output=true] If true , displays the output of the
58 * code block .
59 * @param {String} [jsl_file_name='jslcmdwindow'] The name of the file
60 * context for the code block .
61 */
62 async evalCodeFromMain(code, show_output = true, jsl_file_name = 'jslcmdwindow') {
63   var obj = this;
64
65   this.jsl.onEvaluating();
66   this.jsl.jsl_file_name = jsl_file_name;
67   this.jsl.current_script = jsl_file_name;

```

```
66     this.source_codes = [];
67     this.transformed_codes = [];
68     this.source_maps = [];
69     this.current_source_code;
70     this.current_source_map;
71
72     this.jsl.savePreviousWorkspace();
73     this.jsl.loadPreviousWorkspace();
74
75
76     function onError(err) {
77       obj.jsl.onEvaluated();
78       if(err.name === 'JslabError') {
79         obj.jsl.env.error(err.message, false);
80       } else {
81         obj.rewriteError(err.stack.toString());
82       }
83     }
84
85     try {
86       var data = await this.evalString(code);
87       if(obj.jsl.no_ans === false) {
88         obj.jsl.context.ans = data;
89       }
90       if(show_output && obj.jsl.ignore_output === false) {
91         obj.jsl.env.showAns(prettyPrint(data));
92       }
93       obj.jsl.onEvaluated();
94     } catch(err) {
95       onError(err);
96     }
97   }
98
99 /**
100 * Evaluates a string of code and handles errors and output.
101 * @param {String} code The code string to evaluate.
102 * @returns {*} The result of the evaluated code.
103 */
104 async evalString(code) {
105   var rewrite_result = this.rewriteCode(code);
106   if(rewrite_result) {
107     this.source_maps.push(rewrite_result.map);
108     this.transformed_codes.push(rewrite_result.code);
109     this.source_codes.push(code);
110
111     var current_source_code = this.current_source_code;
112     var current_source_map = this.current_source_map;
113     this.current_source_code = code;
114     this.current_source_map = rewrite_result.map;
115
116     var result = await this.jsl._eval(rewrite_result.code);
117
118     this.current_source_code = current_source_code;
119     this.current_source_map = current_source_map;
120     return result;
121 }
```

```

121     }
122     return false;
123   }
124
125 /**
126 * Executes a script file within the JSLAB environment.
127 * @param {String} script_path The path to the script file to be executed.
128 * @param {Array} [lines] Specifies a range of lines to execute, if provided
129 .
130 * @param {Boolean} [silent=false] If true, suppresses output from the
131   script.
132 */
133 async runScript(script_path, lines, silent = false) {
134   script_path = this.jsl.pathResolve(script_path);
135   if(script_path) {
136     this.jsl.current_script = script_path;
137     this.jsl.jsl_file_name = this.jsl.env.pathBaseName(script_path);
138
139     var script_code = this.jsl.env.readFileSync(script_path);
140     if(script_code === false) {
141       this.jsl.env.error('@runScript: '+language.string(103)+': '+
142         script_path, false);
143     } else {
144       script_code = script_code.toString();
145       if(lines !== undefined) {
146         var code_lines = script_code.split('\n');
147         if(typeof lines === 'number') {
148           if(code_lines.length >= lines) {
149             code_lines[lines-1];
150           } else {
151             this.jsl.env.error('@runScript: '+language.string(104)+'\n '+
152               language.string(105)+': '+script_path, false);
153           }
154         } else {
155           if(code_lines.length >= lines[0] && code_lines.length >= lines[1])
156             {
157               code_lines = code_lines.slice(lines[0]-1, lines[1]-1);
158             } else {
159               this.jsl.env.error('@runScript: '+language.string(104)+'\n '+
160                 language.string(105)+': '+script_path);
161             }
162           }
163         }
164       }
165       return await this.evalString(script_code, !silent);
166     }
167     return false;
168   }
169
170 /**
171 * Re-evaluates the last script file that was executed in the environment.
172 */
173 async runLast() {
174   var cmd;
175   if(this.jsl.last_script_lines !== undefined) {

```

```

170     cmd = `run(' + JSON.stringify(this.jsl.last_script_path) + ', ' + this.
171         jsl.last_script_lines.toString() + '", undefined, true)';
172     } else {
173         cmd = `run(' + JSON.stringify(this.jsl.last_script_path) + ', undefined,
174             false, true)';
175     }
176     await this.evalCodeFromMain(cmd);
177 }
178 /**
179 * Extracts and logs error information from the stack trace.
180 * @param {String} stack The error stack trace.
181 */
182 async rewriteError(stack, on_rewrite = false) {
183     this.jsl._console.log(stack, on_rewrite);
184     const regex_normal = /at\s(.*)\s\(((.*\.\.*?)(\d+):(\d+)\))/;
185     const regex_eval = /eval at evalString\s*\((.*:(\d+):(\d+))$/;
186     const regex_rewrite = /\((\d+):(\d+)\)/;
187     var lines = stack.split('\n');
188     var msg = lines[0];
189
190     if(on_rewrite) {
191         msg = msg.replace(/\((\d+:\d+)/g, '');
192         let matchs = lines[0].match(regex_rewrite);
193         if(matchs) {
194             let line = parseInt(matchs[1]);
195             let column = parseInt(matchs[2]);
196
197             msg += "\n" + language.string(114)+" ";
198             msg += "(" + this.jsl.current_script + ")" " +language.string(112)+" :
199                 + line + ", "+language.string(113)+": " + column;
200         }
201         throw {
202             name: 'JslabError',
203             message: msg
204         };
205     } else {
206         for(let i = 1; i < lines.length; i++) {
207             if(lines[i].includes("eval at evalString ()")) {
208                 msg += "\n" +language.string(114)+" ";
209                 let matchs = lines[i].match(regex_eval);
210                 let line = parseInt(matchs[1]);
211                 let column = parseInt(matchs[2]);
212                 var result = await this.getOriginalPosition(end(this.source_maps),
213                     line, column);
214
215                 msg += "(" + this.jsl.current_script + ")" " +language.string(112)+" :
216                     " + result.line + ", "+language.string(113)+": " + result.
217                         column;
218                 break;
219             } else {
220                 let matchs = lines[i].match(regex_normal);
221                 if(matchs) {
222                     msg += "\n" "+language.string(114)+" ";
223                 }
224             }
225         }
226     }
227 }

```

```

219         let expression = matchs[1];
220         let path = matchs[2];
221         let line = parseInt(matchs[3]);
222         let column = parseInt(matchs[4]);
223         msg += expression + " (" + path + ")" "+language.string(112)+" : " +
224             line + ", "+language.string(113)+" : " + column;
225     }
226   }
227   this.jsl.env.error(msg, false);
228 }
229
230 /**
231 * Extracts the position (line and column) of an expression from an error
232 stack trace, providing context for debugging.
233 * @returns {Array} An array containing the line number, column number, and
234 script path of the expression causing the error.
235 */
236 async getExpressionPosition() {
237   var err = new Error();
238   var stack = err.stack;
239
240   const regex_normal = /at\s(.*)\s\(((.*\\.*?)(:(\\d+):(\\d+))\)/;
241   const regex_eval = /eval at evalString\s*\((.*:(\\d+):(\\d+))\)$/;
242   var lines = stack.split('\n');
243   var line;
244   var script;
245
246   for(let i = 2; i < lines.length; i++) {
247     if(lines[i].includes("eval at evalString ()")) {
248       let matchs = lines[i].match(regex_eval);
249       line = parseInt(matchs[1]);
250       column = parseInt(matchs[2]);
251       script = this.jsl.current_script;
252       break;
253     } else {
254       let matchs = lines[i].match(regex_normal);
255       if(matchs) {
256         line = parseInt(matchs[3]);
257         column = parseInt(matchs[4]);
258         script = matchs[2];
259         break;
260       }
261     }
262   }
263   var result = await this.getOriginalPosition(end(this.source_maps), line,
264                                                 column);
265   return [result.line, result.column, script];
266 }
267 /**
268 * Retrieves the original source position from a source map.
269 * @param {Object} map - The source map object.

```

```

270  * @param {number} line - The line number in the generated code.
271  * @param {number} column - The column number in the generated code.
272  * @returns {Promise<Object>} An object containing the original source
273  * position, including source file, line, and column.
274  */
275  async getOriginalPosition(map, line, column) {
276    this.jsl.override.withoutCheckStop = true;
277    var smc = await new this.jsl.env.SourceMapConsumer(map);
278    this.jsl.override.withoutCheckStop = false;
279    var result = smc.originalPositionFor({ line: line, column: column });
280    smc.destroy();
281    return result;
282  }
283 /**
284  * Extracts the body of a given function as a string.
285  * @param {Function} fun - The function from which to extract the body.
286  * @returns {string|undefined} The body of the function as a string, or 'undefined' if it cannot be extracted.
287  */
288 getFunctionBody(fun) {
289   if(typeof fun == 'function') {
290     const funStr = fun.toString();
291     const bodyMatch = funStr.match(/(\{([\s\S]*\})|)/);
292     if(bodyMatch && bodyMatch[1]) {
293       return bodyMatch[1];
294     }
295   }
296   return undefined;
297 }
298 /**
299  * Rewrites a string of code using Recast and Babel Parser.
300  * @param {String} code The code string to evaluate.
301  * @returns {String} The rewritten code.
302  */
303 rewriteCode(code) {
304   const obj = this;
305
306   // Trick from devtools: Wrap code in parentheses if it's an object literal
307   if(/^\s*\{\/.test(code) && /\}\s*$/ .test(code)) {
308     code = '(' + code + ')';
309   }
310
311   if(config.DEBUG_PRE_TRANSFORMED_CODE) {
312     obj.jsl._console.log(code);
313   }
314
315   // Parse the code into an AST using Recast with Babel parser
316   let ast;
317   try {
318     ast = this.jsl.env.recast.parse(code, {
319       parser: {
320         parse(source) {
321           return obj.jsl.env.babel_parser.parse(source, {
322

```

```

323         sourceType: 'module',
324         plugins: [
325             'jsx',
326             'typescript',
327             'classProperties',
328             'dynamicImport',
329             'optionalChaining',
330             'nullishCoalescingOperator',
331             "@babel/plugin-syntax-top-level-await"
332         ],
333         tolerant: true
334     });
335   },
336   sourceFileName: "source.js"
337 );
338 } catch(err) {
339   this.rewriteError(err.stack, true);
340   return false;
341 }
342
343 // Function to check for forbidden names in patterns
344 function checkPattern(pattern) {
345   if(pattern.type === 'Identifier') {
346     if(config.FORBIDDEN_NAMES.includes(pattern.name)) {
347       throw {
348         name: 'JslabError',
349         message: `${language.string(185)}: ${pattern.name} ${language.
350           string(184)}`,
351       };
352     }
353   } else if(pattern.type === 'ObjectPattern') {
354     pattern.properties.forEach((prop) => {
355       if(prop.type === 'RestElement') {
356         checkPattern(prop.argument);
357       } else {
358         checkPattern(prop.value);
359       }
360     });
361   } else if(pattern.type === 'ArrayPattern') {
362     pattern.elements.forEach((element) => {
363       if(element) checkPattern(element);
364     });
365   } else if(pattern.type === 'RestElement') {
366     checkPattern(pattern.argument);
367   } else if(pattern.type === 'AssignmentPattern') {
368     checkPattern(pattern.left);
369   }
370 }
371
372 // Traverse the AST to check for forbidden names
373 this.jsl.env.recast.types.visit(ast, {
374   visitVariableDeclarator(path) {
375     checkPattern(path.node.id);
376     this.traverse(path);

```



```

429         if (prop.type === 'RestElement') {
430             return b.restElement(transformPatternToContext(prop.argument));
431         }
432         return b.objectProperty(
433             prop.key,
434             transformPatternToContext(prop.value),
435             prop.computed,
436             false
437             );
438         })
439     );
440 } else if (pattern.type === 'ArrayPattern') {
441     return b.arrayPattern(
442         pattern.elements.map((element) => {
443             if (element) {
444                 return transformPatternToContext(element);
445             }
446             return null;
447         });
448     );
449 } else if (pattern.type === 'RestElement') {
450     return b.restElement(transformPatternToContext(pattern.argument));
451 } else if (pattern.type === 'AssignmentPattern') {
452     return b.assignmentPattern(
453         transformPatternToContext(pattern.left),
454         pattern.right
455     );
456 }
457 return pattern;
458 }

459 let tempVarCounter = 0;
460 let functionDepth = 0;

462
463 // Helpers to track function-like scopes
464 function enterFunctionScope() {
465     functionDepth++;
466 }
467 function leaveFunctionScope() {
468     functionDepth--;
469 }

470
471 const visitObj = {
472     // If top-level, after the function declaration, assign it to jsl.context
473     visitFunctionDeclaration(path) {
474         const node = path.node;
475         if (functionDepth === 0 && node.id) {
476             enterFunctionScope();
477             this.traverse(path);
478             leaveFunctionScope();

479             // Insert assignment after the function declaration
480             const funcName = node.id.name;
481             const assignment = b.expressionStatement(
482

```

```

483     b.assignmentExpression(
484         '=',
485         b.memberExpression(
486             b.memberExpression(b.identifier('jsl'), b.identifier('context',
487                 )),
488             b.identifier(funcName)
489         ),
490         b.identifier(funcName)
491     );
492     path.insertAfter(assignment);
493
494     return false;
495 } else {
496     enterFunctionScope();
497     this.traverse(path);
498     leaveFunctionScope();
499     return false;
500 }
501 },
502
503 // Same logic for arrow functions and others - no change required
504 visitFunctionExpression(path) {
505     enterFunctionScope();
506     this.traverse(path);
507     leaveFunctionScope();
508     return false;
509 },
510 visitArrowFunctionExpression(path) {
511     enterFunctionScope();
512     this.traverse(path);
513     leaveFunctionScope();
514     return false;
515 },
516 visitClassMethod(path) {
517     enterFunctionScope();
518     this.traverse(path);
519     leaveFunctionScope();
520     return false;
521 },
522 visitObjectMethod(path) {
523     enterFunctionScope();
524     this.traverse(path);
525     leaveFunctionScope();
526     return false;
527 },
528 visitClassPrivateMethod(path) {
529     enterFunctionScope();
530     this.traverse(path);
531     leaveFunctionScope();
532     return false;
533 },
534
535 // If top-level class declaration, after it assign it to jsl.context
536 visitClassDeclaration(path) {

```

```

537     const node = path.node;
538     if(functionDepth === 0 && node.id) {
539         enterFunctionScope();
540         this.traverse(path);
541         leaveFunctionScope();
542
543         const className = node.id.name;
544         const assignment = b.expressionStatement(
545             b.assignmentExpression(
546                 '=',
547                 b.memberExpression(
548                     b.memberExpression(b.identifier('jsl'), b.identifier('context')),
549                     b.identifier(className)
550                 ),
551                 b.identifier(className)
552             )
553         );
554         path.insertAfter(assignment);
555
556         return false;
557     } else {
558         enterFunctionScope();
559         this.traverse(path);
560         leaveFunctionScope();
561         return false;
562     }
563 },
564
565     visitVariableDeclaration(path) {
566         const node = path.node;
567         const parent = path.parent.node;
568         const parentType = parent.type;
569         const isTopLevel = (functionDepth === 0); // Only transform top-level
570         declarations
571
572         // Handle variable declarations in for of and for in loops only if
573         // at top level.
574         if(
575             isTopLevel &&
576             (parentType === 'ForOfStatement' || parentType === 'ForInStatement')
577             &&
578             parent.left === node
579         ) {
580             if(node.declarations.length !== 1) {
581                 throw new Error("Unexpected multiple declarations in for loop");
582             }
583             // Replace the parent's left with the transformed identifier,
584             // without the var keyword.
585             parent.left = transformPatternToContext(node.declarations[0].id);
586             return false;
587         }
588
589         // Existing logic for top-level declarations:
590         const shouldRewrite =

```

```

587   ((node.kind === 'var') && isTopLevel) ||
588   ((node.kind === 'let' || node.kind === 'const') && parentType ===
589   'Program');

590   if (!shouldRewrite) {
591     this.traverse(path);
592     return false;
593   }

594   const newAssignments = [];

595   node.declarations.forEach((decl) => {
596     if (decl.id.type === 'ArrayPattern') {
597       // Handle ArrayPattern by creating a temporary variable and
598       // individual assignments
599       const tempVarName = `temp${tempCounter++}`;
600       // Create a temporary init variable if needed
601       const tempDecl = b.variableDeclaration('var', [
602         b.variableDeclarator(b.identifier(tempVarName), decl.init)
603       ]);
604       newAssignments.push(tempDecl);

605       // For each element in the ArrayPattern, assign to jsl.context
606       decl.id.elements.forEach((element, index) => {
607         if (element && element.type === 'Identifier') {
608           const transformedLeft = b.memberExpression(
609             b.memberExpression(b.identifier('jsl'), b.identifier(
610               'context')),
611             b.identifier(element.name),
612             false
613           );
614           const rightAccess = b.memberExpression(
615             b.identifier(tempVarName),
616             b.numericLiteral(index),
617             true
618           );
619           newAssignments.push(
620             b.expressionStatement(
621               b.assignmentExpression('=', transformedLeft, rightAccess)
622             )
623           );
624         } else if (element && element.type === 'RestElement') {
625           // Handle RestElement (e.g., ...rest)
626           const transformedLeft = transformPatternToContext(element.
627             argument);
628           const rightSlice = b.callExpression(
629             b.memberExpression(
630               b.identifier(tempVarName),
631               b.identifier('slice'),
632               false
633             ),
634             [b.numericLiteral(index)]
635           );
636           newAssignments.push(
637             b.expressionStatement(

```

```

638             b.assignmentExpression('=', transformedLeft, rightSlice)
639         )
640     );
641   }
642 });
643
644 } else {
645   // For normal patterns or single identifiers
646   const transformedId = transformPatternToContext(decl.id);
647   const assignment = b.expressionStatement(
648     b.assignmentExpression('=', transformedId, decl.init || b.
649       identifier('undefined'))
650   );
651   newAssignments.push(assignment);
652 }
653
654 // Check if this declaration is part of a ForStatement init
655 if(path.parent.node.type === 'ForStatement' && path.parent.node.init
656 === node) {
657   // For a ForStatement init, we need a single expression, not
658   // multiple statements
659   const exprs = [];
660   newAssignments.forEach((stmt) => {
661     if(stmt.type === 'VariableDeclaration' && stmt.declarations.length
662       === 1) {
663       // Convert var temp = init into (temp = init)
664       const declInit = stmt.declarations[0].init || b.identifier(
665         'undefined');
666       exprs.push(b.assignmentExpression('=', b.identifier(stmt.
667         declarations[0].id.name), declInit));
668     } else if(stmt.type === 'ExpressionStatement') {
669       exprs.push(stmt.expression);
670     }
671   });
672
673   let initExpr;
674   if(exprs.length === 1) {
675     initExpr = exprs[0];
676   } else {
677     initExpr = b.sequenceExpression(exprs);
678   }
679
680   path.replace(initExpr);
681   return false;
682 } else {
683   // Normal top-level or block-level declaration: replace with
684   // multiple statements
685   path.replace(...newAssignments);
686   return false;
687 }
688 },
689
690 visitImportDeclaration(path) {
691   const node = path.node;

```

```

686     const newStatements = [];
687
688     node.specifiers.forEach((specifier) => {
689       let requireCall;
690       if(specifier.type === 'ImportDefaultSpecifier') {
691         // const defaultExport = require('module').default;
692         requireCall = b.memberExpression(
693           b.callExpression(b.identifier('require'), [node.source]),
694           b.identifier('default'),
695           false
696         );
697       } else {
698         // const { namedExport } = require('module');
699         requireCall = b.callExpression(b.identifier('require'), [node.
700           source]);
701       }
702
703       const left = b.memberExpression(
704         b.memberExpression(b.identifier('jsl'), b.identifier('context')),
705         b.identifier(specifier.local.name),
706         false
707       );
708
709       const right =
710         specifier.type === 'ImportDefaultSpecifier' ? requireCall
711         : b.memberExpression(requireCall, b.identifier(specifier.imported.
712           name), false);
713
714       newStatements.push(
715         b.expressionStatement(
716           b.assignmentExpression('=', left, right)
717         )
718       );
719     }
720   });
721
722   path.replace(...newStatements);
723   return false;
724 }
725
726 this.jsl.env.recast.types.visit(ast, visitObj);
727
728 // Ensure the value of the last expression is returned
729 const finalBody = ast.program.body;
730 if(finalBody.length > 0) {
731   const lastNode = finalBody[finalBody.length - 1];
732   if(lastNode.type === 'ExpressionStatement') {
733     // Replace it with a return statement
734     finalBody[finalBody.length - 1] = b.returnStatement(lastNode.
735       expression);
736   } else if(lastNode.type !== 'ReturnStatement') {
737     // Not an expression or return statement, so append 'return undefined
738     ;
739     finalBody.push(
740       b.returnStatement(b.identifier('undefined')))
```

```

737         );
738     }
739   } else {
740     // If the body is empty, add 'return undefined;';
741     finalBody.push(
742       b.returnStatement(b.identifier('undefined'))
743     );
744   }
745
746   // Reconstruct the AST with the transformed body
747   const transformedAST = b.program(finalBody);
748
749   // Generate code from the transformed AST with Recast
750   const result = this.jsl.env.recast.print(transformedAST, {
751     sourceMapName: 'transformed.js.map',
752   });
753
754   // Wrap the code in an async IIFE to allow top-level await
755   const transformedCode = `(async () => { ${result.code} })()`;
756
757   if(config.DEBUG_TRANSFORMED_CODE) {
758     obj.jsl._console.log(transformedCode);
759     obj.jsl._console.log(result.map);
760   }
761
762   return { code: transformedCode, map: result.map };
763 }
764 }
765
766 exports.PRDC_JSLAB_EVAL = PRDC_JSLAB_EVAL;

```

Listing 104 - jslab-eval.js

```

1 /**
2  * @file JSLAB library override submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB override submodule.
10 */
11 class PRDC_JSLAB_OVERRIDE {
12
13 /**
14  * Initializes the override submodule, setting up a secure execution
15  * environment by deleting or overriding global properties and methods.
16  * @param {Object} jsl Reference to the main JSLAB object.
17 */
18 constructor(jsl) {
19   var obj = this;
20   this.jsl = jsl;
21
22   this.jsl.builtin_workspace = Object.getOwnPropertyNames(this.jsl.context);
23   this._Module = require('module');

```

```
23 // Overrides
24 this.jsl._console = this.jsl.context.console;
25 this.jsl._eval = this.jsl.context.eval;
26 this.jsl._require = this._Module.prototype.require;
27 this.jsl._requestAnimationFrame = this.jsl.context.requestAnimationFrame.
28     bind(this.context);
29 this.jsl._cancelAnimationFrame = this.jsl.context.cancelAnimationFrame.
30     bind(this.context);
31 this.jsl._setInterval = setInterval.bind(this.jsl.context);
32 this.jsl._clearInterval = clearInterval.bind(this.jsl.context);
33 this.jsl._setTimeout = setTimeout.bind(this.jsl.context);
34 this.jsl._clearTimeout = clearTimeout.bind(this.jsl.context);
35 this.jsl._setImmediate = this.jsl.env.setImmediate;
36 this.jsl._clearImmediate = this.jsl.env.clearImmediate;
37 this.jsl._Promise = this.jsl.context.Promise;
38 if(!global.is_worker) {
39     this.jsl._requestIdleCallback = this.jsl.context.requestIdleCallback.
40         bind(this.context);
41     this.jsl._cancelIdleCallback = this.jsl.context.cancelIdleCallback.bind(
42         this.context);
43 }
44 this.jsl._isNaN = this.jsl.context isNaN;
45
46 // Add toJSON methods to some classes
47 if(typeof Gamepad != 'undefined' && !Gamepad.prototype.toJSON) {
48     Gamepad.prototype.toJSON = function() {
49         return {
50             id: this.id,
51             index: this.index,
52             connected: this.connected,
53             timestamp: this.timestamp,
54             mapping: this.mapping,
55             axes: Array.from(this.axes),
56             buttons: this.buttons.map(button => ({
57                 pressed: button.pressed,
58                 value: button.value
59             }))
60         };
61     };
62 }
63 if(typeof MediaDeviceInfo != 'undefined' && !MediaDeviceInfo.prototype.
64     toJSON) {
65     MediaDeviceInfo.prototype.toJSON = function() {
66         return {
67             deviceId: this.deviceId,
68             kind: this.kind,
69             label: this.label,
70             groupId: this.groupId
71         };
72     };
73 }
74
75 // Assign environment properties to context
76 this.jsl.env.exports.forEach(function(prop) {
```



```

120
121     }
122     obj.jsl.context[prop] = submodules[submodule][prop];
123   });
124
125   Object.getOwnPropertyNames(Object.getPrototypeOf(submodules[submodule]))
126     .forEach(function(prop) {
127       if (!['constructor'].includes(prop)) {
128         if (config.DEBUG_FUN_SHADOW && obj.jsl.context.hasOwnProperty(prop))
129           {
130             obj.jsl._console.log('Shadowing function: ' + prop + ' with
131               submodule ' + submodule);
132           } else if (config.DEBUG_NEW_FUN) {
133             obj.jsl._console.log('Adding function to context: ' + prop + '
134               from submodule ' + submodule);
135           }
136         obj.jsl.context[prop] = submodules[submodule][prop].bind(submodules[
137           submodule]);
138       }
139     });
140   });
141
142   this.jsl.initial_workspace = Object.getOwnPropertyNames(this.jsl.context);
143
144   // Execute override submodule
145   this.execute();
146   Object.getOwnPropertyNames(Object.getPrototypeOf(this)).forEach(function(
147     prop) {
148     if (!['constructor', 'execute'].includes(prop)) {
149       if (config.DEBUG_FUN_SHADOW && obj.jsl.context.hasOwnProperty(prop)) {
150         obj.jsl._console.log('Shadowing function: ' + prop + ' with
151           submodule override');
152       } else if (config.DEBUG_NEW_FUN) {
153         obj.jsl._console.log('Adding function to context: ' + prop + ' from
154           submodule override');
155       }
156     obj.jsl.context[prop] = obj[prop].bind(obj);
157   });
158
159   /**
160    * Executes overrides
161    */
162   execute() {
163     var obj = this;
164
165     this.deleted = {};
166     this.delete_globals = [
167       'eval', 'name', 'closed', 'length', 'frameElement', 'navigator', '

```

```

styleMedia', 'onsearch', 'isSeureContext', 'trustedTypes', ,
onappinstalled', 'onbeforeinstallprompt', 'customElements', 'history
', 'navigation', 'locationbar', 'menubar', 'personalbar', 'onunload'
, 'scheduler', 'chrome', 'scrollbars', 'clientInformation',
'onstorage', 'launchQueue', 'originAgentCluster', 'isSecureContext',
'statusbar', 'toolbar', 'status', 'origin', 'credentialless',
'external', 'screen', 'innerWidth', 'innerHeight', 'scrollX', ,
pageXOffset', 'scrollY', 'pageYOffset', 'visualViewport', 'screenX',
'screenY', 'outerWidth', 'outerHeight', 'screenLeft', 'screenTop',
'onbeforexrselect', 'onabort', 'onbeforeinput', 'onblur', 'oncancel',
, 'oncanplay', 'oncanplaythrough', 'onchange', 'onclick', 'onclose',
'oncontextlost', 'oncontextmenu', 'oncontextrestored', 'oncuechange',
', 'ondblclick', 'ondrag', 'ondragend', 'ondragenter', 'ondragleave',
, 'ondragover', 'ondragstart', 'ondrop', 'ondurationchange',
onemptied', 'onended', 'onfocus', 'onformdata', 'oninput',
'oninvalid', 'onload', 'onloadededata', 'onloadedmetadata',
'onloadstart', 'onmousedown', 'onmouseenter', 'onmouseleave',
'onmousemove', 'onmouseout', 'onmouseover', 'onmouseup',
'onmousewheel', 'onpause', 'onplay', 'onplaying', 'onprogress',
'onratechange', 'onreset', 'onresize', 'onscroll',
'onsecuritypolicyviolation', 'onseeked', 'onseeking', 'onselect',
'onslotchange', 'onstalled', 'onsubmit', 'onsuspend', 'ontimeupdate',
, 'ontoggle', 'onvolumechange', 'onwaiting', 'onwebkitanimationend',
, 'onwebkitanimationiteration', 'onwebkitanimationstart',
'onwebkittransitionend', 'onwheel', 'onauxclick',
, 'ongotpointercapture', 'onlostpointercapture', 'onpointerdown',
, 'onpointermove', 'onpointerrawupdate', 'onpointerup',
'onpointercancel', 'onpointerover', 'onpointerout', 'onpointerenter',
, 'onpointerleave', 'onselectstart', 'onselectionchange',
'onanimationend', 'onanimationiteration', 'onanimationstart',
'ontransitionrun', 'ontransitionstart', 'ontransitionend',
, 'ontransitioncancel', 'onafterprint', 'onbeforeprint',
'onbeforeunload', 'onhashchange', 'onlanguagechange', 'onmessage',
'onmessageerror', 'onoffline', 'ononline', 'onpagehide', 'onpageshow',
, 'onpopstate', 'ondevicemotion', 'ondeviceorientation',
'ondeviceorientationabsolute', 'onbeforematch', 'onbeforetoggle',
, 'onscrollend', 'oncontentvisibilityautostatechange', 'cookieStore',
, 'caches',
];
168
169 // Delete globals
170 this.delete_globals.forEach(function(prop) {
171   obj.deleted[prop] = obj.jsl.env.context[prop];
172   delete obj.jsl.env.context[prop];
173 });
174
175 /**
176 * Custom implementation of console methods to integrate with JSLAB's
177 * display and logging mechanisms.
178 */
179 this.console = {
180   log: function(...arg) {
181     obj.jsl.env.disp(...arg);
182     obj.jsl.no_ans = true;
183     obj.jsl.ignore_output = true;

```

```

183 },
184 warn: function (... arg) {
185     obj.jsl.env.warn(... arg);
186     obj.jsl.no_ans = true;
187     obj.jsl.ignore_output = true;
188 },
189 info: function (... arg) {
190     obj.jsl.env.disp(... arg);
191     obj.jsl.no_ans = true;
192     obj.jsl.ignore_output = true;
193 },
194 error: function (... arg) {
195     obj.jsl.env.error(... arg);
196     obj.jsl.no_ans = true;
197     obj.jsl.ignore_output = true;
198 },
199 trace: function() {
200     obj.jsl.notImplemented();
201 },
202 assert: function(expression, msg) {
203     if (!expression) {
204         obj.jsl.env.error(msg);
205         obj.jsl.no_ans = true;
206         obj.jsl.ignore_output = true;
207     }
208 },
209 table: function (... arg) {
210     obj.jsl.notImplemented();
211 },
212 clear: function() {
213     obj.jsl.basic._clear();
214 },
215 debug: function() {
216     obj.jsl.notImplemented();
217 },
218 dir: function() {
219     obj.jsl.notImplemented();
220 },
221 dirxml: function() {
222     obj.jsl.notImplemented();
223 },
224 time: function() {
225     obj.jsl.notImplemented();
226 },
227 timeLog: function() {
228     obj.jsl.notImplemented();
229 },
230 timeEnd: function() {
231     obj.jsl.notImplemented();
232 },
233 timeStamp: function() {
234     obj.jsl.notImplemented();
235 },
236 count: function() {
237     obj.jsl.notImplemented();

```

```

238 },
239 countreset: function() {
240   obj.jsl.notImplemented();
241 },
242 group: function() {
243   obj.jsl.notImplemented();
244 },
245 groupCollapsed: function() {
246   obj.jsl.notImplemented();
247 },
248 groupEnd: function() {
249   obj.jsl.notImplemented();
250 },
251 profile: function() {
252   obj.jsl.notImplemented();
253 },
254 profileEnd: function() {
255   obj.jsl.notImplemented();
256 }
257 };
258
259 /**
260 * Provides a custom Promise implementation with enhanced functionality to
261 * integrate with JSLAB's execution flow.
262 */
263 this.Promise = class extends this.jsl._Promise {
264   constructor(executor) {
265     if(obj.jsl.basic.checkStopLoop() || !obj.jsl.basic.checkStopLoop()) {
266       obj.jsl.promises_number += 1;
267       obj.jsl.last_promise_id += 1;
268       let promises_id = obj.jsl.last_promise_id;
269       obj.jsl.onStatsChange();
270       let data = super(function(_resolve, _reject) {
271         return executor(function(...args) {
272           obj.jsl.clearPromise(promises_id);
273           _resolve(...args);
274         }, function(...args) {
275           obj.jsl.clearPromise(promises_id);
276           _reject(...args);
277         });
278       });
279       data.loop_stoped = false;
280       obj.jsl.started_promises[promises_id] = data;
281       return data;
282     } else {
283       let data = super(function(resolve, reject) {
284         reject();
285       });
286       return data;
287     }
288
289   // Override the 'then' method
290   then(...args) {
291     if (!this.loop_stoped) {

```

```

292         let newPromise = super.then(...args);
293         return newPromise;
294     }
295     return false;
296 }
297
298 // Override the 'catch' method
299 catch(...args) {
300     if(!this.loop_stoped) {
301         let newPromise = super.catch(...args);
302         return newPromise;
303     }
304     return false;
305 }
306
307 // Override the 'finally' method
308 finally(...args) {
309     if(!this.loop_stoped) {
310         let newPromise = super.finally(...args);
311         return newPromise;
312     }
313     return false;
314 }
315 };
316
317 this._Module.prototype.require = function(...args) {
318     var name = obj.jsl.pathResolve(args[0], this);
319     if(name) {
320         if(!obj.jsl.required_modules.includes(name)) {
321             obj.jsl.required_modules.push(name);
322         }
323         args[0] = name;
324         return obj.jsl._require.apply(this, args);
325     }
326     return false;
327 };
328 }
329
330 /**
331 * Schedules a function to be called on the next animation frame in a non-
332 * blocking manner.
333 * @param {Function} fun The function to call on the next animation frame.
334 * @returns {number} The request ID of the animation frame request.
335 */
336 requestAnimationFrame(fun) {
337     var obj = this;
338     var request_id = this.jsl._requestAnimationFrame(function() {
339         fun();
340         obj.jsl.array.removeElementByValue(obj.jsl.started_animation_frames,
341             request_id);
342         obj.jsl.onStatsChange();
343     });
344     this.jsl.started_animation_frames.push(request_id);
345     this.jsl.onStatsChange();
346     return request_id;

```

```

345   }
346
347  /**
348   * Cancels an animation frame request.
349   * @param {number} request_id The ID of the request to cancel.
350   */
351 cancelAnimationFrame(request_id) {
352   this.jsl._cancelAnimationFrame(request_id);
353   this.jsl.array.removeElementByValue(this.jsl.started_animation_frames,
354   request_id);
355   this.jsl.onStatsChange();
356 }
357 /**
358  * Schedules a function to run during the browser's idle periods in a non-
359  * blocking manner.
360  * @param {Function} fun The function to execute during idle time.
361  * @param {Object} options Optional settings for the idle callback.
362  * @returns {number} The request ID of the idle callback request.
363  */
364 requestIdleCallback(fun, options) {
365   var obj = this;
366   var request_id = this.jsl.requestIdleCallback(function() {
367     fun(options);
368     obj.jsl.array.removeElementByValue(obj.jsl.started_idle_callbacks,
369     request_id);
370     obj.jsl.onStatsChange();
371   });
372   this.jsl.started_idle_callbacks.push(request_id);
373   this.jsl.onStatsChange();
374   return request_id;
375 }
376 /**
377  * Cancels an idle callback request.
378  * @param {number} request_id The ID of the request to cancel.
379  */
380 cancelIdleCallback(request_id) {
381   this.jsl._cancelIdleCallback(request_id);
382   this.jsl.array.removeElementByValue(this.jsl.started_idle_callbacks,
383   request_id);
384   this.jsl.onStatsChange();
385 }
386 /**
387  * Sets a repeated interval in a non-blocking manner.
388  * @param {Function} fun The function to execute at each interval.
389  * @param {number} delay The number of milliseconds between each execution
390  *   of the function.
391  * @returns {number} The interval ID.
392  */
393 setInterval(...arg) {
394   var request_id = this.jsl._setInterval(...arg);
395   this.jsl.started_intervals.push(request_id);
396   this.jsl.onStatsChange();

```

```
395     return request_id;
396 }
397
398 /**
399 * Clears a repeated interval.
400 * @param {number} request_id The ID of the interval to clear.
401 */
402 clearInterval(request_id) {
403     this.jsl._clearInterval(request_id);
404     this.jsl.array.removeElementByValue(this.jsl.started_intervals, request_id
405         );
406     this.jsl.onStatsChange();
407 }
408 /**
409 * Sets a timeout to execute a function once after a delay in a non-blocking
410 * manner.
411 * @param {Function} fun The function to execute after the delay.
412 * @param {number} delay The delay in milliseconds before the function is
413 * executed.
414 * @returns {number} The timeout ID.
415 */
416 setTimeout(fun, delay, ...arg) {
417     var obj = this;
418     var request_id = this.jsl._setTimeout(function() {
419         fun(...arg);
420         obj.jsl.array.removeElementByValue(obj.jsl.started_timeouts, request_id)
421             ;
422         obj.jsl.onStatsChange();
423     }, delay);
424     this.jsl.started_timeouts.push(request_id);
425     this.jsl.onStatsChange();
426     return request_id;
427 }
428 /**
429 * Clears a timeout.
430 * @param {number} request_id The ID of the timeout to clear.
431 */
432 clearTimeout(request_id) {
433     this.jsl._clearTimeout(request_id);
434     this.jsl.array.removeElementByValue(this.jsl.started_timeouts, request_id)
435         ;
436     this.jsl.onStatsChange();
437 }
438 /**
439 * Schedules a function to be executed immediately in a non-blocking manner.
440 * @param {Function} fun The function to execute.
441 * @returns {number} The immediate ID.
442 */
443 setImmediate(fun) {
444     var obj = this;
445     var request_id = this.jsl._setImmediate(function() {
446         fun();
447     });
448 }
```

```

445     obj.jsl.array.removeElementByValue(obj.jsl.started_immediates,
446         request_id);
447     obj.jsl.onStatsChange();
448 });
449 this.jsl.started_immediates.push(request_id);
450 this.jsl.onStatsChange();
451 return request_id;
452 }
453 /**
454 * Clears an immediate.
455 * @param {number} request_id The ID of the immediate to clear.
456 */
457 clearImmediate(request_id) {
458     this._clearImmediate(request_id);
459     this.jsl.array.removeElementByValue(this.jsl.started_immediates,
460         request_id);
461     this.jsl.onStatsChange();
462 }
463
464 exports.PRDC_JSLAB_OVERRIDE = PRDC_JSLAB_OVERRIDE;

```

Listing 105 - jslab-override.js

```

1 /**
2 * @file JSLAB ploter echarts based
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB ploter echarts based.
10 */
11 class PRDC_JSLAB_PLOTER {
12
13 /**
14 * Create ploter object.
15 */
16 constructor() {
17     this.library = 'echarts';
18 }
19
20
21 exports.PRDC_JSLAB_PLOTER = PRDC_JSLAB_PLOTER;

```

Listing 106 - jslab-ploter-echarts.js

```

1 /**
2 * @file JSLAB ploter plotly based
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7

```

```

8  /**
9   * Class for JSLAB ploter plotly based.
10  */
11 class PRDC_JSLAB_PLOTER {
12
13  /**
14   * Initializes the plotter object, setting Plotly as the underlying library
15   * for plotting operations.
16   * @param {Object} js1 - Reference to the JSLAB environment.
17   */
18  constructor(js1) {
19    this.js1 = js1;
20    this.library = 'plotly';
21
22  }
23  /**
24   * Sets the plot for the given figure identifier, displaying data and
25   * configuration.
26   * @param {number} fid - The identifier for the figure to update.
27   */
28  async plot(fid) {
29    if(!this.js1.figures.open_figures.hasOwnProperty(fid)) {
30      this.js1.env.error('@ploter/plot: '+language.string(172));
31      return;
32    }
33
34    var obj = this;
35    var figure = this.js1.figures.open_figures[fid];
36    var plot = figure.plot;
37    var context_plot = figure.context.plot;
38
39    var plot_cont = figure.dom.querySelector('#figure-content .plot-cont');
40    if(!plot_cont) {
41      context_plot.setCont();
42      plot_cont = context_plot.plot_cont;
43    }
44
45    // Plot traces
46    var traces = [];
47    var plot_traces = structuredClone(plot.traces);
48    if(!Array.isArray(plot_traces)) {
49      plot_traces = [plot_traces];
50    }
51    var ci = 0;
52    figure.layout_3d = false;
53
54    plot_traces.forEach(function(trace_options) {
55      if(!figure.layout_3d && (trace_options.hasOwnProperty('z') &&
56          (trace_options.hasOwnProperty('type') && trace_options.type != 'heatmap'))) {
57        figure.layout_3d = true; // It is 3D plot
58      }
59
60      var type;

```

```

60   if(trace_options.hasOwnProperty('type')) {
61     type = trace_options.type;
62   }
63   if(type != 'pie') {
64     var trace = {};
65     trace.type = type;
66
67   Object.keys(trace_options).forEach(function(key) {
68     if(['x', 'y', 'z'].includes(key) && !Array.isArray(trace_options[key]))
69       trace[key] = [trace_options[key]];
70     else if(!['mesh3d'].includes(type) && key == 'color' || key == 'width') {
71       if(!trace.hasOwnProperty('line')) {
72         trace.line = {};
73       }
74       trace.line[key] = trace_options[key];
75     } else {
76       var key_lc = key.toLowerCase();
77       if(trace.hasOwnProperty(key_lc) && typeof trace[key_lc] == 'object')
78         Object.assign(trace[key_lc], trace_options[key]);
79       else {
80         trace[key_lc] = trace_options[key];
81       }
82     }
83   });
84
85   if(!['mesh3d'].includes(type) && (!trace.hasOwnProperty('line') || !
86     trace.line.hasOwnProperty('color'))) {
87     if(!trace.hasOwnProperty('line')) {
88       trace.line = {};
89     }
90     trace.line.color = obj.jsl.color.colororder[ci];
91     ci += 1;
92     trace.cliponaxis = false;
93   }
94   traces.push(trace);
95 } else {
96   traces.push(trace_options);
97 }
98
99 if(figure.layout_3d) {
100   figure.dom.getElementById('figure-menu').className = 'figure-3d';
101   if(!plot.axis_style_val) {
102     plot.axis_style_val = 'equal';
103   }
104 } else {
105   figure.dom.getElementById('figure-menu').className = 'figure-2d';
106   if(!plot.axis_style_val) {
107     plot.axis_style_val = 'normal';
108   }
109 }
110

```

```
111 // Plot config
112 var config = {
113   responsive: true,
114   scrollZoom: true
115 };
116
117 var layout = {};
118
119 await context_plot.newPlot(plot, traces, layout, config);
120 figure.dom.getElementById('rotate-menu').style.display = "none";
121 }
122
123 /**
124 * Updates the layout for the plot associated with the specified figure
125 * identifier.
126 * @param {number} fid - The identifier for the figure whose layout is to be
127 * updated.
128 */
129 updatePlotLayout(fid) {
130   if(!this.jsl.figures.open_figures.hasOwnProperty(fid)) {
131     this.jsl.env.error('@ploter/updatePlotLayout: '+language.string(172));
132     return;
133   }
134   var figure = this.jsl.figures.open_figures[fid];
135   var plot = figure.plot;
136   var context_plot = figure.context.plot;
137   var plot_cont = context_plot.plot_cont;
138
139   // Plot layout
140   var layout = {
141     title: {},
142     autosize: true,
143     plot_bgcolor: "#fff",
144     paper_bgcolor: "#fff",
145     showlegend: plot.traces.length > 1,
146     legend: {
147       x: 0.98,
148       xanchor: 'right',
149       y: 0.98,
150       bgcolor: '#fff',
151       bordercolor: '#000',
152       borderwidth: 1,
153       borderpad: 10
154     },
155     font: {
156       family: 'Roboto',
157       size: 18
158     }
159   };
160
161   if(figure.layout_3d) {
162     Object.assign(layout,
163     {
164       margin: {
165         l: 0,
```

```
164     r: 0,
165     b: 0,
166     t: 0,
167     pad: 0
168   },
169   scene: {
170     xaxis: {
171       showgrid: true,
172       zeroline: false,
173       showline: true,
174       ticks: 'inside',
175       ticklen: 8,
176       linewidth: 1,
177       tickwidth: 1,
178       gridwidth: 0.5,
179       tickcolor: '#000',
180       linecolor: '#000',
181       gridcolor: '#999',
182       mirror: 'ticks',
183       autorange: true,
184       title: {
185         text: 'x [m]'
186       },
187       tickangle: 0
188     },
189     yaxis: {
190       showgrid: true,
191       zeroline: false,
192       showline: true,
193       ticks: 'inside',
194       ticklen: 8,
195       linewidth: 1,
196       tickwidth: 1,
197       gridwidth: 0.5,
198       tickcolor: '#000',
199       linecolor: '#000',
200       gridcolor: '#999',
201       mirror: 'ticks',
202       autorange: true,
203       title: {
204         text: 'y [m]'
205       },
206       tickangle: 0
207     },
208     zaxis: {
209       showgrid: true,
210       zeroline: false,
211       showline: true,
212       ticks: 'inside',
213       ticklen: 8,
214       linewidth: 1,
215       tickwidth: 1,
216       gridwidth: 0.5,
217       tickcolor: '#000',
218       linecolor: '#000',
```

```
219         gridcolor: '#999',
220         mirror: 'ticks',
221         autorange: true,
222         title: {
223             text: 'z [m]'
224         },
225         tickangle: 0
226     },
227     aspectratio: {
228         x: 1, y: 1, z: 1
229     },
230     camera: {
231         projection: { type: 'orthographic' }
232     },
233     dragmode: 'orbit' // "zoom" | "pan" | "orbit"
234     }
235   );
236 } else {
237   Object.assign(layout,
238   {
239     margin: {
240       l: 60,
241       r: 15,
242       b: 60,
243       t: 15,
244       pad: 0
245     },
246     xaxis: {
247       showgrid: true,
248       zeroline: false,
249       showline: true,
250       automargin: true,
251       mirror: 'ticks',
252       ticks: 'inside',
253       ticklen: 8,
254       tickwidth: 0.5,
255       tickcolor: '#000',
256       linecolor: '#000',
257       linewidth: 0.5,
258       exponentformat: 'power'
259     },
260     yaxis: {
261       showgrid: true,
262       zeroline: false,
263       showline: true,
264       automargin: true,
265       mirror: 'ticks',
266       ticks: 'inside',
267       ticklen: 8,
268       tickwidth: 0.5,
269       tickcolor: '#000',
270       linecolor: '#000',
271       linewidth: 0.5,
272       exponentformat: 'power'
273     }
```

```
274         }
275     }
276   );
277 }
278
279 // Options
280 if(plot.options.hasOwnProperty('legend')) {
281   Object.assign(layout.legend, plot.options.legend);
282 }
283 if(plot.options.hasOwnProperty('showLegend')) {
284   layout.showlegend = plot.options.showLegend;
285 }
286 if(plot.options.hasOwnProperty('legendLocation')) {
287   switch(plot.options.legendLocation.toLowerCase()) {
288     case 'north':
289       layout.legend.xanchor = 'center',
290       layout.legend.yanchor = 'top',
291       layout.legend.x = 0.5,
292       layout.legend.y = 0.98;
293       break;
294     case 'south':
295       layout.legend.xanchor = 'center',
296       layout.legend.yanchor = 'bottom',
297       layout.legend.x = 0.5,
298       layout.legend.y = 0.02;
299       break;
300     case 'east':
301       layout.legend.xanchor = 'right',
302       layout.legend.yanchor = 'center',
303       layout.legend.x = 0.98,
304       layout.legend.y = 0.5;
305       break;
306     case 'west':
307       layout.legend.xanchor = 'left',
308       layout.legend.yanchor = 'center',
309       layout.legend.x = 0.02,
310       layout.legend.y = 0.5;
311       break;
312     case 'northeast':
313       layout.legend.xanchor = 'right',
314       layout.legend.yanchor = 'top',
315       layout.legend.x = 0.98,
316       layout.legend.y = 0.98;
317       break;
318     case 'northwest':
319       layout.legend.xanchor = 'left',
320       layout.legend.yanchor = 'top',
321       layout.legend.x = 0.02,
322       layout.legend.y = 0.98;
323       break;
324     case 'southeast':
325       layout.legend.xanchor = 'right',
326       layout.legend.yanchor = 'bottom',
327       layout.legend.x = 0.98,
328       layout.legend.y = 0.02;
```

```
329     break;
330 case 'southwest':
331     layout.legend.xanchor = 'left',
332     layout.legend.yanchor = 'bottom',
333     layout.legend.x = 0.02,
334     layout.legend.y = 0.02;
335     break;
336 case 'northoutside':
337     layout.legend.xanchor = 'center',
338     layout.legend.yanchor = 'bottom',
339     layout.legend.x = 0.5,
340     layout.legend.y = 1.02;
341     break;
342 case 'southoutside':
343     layout.legend.xanchor = 'center',
344     layout.legend.yanchor = 'top',
345     layout.legend.x = 0.5,
346     layout.legend.y = -0.02;
347     break;
348 case 'eastoutside':
349     layout.legend.xanchor = 'left',
350     layout.legend.yanchor = 'center',
351     layout.legend.x = 1.02,
352     layout.legend.y = 0.5;
353     break;
354 case 'westoutside':
355     layout.legend.xanchor = 'right',
356     layout.legend.yanchor = 'center',
357     layout.legend.x = -0.02,
358     layout.legend.y = 0.5;
359     break;
360 case 'northeastoutside':
361     layout.legend.xanchor = 'left',
362     layout.legend.yanchor = 'bottom',
363     layout.legend.x = 1.02,
364     layout.legend.y = 1.02;
365     break;
366 case 'northwestoutside':
367     layout.legend.xanchor = 'right',
368     layout.legend.yanchor = 'bottom',
369     layout.legend.x = -0.02,
370     layout.legend.y = 1.02;
371     break;
372 case 'southeastoutside':
373     layout.legend.xanchor = 'left',
374     layout.legend.yanchor = 'top',
375     layout.legend.x = 1.02,
376     layout.legend.y = -0.02;
377     break;
378 case 'southwestoutside':
379     layout.legend.xanchor = 'right',
380     layout.legend.yanchor = 'top',
381     layout.legend.x = -0.02,
382     layout.legend.y = -0.02;
383     break;
```

```
384     }
385 }
386 if(plot.options.hasOwnProperty('legendOrientation')) {
387   if(plot.options.legendOrientation == 'horizontal' ||
388     plot.options.legendOrientation == 'h') {
389     layout.legend.orientation = 'h';
390   } else {
391     layout.legend.orientation = 'v';
392   }
393 }
394
395 if(figure.layout_3d) {
396   if(plot.options.hasOwnProperty('xlim')) {
397     layout.scene.xaxis.range = plot.options.xlim;
398   }
399   if(plot.options.hasOwnProperty('ylim')) {
400     layout.scene.yaxis.range = plot.options.ylim;
401   }
402   if(plot.options.hasOwnProperty('zlim')) {
403     layout.scene.zaxis.range = plot.options.zlim;
404   }
405 } else {
406   if(plot.options.hasOwnProperty('xlim')) {
407     layout.xaxis.range = plot.options.xlim;
408   }
409   if(plot.options.hasOwnProperty('ylim')) {
410     layout.yaxis.range = plot.options.ylim;
411   }
412 }
413
414 if(plot.options.hasOwnProperty('font')) {
415   layout.font = plot.options.font;
416 }
417 if(plot.options.hasOwnProperty('margin')) {
418   var margin = plot.options.margin;
419   if(margin.hasOwnProperty('top')) {
420     layout.margin.t = margin.top;
421   }
422   if(margin.hasOwnProperty('bottom')) {
423     layout.margin.b = margin.bottom;
424   }
425   if(margin.hasOwnProperty('left')) {
426     layout.margin.l = margin.left;
427   }
428   if(margin.hasOwnProperty('right')) {
429     layout.margin.r = margin.right;
430   }
431 }
432
433 if(plot.hasOwnProperty('title_val')) {
434   layout.title.text = plot.title_val;
435   layout.title.font = {
436     size: 18
437   };
438   layout.margin.t = 40;
```

```

439 }
440
441 if(plot.hasOwnProperty('legend_state_val') &&
442     ['on', 'off', 1, 0].includes(plot.legend_state_val)) {
443     layout.showlegend = plot.legend_state_val === 'on' || plot.
444         legend_state_val;
445 }
446
447 if(figure.layout_3d) {
448     if(plot.hasOwnProperty('xlabel_val')) {
449         layout.scene.xaxis.title = plot.xlabel_val;
450         layout.scene.xaxis.automargin = true;
451         layout.scene.xaxis.standoff = 20;
452     }
453     if(plot.hasOwnProperty('ylabel_val')) {
454         layout.scene.yaxis.title = plot.ylabel_val;
455         layout.scene.yaxis.automargin = true;
456         layout.scene.yaxis.standoff = 20;
457     }
458     if(plot.hasOwnProperty('zlabel_val')) {
459         layout.scene.zaxis.title = plot.zlabel_val;
460         layout.scene.zaxis.automargin = true;
461         layout.scene.zaxis.standoff = 20;
462     }
463 } else {
464     if(plot.hasOwnProperty('xlabel_val')) {
465         layout.xaxis.title = plot.xlabel_val;
466         layout.xaxis.automargin = true;
467         layout.xaxis.standoff = 20;
468     }
469     if(plot.hasOwnProperty('ylabel_val')) {
470         layout.yaxis.title = plot.ylabel_val;
471         layout.yaxis.automargin = true;
472         layout.yaxis.standoff = 20;
473     }
474     if(plot.axis_style_val === 'equal') {
475         layout.yaxis.scaleanchor = "x";
476     } else {
477         layout.yaxis.scaleanchor = false;
478     }
479 }
480 context_plot.relayout(layout);
481 context_plot.resize();
482
483 if(plot.lim_update) {
484     plot.lim_update = false;
485     var layout2 = {};
486     if(figure.layout_3d) {
487         if(plot.hasOwnProperty('xlim_val')) {
488             if(plot.xlim_val[0] === 'auto') {
489                 plot.xlim_val[0] = plot_cont._fullLayout.scene.xaxis.range[0];
490             }
491             if(plot.xlim_val[1] === 'auto') {
492                 plot.xlim_val[1] = plot_cont._fullLayout.scene.xaxis.range[1];
493             }
494         }
495     }
496 }
497
498 if(plot.lim_update) {
499     plot.lim_update = false;
500     var layout2 = {};
501     if(figure.layout_3d) {
502         if(plot.hasOwnProperty('ylim_val')) {
503             if(plot.ylim_val[0] === 'auto') {
504                 plot.ylim_val[0] = plot_cont._fullLayout.scene.yaxis.range[0];
505             }
506             if(plot.ylim_val[1] === 'auto') {
507                 plot.ylim_val[1] = plot_cont._fullLayout.scene.yaxis.range[1];
508             }
509         }
510     }
511 }
512
513 if(plot.lim_update) {
514     plot.lim_update = false;
515     var layout2 = {};
516     if(figure.layout_3d) {
517         if(plot.hasOwnProperty('zlim_val')) {
518             if(plot.zlim_val[0] === 'auto') {
519                 plot.zlim_val[0] = plot_cont._fullLayout.scene.zaxis.range[0];
520             }
521             if(plot.zlim_val[1] === 'auto') {
522                 plot.zlim_val[1] = plot_cont._fullLayout.scene.zaxis.range[1];
523             }
524         }
525     }
526 }
527
528 if(plot.lim_update) {
529     plot.lim_update = false;
530     var layout2 = {};
531     if(figure.layout_3d) {
532         if(plot.hasOwnProperty('colorbar_val')) {
533             if(plot.colorbar_val === 'true') {
534                 plot.colorbar_val = true;
535             }
536             if(plot.colorbar_val === 'false') {
537                 plot.colorbar_val = false;
538             }
539             if(plot.colorbar_val === 'auto') {
540                 plot.colorbar_val = true;
541             }
542             if(plot.colorbar_val === 'none') {
543                 plot.colorbar_val = false;
544             }
545         }
546     }
547 }
548
549 if(plot.lim_update) {
550     plot.lim_update = false;
551     var layout2 = {};
552     if(figure.layout_3d) {
553         if(plot.hasOwnProperty('lighting_val')) {
554             if(plot.lighting_val === 'true') {
555                 plot.lighting_val = true;
556             }
557             if(plot.lighting_val === 'false') {
558                 plot.lighting_val = false;
559             }
560             if(plot.lighting_val === 'auto') {
561                 plot.lighting_val = true;
562             }
563         }
564     }
565 }
566
567 if(plot.lim_update) {
568     plot.lim_update = false;
569     var layout2 = {};
570     if(figure.layout_3d) {
571         if(plot.hasOwnProperty('camera_val')) {
572             if(plot.camera_val === 'true') {
573                 plot.camera_val = true;
574             }
575             if(plot.camera_val === 'false') {
576                 plot.camera_val = false;
577             }
578             if(plot.camera_val === 'auto') {
579                 plot.camera_val = true;
580             }
581         }
582     }
583 }
584
585 if(plot.lim_update) {
586     plot.lim_update = false;
587     var layout2 = {};
588     if(figure.layout_3d) {
589         if(plot.hasOwnProperty('projection_val')) {
590             if(plot.projection_val === 'true') {
591                 plot.projection_val = true;
592             }
593             if(plot.projection_val === 'false') {
594                 plot.projection_val = false;
595             }
596             if(plot.projection_val === 'auto') {
597                 plot.projection_val = true;
598             }
599         }
600     }
601 }
602
603 if(plot.lim_update) {
604     plot.lim_update = false;
605     var layout2 = {};
606     if(figure.layout_3d) {
607         if(plot.hasOwnProperty('backgroundcolor_val')) {
608             if(plot.backgroundcolor_val === 'true') {
609                 plot.backgroundcolor_val = true;
610             }
611             if(plot.backgroundcolor_val === 'false') {
612                 plot.backgroundcolor_val = false;
613             }
614             if(plot.backgroundcolor_val === 'auto') {
615                 plot.backgroundcolor_val = true;
616             }
617         }
618     }
619 }
620
621 if(plot.lim_update) {
622     plot.lim_update = false;
623     var layout2 = {};
624     if(figure.layout_3d) {
625         if(plot.hasOwnProperty('gridcolor_val')) {
626             if(plot.gridcolor_val === 'true') {
627                 plot.gridcolor_val = true;
628             }
629             if(plot.gridcolor_val === 'false') {
630                 plot.gridcolor_val = false;
631             }
632             if(plot.gridcolor_val === 'auto') {
633                 plot.gridcolor_val = true;
634             }
635         }
636     }
637 }
638
639 if(plot.lim_update) {
640     plot.lim_update = false;
641     var layout2 = {};
642     if(figure.layout_3d) {
643         if(plot.hasOwnProperty('gridwidth_val')) {
644             if(plot.gridwidth_val === 'true') {
645                 plot.gridwidth_val = true;
646             }
647             if(plot.gridwidth_val === 'false') {
648                 plot.gridwidth_val = false;
649             }
650             if(plot.gridwidth_val === 'auto') {
651                 plot.gridwidth_val = true;
652             }
653         }
654     }
655 }
656
657 if(plot.lim_update) {
658     plot.lim_update = false;
659     var layout2 = {};
660     if(figure.layout_3d) {
661         if(plot.hasOwnProperty('griddash_val')) {
662             if(plot.griddash_val === 'true') {
663                 plot.griddash_val = true;
664             }
665             if(plot.griddash_val === 'false') {
666                 plot.griddash_val = false;
667             }
668             if(plot.griddash_val === 'auto') {
669                 plot.griddash_val = true;
670             }
671         }
672     }
673 }
674
675 if(plot.lim_update) {
676     plot.lim_update = false;
677     var layout2 = {};
678     if(figure.layout_3d) {
679         if(plot.hasOwnProperty('gridcolor2_val')) {
680             if(plot.gridcolor2_val === 'true') {
681                 plot.gridcolor2_val = true;
682             }
683             if(plot.gridcolor2_val === 'false') {
684                 plot.gridcolor2_val = false;
685             }
686             if(plot.gridcolor2_val === 'auto') {
687                 plot.gridcolor2_val = true;
688             }
689         }
690     }
691 }
692
693 if(plot.lim_update) {
694     plot.lim_update = false;
695     var layout2 = {};
696     if(figure.layout_3d) {
697         if(plot.hasOwnProperty('gridwidth2_val')) {
698             if(plot.gridwidth2_val === 'true') {
699                 plot.gridwidth2_val = true;
700             }
701             if(plot.gridwidth2_val === 'false') {
702                 plot.gridwidth2_val = false;
703             }
704             if(plot.gridwidth2_val === 'auto') {
705                 plot.gridwidth2_val = true;
706             }
707         }
708     }
709 }
710
711 if(plot.lim_update) {
712     plot.lim_update = false;
713     var layout2 = {};
714     if(figure.layout_3d) {
715         if(plot.hasOwnProperty('griddash2_val')) {
716             if(plot.griddash2_val === 'true') {
717                 plot.griddash2_val = true;
718             }
719             if(plot.griddash2_val === 'false') {
720                 plot.griddash2_val = false;
721             }
722             if(plot.griddash2_val === 'auto') {
723                 plot.griddash2_val = true;
724             }
725         }
726     }
727 }
728
729 if(plot.lim_update) {
730     plot.lim_update = false;
731     var layout2 = {};
732     if(figure.layout_3d) {
733         if(plot.hasOwnProperty('gridcolor3_val')) {
734             if(plot.gridcolor3_val === 'true') {
735                 plot.gridcolor3_val = true;
736             }
737             if(plot.gridcolor3_val === 'false') {
738                 plot.gridcolor3_val = false;
739             }
740             if(plot.gridcolor3_val === 'auto') {
741                 plot.gridcolor3_val = true;
742             }
743         }
744     }
745 }
746
747 if(plot.lim_update) {
748     plot.lim_update = false;
749     var layout2 = {};
750     if(figure.layout_3d) {
751         if(plot.hasOwnProperty('gridwidth3_val')) {
752             if(plot.gridwidth3_val === 'true') {
753                 plot.gridwidth3_val = true;
754             }
755             if(plot.gridwidth3_val === 'false') {
756                 plot.gridwidth3_val = false;
757             }
758             if(plot.gridwidth3_val === 'auto') {
759                 plot.gridwidth3_val = true;
760             }
761         }
762     }
763 }
764
765 if(plot.lim_update) {
766     plot.lim_update = false;
767     var layout2 = {};
768     if(figure.layout_3d) {
769         if(plot.hasOwnProperty('griddash3_val')) {
770             if(plot.griddash3_val === 'true') {
771                 plot.griddash3_val = true;
772             }
773             if(plot.griddash3_val === 'false') {
774                 plot.griddash3_val = false;
775             }
776             if(plot.griddash3_val === 'auto') {
777                 plot.griddash3_val = true;
778             }
779         }
780     }
781 }
782
783 if(plot.lim_update) {
784     plot.lim_update = false;
785     var layout2 = {};
786     if(figure.layout_3d) {
787         if(plot.hasOwnProperty('gridcolor4_val')) {
788             if(plot.gridcolor4_val === 'true') {
789                 plot.gridcolor4_val = true;
790             }
791             if(plot.gridcolor4_val === 'false') {
792                 plot.gridcolor4_val = false;
793             }
794             if(plot.gridcolor4_val === 'auto') {
795                 plot.gridcolor4_val = true;
796             }
797         }
798     }
799 }
800
801 if(plot.lim_update) {
802     plot.lim_update = false;
803     var layout2 = {};
804     if(figure.layout_3d) {
805         if(plot.hasOwnProperty('gridwidth4_val')) {
806             if(plot.gridwidth4_val === 'true') {
807                 plot.gridwidth4_val = true;
808             }
809             if(plot.gridwidth4_val === 'false') {
810                 plot.gridwidth4_val = false;
811             }
812             if(plot.gridwidth4_val === 'auto') {
813                 plot.gridwidth4_val = true;
814             }
815         }
816     }
817 }
818
819 if(plot.lim_update) {
820     plot.lim_update = false;
821     var layout2 = {};
822     if(figure.layout_3d) {
823         if(plot.hasOwnProperty('griddash4_val')) {
824             if(plot.griddash4_val === 'true') {
825                 plot.griddash4_val = true;
826             }
827             if(plot.griddash4_val === 'false') {
828                 plot.griddash4_val = false;
829             }
830             if(plot.griddash4_val === 'auto') {
831                 plot.griddash4_val = true;
832             }
833         }
834     }
835 }
836
837 if(plot.lim_update) {
838     plot.lim_update = false;
839     var layout2 = {};
840     if(figure.layout_3d) {
841         if(plot.hasOwnProperty('gridcolor5_val')) {
842             if(plot.gridcolor5_val === 'true') {
843                 plot.gridcolor5_val = true;
844             }
845             if(plot.gridcolor5_val === 'false') {
846                 plot.gridcolor5_val = false;
847             }
848             if(plot.gridcolor5_val === 'auto') {
849                 plot.gridcolor5_val = true;
850             }
851         }
852     }
853 }
854
855 if(plot.lim_update) {
856     plot.lim_update = false;
857     var layout2 = {};
858     if(figure.layout_3d) {
859         if(plot.hasOwnProperty('gridwidth5_val')) {
860             if(plot.gridwidth5_val === 'true') {
861                 plot.gridwidth5_val = true;
862             }
863             if(plot.gridwidth5_val === 'false') {
864                 plot.gridwidth5_val = false;
865             }
866             if(plot.gridwidth5_val === 'auto') {
867                 plot.gridwidth5_val = true;
868             }
869         }
870     }
871 }
872
873 if(plot.lim_update) {
874     plot.lim_update = false;
875     var layout2 = {};
876     if(figure.layout_3d) {
877         if(plot.hasOwnProperty('griddash5_val')) {
878             if(plot.griddash5_val === 'true') {
879                 plot.griddash5_val = true;
880             }
881             if(plot.griddash5_val === 'false') {
882                 plot.griddash5_val = false;
883             }
884             if(plot.griddash5_val === 'auto') {
885                 plot.griddash5_val = true;
886             }
887         }
888     }
889 }
890
891 if(plot.lim_update) {
892     plot.lim_update = false;
893     var layout2 = {};
894     if(figure.layout_3d) {
895         if(plot.hasOwnProperty('gridcolor6_val')) {
896             if(plot.gridcolor6_val === 'true') {
897                 plot.gridcolor6_val = true;
898             }
899             if(plot.gridcolor6_val === 'false') {
900                 plot.gridcolor6_val = false;
901             }
902             if(plot.gridcolor6_val === 'auto') {
903                 plot.gridcolor6_val = true;
904             }
905         }
906     }
907 }
908
909 if(plot.lim_update) {
910     plot.lim_update = false;
911     var layout2 = {};
912     if(figure.layout_3d) {
913         if(plot.hasOwnProperty('gridwidth6_val')) {
914             if(plot.gridwidth6_val === 'true') {
915                 plot.gridwidth6_val = true;
916             }
917             if(plot.gridwidth6_val === 'false') {
918                 plot.gridwidth6_val = false;
919             }
920             if(plot.gridwidth6_val === 'auto') {
921                 plot.gridwidth6_val = true;
922             }
923         }
924     }
925 }
926
927 if(plot.lim_update) {
928     plot.lim_update = false;
929     var layout2 = {};
930     if(figure.layout_3d) {
931         if(plot.hasOwnProperty('griddash6_val')) {
932             if(plot.griddash6_val === 'true') {
933                 plot.griddash6_val = true;
934             }
935             if(plot.griddash6_val === 'false') {
936                 plot.griddash6_val = false;
937             }
938             if(plot.griddash6_val === 'auto') {
939                 plot.griddash6_val = true;
940             }
941         }
942     }
943 }
944
945 if(plot.lim_update) {
946     plot.lim_update = false;
947     var layout2 = {};
948     if(figure.layout_3d) {
949         if(plot.hasOwnProperty('gridcolor7_val')) {
950             if(plot.gridcolor7_val === 'true') {
951                 plot.gridcolor7_val = true;
952             }
953             if(plot.gridcolor7_val === 'false') {
954                 plot.gridcolor7_val = false;
955             }
956             if(plot.gridcolor7_val === 'auto') {
957                 plot.gridcolor7_val = true;
958             }
959         }
960     }
961 }
962
963 if(plot.lim_update) {
964     plot.lim_update = false;
965     var layout2 = {};
966     if(figure.layout_3d) {
967         if(plot.hasOwnProperty('gridwidth7_val')) {
968             if(plot.gridwidth7_val === 'true') {
969                 plot.gridwidth7_val = true;
970             }
971             if(plot.gridwidth7_val === 'false') {
972                 plot.gridwidth7_val = false;
973             }
974             if(plot.gridwidth7_val === 'auto') {
975                 plot.gridwidth7_val = true;
976             }
977         }
978     }
979 }
980
981 if(plot.lim_update) {
982     plot.lim_update = false;
983     var layout2 = {};
984     if(figure.layout_3d) {
985         if(plot.hasOwnProperty('griddash7_val')) {
986             if(plot.griddash7_val === 'true') {
987                 plot.griddash7_val = true;
988             }
989             if(plot.griddash7_val === 'false') {
990                 plot.griddash7_val = false;
991             }
992             if(plot.griddash7_val === 'auto') {
993                 plot.griddash7_val = true;
994             }
995         }
996     }
997 }
998
999 if(plot.lim_update) {
1000    plot.lim_update = false;
1001    var layout2 = {};
1002    if(figure.layout_3d) {
1003        if(plot.hasOwnProperty('gridcolor8_val')) {
1004            if(plot.gridcolor8_val === 'true') {
1005                plot.gridcolor8_val = true;
1006            }
1007            if(plot.gridcolor8_val === 'false') {
1008                plot.gridcolor8_val = false;
1009            }
1010            if(plot.gridcolor8_val === 'auto') {
1011                plot.gridcolor8_val = true;
1012            }
1013        }
1014    }
1015 }
1016
1017 if(plot.lim_update) {
1018    plot.lim_update = false;
1019    var layout2 = {};
1020    if(figure.layout_3d) {
1021        if(plot.hasOwnProperty('gridwidth8_val')) {
1022            if(plot.gridwidth8_val === 'true') {
1023                plot.gridwidth8_val = true;
1024            }
1025            if(plot.gridwidth8_val === 'false') {
1026                plot.gridwidth8_val = false;
1027            }
1028            if(plot.gridwidth8_val === 'auto') {
1029                plot.gridwidth8_val = true;
1030            }
1031        }
1032    }
1033 }
1034
1035 if(plot.lim_update) {
1036    plot.lim_update = false;
1037    var layout2 = {};
1038    if(figure.layout_3d) {
1039        if(plot.hasOwnProperty('griddash8_val')) {
1040            if(plot.griddash8_val === 'true') {
1041                plot.griddash8_val = true;
1042            }
1043            if(plot.griddash8_val === 'false') {
1044                plot.griddash8_val = false;
1045            }
1046            if(plot.griddash8_val === 'auto') {
1047                plot.griddash8_val = true;
1048            }
1049        }
1050    }
1051 }
1052
1053 if(plot.lim_update) {
1054    plot.lim_update = false;
1055    var layout2 = {};
1056    if(figure.layout_3d) {
1057        if(plot.hasOwnProperty('gridcolor9_val')) {
1058            if(plot.gridcolor9_val === 'true') {
1059                plot.gridcolor9_val = true;
1060            }
1061            if(plot.gridcolor9_val === 'false') {
1062                plot.gridcolor9_val = false;
1063            }
1064            if(plot.gridcolor9_val === 'auto') {
1065                plot.gridcolor9_val = true;
1066            }
1067        }
1068    }
1069 }
1070
1071 if(plot.lim_update) {
1072    plot.lim_update = false;
1073    var layout2 = {};
1074    if(figure.layout_3d) {
1075        if(plot.hasOwnProperty('gridwidth9_val')) {
1076            if(plot.gridwidth9_val === 'true') {
1077                plot.gridwidth9_val = true;
1078            }
1079            if(plot.gridwidth9_val === 'false') {
1080                plot.gridwidth9_val = false;
1081            }
1082            if(plot.gridwidth9_val === 'auto') {
1083                plot.gridwidth9_val = true;
1084            }
1085        }
1086    }
1087 }
1088
1089 if(plot.lim_update) {
1090    plot.lim_update = false;
1091    var layout2 = {};
1092    if(figure.layout_3d) {
1093        if(plot.hasOwnProperty('griddash9_val')) {
1094            if(plot.griddash9_val === 'true') {
1095                plot.griddash9_val = true;
1096            }
1097            if(plot.griddash9_val === 'false') {
1098                plot.griddash9_val = false;
1099            }
1100            if(plot.griddash9_val === 'auto') {
1101                plot.griddash9_val = true;
1102            }
1103        }
1104    }
1105 }
1106
1107 if(plot.lim_update) {
1108    plot.lim_update = false;
1109    var layout2 = {};
1110    if(figure.layout_3d) {
1111        if(plot.hasOwnProperty('gridcolor10_val')) {
1112            if(plot.gridcolor10_val === 'true') {
1113                plot.gridcolor10_val = true;
1114            }
1115            if(plot.gridcolor10_val === 'false') {
1116                plot.gridcolor10_val = false;
1117            }
1118            if(plot.gridcolor10_val === 'auto') {
1119                plot.gridcolor10_val = true;
1120            }
1121        }
1122    }
1123 }
1124
1125 if(plot.lim_update) {
1126    plot.lim_update = false;
1127    var layout2 = {};
1128    if(figure.layout_3d) {
1129        if(plot.hasOwnProperty('gridwidth10_val')) {
1130            if(plot.gridwidth10_val === 'true') {
1131                plot.gridwidth10_val = true;
1132            }
1133            if(plot.gridwidth10_val === 'false') {
1134                plot.gridwidth10_val = false;
1135            }
1136            if(plot.gridwidth10_val === 'auto') {
1137                plot.gridwidth10_val = true;
1138            }
1139        }
1140    }
1141 }
1142
1143 if(plot.lim_update) {
1144    plot.lim_update = false;
1145    var layout2 = {};
1146    if(figure.layout_3d) {
1147        if(plot.hasOwnProperty('griddash10_val')) {
1148            if(plot.griddash10_val === 'true') {
1149                plot.griddash10_val = true;
1150            }
1151            if(plot.griddash10_val === 'false') {
1152                plot.griddash10_val = false;
1153            }
1154            if(plot.griddash10_val === 'auto') {
1155                plot.griddash10_val = true;
1156            }
1157        }
1158    }
1159 }
1160
1161 if(plot.lim_update) {
1162    plot.lim_update = false;
1163    var layout2 = {};
1164    if(figure.layout_3d) {
1165        if(plot.hasOwnProperty('gridcolor11_val')) {
1166            if(plot.gridcolor11_val === 'true') {
1167                plot.gridcolor11_val = true;
1168            }
1169            if(plot.gridcolor11_val === 'false') {
1170                plot.gridcolor11_val = false;
1171            }
1172            if(plot.gridcolor11_val === 'auto') {
1173                plot.gridcolor11_val = true;
1174            }
1175        }
1176    }
1177 }
1178
1179 if(plot.lim_update) {
1180    plot.lim_update = false;
1181    var layout2 = {};
1182    if(figure.layout_3d) {
1183        if(plot.hasOwnProperty('gridwidth11_val')) {
1184            if(plot.gridwidth11_val === 'true') {
1185                plot.gridwidth11_val = true;
1186            }
1187            if(plot.gridwidth11_val === 'false') {
1188                plot.gridwidth11_val = false;
1189            }
1190            if(plot.gridwidth11_val === 'auto') {
1191                plot.gridwidth11_val = true;
1192            }
1193        }
1194    }
1195 }
1196
1197 if(plot.lim_update) {
1198    plot.lim_update = false;
1199    var layout2 = {};
1200    if(figure.layout_3d) {
1201        if(plot.hasOwnProperty('griddash11_val')) {
1202            if(plot.griddash11_val === 'true') {
1203                plot.griddash11_val = true;
1204            }
1205            if(plot.griddash11_val === 'false') {
1206                plot.griddash11_val = false;
1207            }
1208            if(plot.griddash11_val === 'auto') {
1209                plot.griddash11_val = true;
1210            }
1211        }
1212    }
1213 }
1214
1215 if(plot.lim_update) {
1216    plot.lim_update = false;
1217    var layout2 = {};
1218    if(figure.layout_3d) {
1219        if(plot.hasOwnProperty('gridcolor12_val')) {
1220            if(plot.gridcolor12_val === 'true') {
1221                plot.gridcolor12_val = true;
1222            }
1223            if(plot.gridcolor12_val === 'false') {
1224                plot.gridcolor12_val = false;
1225            }
1226            if(plot.gridcolor12_val === 'auto') {
1227                plot.gridcolor12_val = true;
1228            }
1229        }
1230    }
1231 }
1232
1233 if(plot.lim_update) {
1234    plot.lim_update = false;
1235    var layout2 = {};
1236    if(figure.layout_3d) {
1237        if(plot.hasOwnProperty('gridwidth12_val')) {
1238            if(plot.gridwidth12_val === 'true') {
1239                plot.gridwidth12_val = true;
1240            }
1241            if(plot.gridwidth12_val === 'false') {
1242                plot.gridwidth12_val = false;
1243            }
1244            if(plot.gridwidth12_val === 'auto') {
1245                plot.gridwidth12_val = true;
1246            }
1247        }
1248    }
1249 }
1250
1251 if(plot.lim_update) {
1252    plot.lim_update = false;
1253    var layout2 = {};
1254    if(figure.layout_3d) {
1255        if(plot.hasOwnProperty('griddash12_val')) {
1256            if(plot.griddash12_val === 'true') {
1257                plot.griddash12_val = true;
1258            }
1259            if(plot.griddash12_val === 'false') {
1260                plot.griddash12_val = false;
1261            }
1262            if(plot.griddash12_val === 'auto') {
1263                plot.griddash12_val = true;
1264            }
1265        }
1266    }
1267 }
1268
1269 if(plot.lim_update) {
1270    plot.lim_update = false;
1271    var layout2 = {};
1272    if(figure.layout_3d) {
1273        if(plot.hasOwnProperty('gridcolor13_val')) {
1274            if(plot.gridcolor13_val === 'true') {
1275                plot.gridcolor13_val = true;
1276            }
1277            if(plot.gridcolor13_val === 'false') {
1278                plot.gridcolor13_val = false;
1279            }
1280            if(plot.gridcolor13_val === 'auto') {
1281                plot.gridcolor13_val = true;
1282            }
1283        }
1284    }
1285 }
1286
1287 if(plot.lim_update) {
1288    plot.lim_update = false;
1289    var layout2 = {};
1290    if(figure.layout_3d) {
1291        if(plot.hasOwnProperty('gridwidth13_val')) {
1292            if(plot.gridwidth13_val === 'true') {
1293                plot.gridwidth13_val = true;
1294            }
1295            if(plot.gridwidth13_val === 'false') {
1296                plot.gridwidth13_val = false;
1297            }
1298            if(plot.gridwidth13_val === 'auto') {
1299                plot.gridwidth13_val = true;
1300            }
1301        }
1302    }
1303 }
1304
1305 if(plot.lim_update) {
1306    plot.lim_update = false;
1307    var layout2 = {};
1308    if(figure.layout_3d) {
1309        if(plot.hasOwnProperty('griddash13_val')) {
1310            if(plot.griddash13_val === 'true') {
1311                plot.griddash13_val = true;
1312            }
1313            if(plot.griddash13_val === 'false') {
1314                plot.griddash13_val = false;
1315            }
1316            if(plot.griddash13_val === 'auto') {
1317                plot.griddash13_val = true;
1318            }
1319        }
1320    }
1321 }
1322
1323 if(plot.lim_update) {
1324    plot.lim_update = false;
1325    var layout2 = {};
1326    if(figure.layout_3d) {
1327        if(plot.hasOwnProperty('gridcolor14_val')) {
1328            if(plot.gridcolor14_val === 'true') {
1329                plot.gridcolor14_val = true;
1330            }
1331            if(plot.gridcolor14_val === 'false') {
1332                plot.gridcolor14_val = false;
1333            }
1334            if(plot.gridcolor14_val === 'auto') {
1335                plot.gridcolor14_val = true;
1336            }
1337        }
1338    }
1339 }
1340
1341 if(plot.lim_update) {
1342    plot.lim_update = false;
1343    var layout2 = {};
1344    if(figure.layout_3d) {
1345        if(plot.hasOwnProperty('gridwidth14_val')) {
1346            if(plot.gridwidth14_val === 'true') {
1347                plot.gridwidth14_val = true;
1348            }
1349            if(plot.gridwidth14_val === 'false') {
1350                plot.gridwidth14_val = false;
1351            }
1352            if(plot.gridwidth14_val === 'auto') {
1353                plot.gridwidth14_val = true;
1354            }
1355        }
1356    }
1357 }
1358
1359 if(plot.lim_update) {
1360    plot.lim_update = false;
1361    var layout2 = {};
1362    if(figure.layout_3d) {
1363        if(plot.hasOwnProperty('griddash14_val')) {
1364            if(plot.griddash14_val === 'true') {
1365                plot.griddash14_val = true;
1366            }
1367            if(plot.griddash14_val === 'false') {
1368                plot.griddash14_val = false;
1369            }
1370            if(plot.griddash14_val === 'auto') {
1371                plot.griddash14_val = true;
1372            }
1373        }
1374    }
1375 }
1376
1377 if(plot.lim_update) {
1378    plot.lim_update = false;
1379    var layout2 = {};
1380    if(figure.layout_3d) {
1381        if(plot.hasOwnProperty('gridcolor15_val')) {
1382            if(plot.gridcolor15_val === 'true') {
1383                plot.gridcolor15_val = true;
1384            }
1385            if(plot.gridcolor15_val === 'false') {
1386                plot.gridcolor15_val = false;
1387            }
1388            if(plot.gridcolor15_val === 'auto') {
1389                plot.gridcolor15_val = true;
1390            }
1391        }
1392    }
1393 }
1394
1395 if(plot.lim_update) {
1396    plot.lim_update = false;
1397    var layout2 = {};
1398    if(figure.layout_3d) {
1399        if(plot.hasOwnProperty('gridwidth15_val')) {
1400            if(plot.gridwidth15_val === 'true') {
1401                plot.gridwidth15_val = true;
1402            }
1403            if(plot.gridwidth15_val === 'false') {
1404                plot.gridwidth15_val = false;
1405            }
1406            if(plot.gridwidth15_val === 'auto') {
1407                plot.gridwidth15_val = true;
1408            }
1409        }
1410    }
1411 }
1412
1413 if(plot.lim_update) {
1414    plot.lim_update = false;
1415    var layout2 = {};
1416    if(figure.layout_3d) {
1417        if(plot.hasOwnProperty('griddash15_val')) {
1418            if(plot.griddash15_val === 'true') {
1419                plot.griddash15_val = true;
1420            }
1421            if(plot.griddash15_val === 'false') {
1422                plot.griddash15_val = false;
1423            }
1424            if(plot.griddash15_val === 'auto') {
1425                plot.griddash15_val = true;
1426            }
1427        }
1428    }
1429 }
1430
1431 if(plot.lim_update) {
1432    plot.lim_update = false;
1433    var layout2 = {};
1434    if(figure.layout_3d) {
1435        if(plot.hasOwnProperty('gridcolor16_val')) {
1436            if(plot.gridcolor16_val === 'true') {
1437                plot.gridcolor16_val = true;
1438            }
1439            if(plot.gridcolor16_val === 'false') {
1440                plot.gridcolor16_val = false;
1441            }
1442            if(plot.gridcolor16_val === 'auto') {
1443                plot.gridcolor16_val = true;
1444            }
1445        }
1446    }
1447 }
1448
1449 if(plot.lim_update) {
1450    plot.lim_update = false;
1451    var layout2 = {};
1452    if(figure.layout_3d) {
1453        if(plot.hasOwnProperty('gridwidth16_val')) {
1454            if(plot.gridwidth16_val === 'true') {
1455                plot.gridwidth16_val = true;
1456            }
1457            if(plot.gridwidth16_val === 'false') {
1458                plot.gridwidth16_val = false;
1459            }
1460            if(plot.gridwidth16_val === 'auto') {
1461                plot.gridwidth16_val = true;
1462            }
1463        }
1464    }
1465 }
1466
1467 if(plot.lim_update) {
1468    plot.lim_update = false;
1469    var layout2 = {};
1470    if(figure.layout_3d) {
1471        if(plot.hasOwnProperty('griddash16_val')) {
1472            if(plot.griddash16_val === 'true') {
1473                plot.griddash16_val = true;
1474            }
1475            if(plot.griddash16_val === 'false') {
1476                plot.griddash16_val = false;
1477            }
1478            if(plot.griddash16_val === 'auto') {
1479                plot.griddash16_val = true;
1480            }
1481        }
1482    }
1483 }
1484
1485 if(plot.lim_update) {
1486    plot.lim_update = false;
1487    var layout2 = {};
1488    if(figure.layout_3d) {
1489        if(plot.hasOwnProperty('gridcolor17_val')) {
1490            if(plot.gridcolor17_val === 'true') {
1491                plot.gridcolor17_val = true;
1492            }
1493            if(plot.gridcolor17_val === 'false') {
1494                plot.gridcolor17_val = false;
1495            }
1496            if(plot.gridcolor17_val === 'auto') {
1497                plot.gridcolor17_val = true;
1498            }
1499        }
1500    }
1501 }
1502
1503 if(plot.lim_update) {
1504    plot.lim_update = false;
1505    var layout
```

```

493
494     }
495     layout2[ 'scene.xaxis.range' ] = plot.xlim_val;
496   }
497   if( plot.hasOwnProperty('ylim_val')) {
498     if( plot.ylim_val[0] === 'auto') {
499       plot.ylim_val[0] = plot_cont._fullLayout.scene.yaxis.range[0];
500     }
501     if( plot.ylim_val[1] === 'auto') {
502       plot.ylim_val[1] = plot_cont._fullLayout.scene.yaxis.range[1];
503     }
504     layout2[ 'scene.yaxis.range' ] = plot.ylim_val;
505   }
506   if( plot.hasOwnProperty('zlim_val')) {
507     if( plot.zlim_val[0] === 'auto') {
508       plot.zlim_val[0] = plot_cont._fullLayout.scene.xaxis.range[0];
509     }
510     if( plot.zlim_val[1] === 'auto') {
511       plot.zlim_val[1] = plot_cont._fullLayout.scene.xaxis.range[1];
512     }
513     layout2[ 'scene.xaxis.range' ] = plot.zlim_val;
514   } else {
515     if( plot.hasOwnProperty('xlim_val')) {
516       if( plot.xlim_val[0] === 'auto') {
517         plot.xlim_val[0] = plot_cont._fullLayout.xaxis.range[0];
518       }
519       if( plot.xlim_val[1] === 'auto') {
520         plot.xlim_val[1] = plot_cont._fullLayout.xaxis.range[1];
521       }
522       layout2[ 'xaxis.range' ] = plot.xlim_val;
523     }
524     if( plot.hasOwnProperty('ylim_val')) {
525       if( plot.ylim_val[0] === 'auto') {
526         plot.ylim_val[0] = plot_cont._fullLayout.yaxis.range[0];
527       }
528       if( plot.ylim_val[1] === 'auto') {
529         plot.ylim_val[1] = plot_cont._fullLayout.yaxis.range[1];
530       }
531       layout2[ 'yaxis.range' ] = plot.ylim_val;
532     }
533   }
534   context_plot.relayout(layout2);
535   context_plot.resize();
536 }
537
538 if( figure.layout_3d) {
539   var layout3 = {};
540
541   // Equal axes
542   var xRange = plot_cont._fullLayout.scene.xaxis.range;
543   if( plot.hasOwnProperty('xlim_val')) {
544     xRange = plot.xlim_val;
545     layout3[ 'scene.xaxis.range' ] = plot.xlim_val;
546   }
547   var yRange = plot_cont._fullLayout.scene.yaxis.range;

```

```

548 if (plot .hasOwnProperty( 'ylim_val' )) {
549   yRange = plot .ylim_val;
550   layout3[ 'scene .yaxis .range' ] = plot .ylim_val;
551 }
552 var zRange = plot _cont .fullLayout .scene .zaxis .range;
553 if (plot .hasOwnProperty( 'zlim_val' )) {
554   zRange = plot .zlim_val;
555   layout3[ 'scene .zaxis .range' ] = plot .zlim_val;
556 }
557
558 var zoom = 1;
559 if (typeof plot .zoom_val != 'undefined' ) {
560   zoom = plot .zoom_val;
561 }
562
563 if (plot .axis_style_val == 'equal' ) {
564   var xRangeSize = Math .abs(xRange[1] - xRange[0]);
565   var yRangeSize = Math .abs(yRange[1] - yRange[0]);
566   var zRangeSize = Math .abs(zRange[1] - zRange[0]);
567   var maxRange = Math .max(xRangeSize , yRangeSize , zRangeSize );
568
569   layout3[ "scene .aspectratio" ] = {
570     x: xRangeSize / maxRange * zoom ,
571     y: yRangeSize / maxRange * zoom ,
572     z: zRangeSize / maxRange * zoom
573   };
574 } else {
575   layout3[ "scene .aspectratio" ] = {
576     x: zoom ,
577     y: zoom ,
578     z: zoom
579   };
580 }
581
582 var radius = maxRange/2;
583 var azimuthRad = plot .view_val[0] * Math .PI / 180;
584 var elevationRad = plot .view_val[1] * Math .PI / 180;
585
586 var eye = {
587   x: radius * Math .cos(elevationRad) * Math .cos(azimuthRad) ,
588   y: radius * Math .cos(elevationRad) * Math .sin(azimuthRad) ,
589   z: radius * Math .sin(elevationRad)
590 };
591
592 var up = { x: 0 , y: 0 , z: 1 };
593
594 layout3[ 'scene .camera .eye' ] = eye;
595 layout3[ 'scene .camera .up' ] = up;
596
597 context_plot .relayout(layout3);
598 context_plot .resize();
599 }
600
601 /**
602 */

```

```

603  * Updates plot data for a specified figure.
604  * @param {string} fid - The identifier of the figure to update.
605  * @param {Object} traces - The trace data to be updated in the plot.
606  * @param {number} N - The new data length or index for updating the plot.
607  */
608 updateData(fid, traces, N) {
609   if(!this.jsl.figures.open_figures.hasOwnProperty(fid)) {
610     this.jsl.env.error('@ploter/updateData: '+language.string(172));
611     return;
612   }
613   var figure = this.jsl.figures.open_figures[fid];
614   figure.context.plot.updateData(traces, N);
615 }

616 /**
617  * Updates plot data for a specified figure.
618  * @param {string} fid - Identifier of the figure to update.
619  * @param {Object|Object[]} data - Single trace or array of traces, each
620  *       with an 'id' and properties to restyle.
621  */
622 updateDataById(fid, data) {
623   if(!this.jsl.figures.open_figures.hasOwnProperty(fid)) {
624     this.jsl.env.error('@ploter/updateDataById: '+language.string(172));
625     return;
626   }
627   var figure = this.jsl.figures.open_figures[fid];
628   figure.context.plot.updateDataById(data);
629 }

630 /**
631  * Handles plot resizing for a specified figure identifier.
632  * @param {number} fid - The identifier for the figure to resize.
633  */
634 onResize(fid) {
635   if(!this.jsl.figures.open_figures.hasOwnProperty(fid)) {
636     this.jsl.env.error('@ploter/onResize: '+language.string(172));
637     return;
638   }
639   var figure = this.jsl.figures.open_figures[fid];
640   figure.context.plot.resize();
641 }

642 /**
643  * Removes the plot container for a specified figure identifier.
644  * @param {number} fid - The identifier for the figure from which to remove
645  *       the plot container.
646  */
647 remove(fid) {
648   if(!this.jsl.figures.open_figures.hasOwnProperty(fid)) {
649     this.jsl.env.error('@ploter/remove: '+language.string(172));
650     return;
651   }
652   var figure = this.jsl.figures.open_figures[fid];
653   var plot_cont = figure.context.plot.plot_cont;
654   if(plot_cont) {
655

```

```

656         plot._cont.remove();
657     }
658 }
659 /**
660 * Sets the plot data from JSON.
661 * @param {number} fid - The identifier for the figure to update.
662 * @param {Array} data Data for the plot.
663 */
664
665 async fromJSON(fid, data) {
666   var figure = this.jsl.figures.open_figures[fid];
667   var plot = figure.plot;
668   var context_plot = figure.context.plot;
669
670   var plot_cont = figure.dom.querySelector('#figure-content .plot-cont');
671   if(!plot_cont) {
672     context_plot.setCont();
673   }
674
675   plot.traces = data.data;
676   this._JsonToOptions(figure, data);
677   await context_plot.fromJSON(data);
678 }
679
680 /**
681 * Sets the plot data from JSON.
682 * @param {Array} fig Data for the plot.
683 * @returns {Boolean} - Returns true if the figure is 3D.
684 */
685 _is3DFigure(fig) {
686   var types_3d = new Set([
687     'scatter3d', 'surface', 'mesh3d',
688     'cone', 'streamtube', 'volume', 'isosurface'
689   ]);
690   var trace_is_3d = fig.data?.some(t => types_3d.has(t.type));
691   var layout_has_scene = Object.keys(fig.layout || {}).some(k => k.startsWith('scene'));
692
693   return trace_is_3d || layout_has_scene;
694 }
695
696 /**
697 * Re-creates plot.options and the extra synthetic fields
698 * (title_val, xlim_val, ...) from a saved Plotly figure JSON.
699 *
700 * @param {Object} figure runtime figure wrapper (has .plot, .dom, ...)
701 * @param {Object} data JSON produced by Plotly.toJSON / graphJson
702 */
703 _JsonToOptions(figure, data) {
704   var plot = figure.plot;
705   var layout = data.layout ?? {};
706
707   figure.layout_3d = this._is3DFigure(data);
708
709   figure.dom.getElementById('figure-menu').className =
710     figure.layout_3d ? 'figure-3d' : 'figure-2d';

```

```

711
712   if (! plot.axis_style_val) {
713     plot.axis_style_val = figure.layout_3d ? 'equal' : 'normal';
714   }
715
716   var options = {};
717
718   if (typeof layout.showlegend === 'boolean') {
719     options.showLegend = layout.showlegend;
720     plot.legend_state_val = layout.showlegend ? 'on' : 'off';
721   }
722
723   if (layout.legend && Object.keys(layout.legend).length) {
724     var lg = JSON.parse(JSON.stringify(layout.legend));
725     options.legend = lg;
726
727     if (lg.orientation)
728       options.legendOrientation =
729         lg.orientation === 'h' ? 'horizontal' : 'vertical';
730
731     {
732       var { x=0, y=0 } = lg;
733       var outsideX = x < 0 || x > 1;
734       var outsideY = y < 0 || y > 1;
735
736       var dirNS = y <= (outsideY ? -0.02 : 0.02) ? 'south'
737                     : y >= (outsideY ? 1.02 : 0.98) ? 'north' : '';
738       var dirEW = x <= (outsideX ? -0.02 : 0.02) ? 'west'
739                     : x >= (outsideX ? 1.02 : 0.98) ? 'east' : '';
740
741       let loc = (dirNS + dirEW) || 'north';
742       if (outsideX || outsideY) loc += 'outside';
743       options.legendLocation = loc.toLowerCase();
744     }
745   }
746
747   if (figure.layout_3d) {
748     var sc = layout.scene ?? {};
749     if (sc.xaxis?.range) options.xlim = plot.xlim_val = [...sc.xaxis.range];
750     if (sc.yaxis?.range) options.ylim = plot.ylim_val = [...sc.yaxis.range];
751     if (sc.zaxis?.range) options.zlim = plot.zlim_val = [...sc.zaxis.range];
752   } else {
753     if (layout.xaxis?.range) options.xlim = plot.xlim_val = [...layout.xaxis.range];
754     if (layout.yaxis?.range) options.ylim = plot.ylim_val = [...layout.yaxis.range];
755   }
756
757   if (layout.font) options.font = JSON.parse(JSON.stringify(layout.font))
758   ;
759   if (layout.margin) {
760     var {t, b, l, r} = layout.margin;
761     options.margin = {
762       ... (t != null && { top: t }) ,
763       ... (b != null && { bottom: b }) ,

```

```

763     ... ( l != null && { left: l } ) ,
764     ... ( r != null && { right: r } )
765   };
766 }
767
768 if(layout.title?.text) plot.title_val = layout.title.text;
769
770 if (figure.layout_3d) {
771   var sc = layout.scene ?? {};
772   if (sc.xaxis?.title?.text) plot.xlabel_val = sc.xaxis.title.text;
773   if (sc.yaxis?.title?.text) plot.ylabel_val = sc.yaxis.title.text;
774   if (sc.zaxis?.title?.text) plot.zlabel_val = sc.zaxis.title.text;
775 } else {
776   if (layout.xaxis?.title?.text) plot.xlabel_val = layout.xaxis.title.text;
777   if (layout.yaxis?.title?.text) plot.ylabel_val = layout.yaxis.title.text;
778 }
779
780 if (figure.layout_3d) {
781   var sc = layout.scene ?? {};
782
783   if (sc.aspectratio) {
784     var {x = 1, y = 1, z = 1} = sc.aspectratio;
785     var minAR = Math.min(x,y,z), maxAR = Math.max(x,y,z);
786     var approxEqual = (maxAR - minAR) < 1e-6;
787     plot.axis_style_val = approxEqual ? 'equal' : 'normal';
788   }
789
790   if (sc.camera?.eye) {
791     var {x: ex = 1, y: ey = 1, z: ez = 1} = sc.camera.eye;
792     var radius = Math.hypot(ex, ey, ez);
793     var azimuth = (Math.atan2(ey, ex) * 180 / Math.PI + 360) % 360;
794     var elevation = Math.asin(ez / radius) * 180 / Math.PI;
795     plot.view_val = [azimuth, elevation];
796
797     if (sc.aspectratio && plot.xlim_val && plot.ylim_val && plot.zlim_val)
798     {
799       var dx = Math.abs(plot.xlim_val[1] - plot.xlim_val[0]);
800       var dy = Math.abs(plot.ylim_val[1] - plot.ylim_val[0]);
801       var dz = Math.abs(plot.zlim_val[1] - plot.zlim_val[0]);
802       var maxRange = Math.max(dx, dy, dz);
803       var intrinsicAR = { x: dx/maxRange, y: dy/maxRange, z: dz/maxRange };
804
805       var {x: ax = 1, y: ay = 1, z: az = 1} = sc.aspectratio;
806       var zoom = (ax/(intrinsicAR.x || 1) +
807                  ay/(intrinsicAR.y || 1) +
808                  az/(intrinsicAR.z || 1)) / 3;
809       plot.zoom_val = zoom;
810     }
811   } else {
812     plot.axis_style_val =
813       layout.yaxis?.scaleanchor === 'x' ? 'equal' : 'normal';
814   }
815   figure.plot.options = options;
}

```



```
816 }
817
818 exports.PRDC_JSLAB_PLOTER = PRDC_JSLAB_PLOTER;
```

Listing 107 - jslab-ploter-plotly.js

```
1 /**
2  * @file JSLAB library
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 const { PRDC_JSLAB_EVAL } = require('./jslab-eval');
9 const { PRDC_JSLAB_OVERRIDE } = require('./jslab-override');
10 const { PRDC_JSLAB_ENV } = require('./jslab-env-electron');
11
12 /**
13  * Class for JSLAB library .
14 */
15 class PRDC_JSLAB_LIB {
16
17 /**
18  * Constructs the JSLAB library environment , initializing submodules and
19  * setting up the execution context .
20 */
21 constructor() {
22     var obj = this ;
23     this.ready = false ;
24
25     // Library built in properties
26     this.previous_workspace = {};
27     this.previous_properties = [] ;
28
29     this.stop_loop = false ;
30     this.last_script_path ;
31     this.last_script_lines ;
32     this.last_script_silent ;
33
34     this.current_script ;
35
36     this.required_modules = [];
37     this.started_timeouts = [];
38     this.started_intervals = [];
39     this.started_animation_frames = [];
40     this.started_idle_callbacks = [];
41     this.started_immediates = [];
42     this.cleanup_registry = [];
43     this.started_promises = {};
44     this.promises_number = 0;
45     this.last.promise_id = 0;
46
47     this.env = new PRDC_JSLAB_ENV(this) ;
48     this.context = this.env.context ;
49     this.setDepthSafeStringify(this.context , 1) ;
50     this.context._ = Symbol('_') ;
```

```
50     this.eval = new PRDC_JSLAB_EVAL(this);
51
52     this.debug = this.env.debug;
53     this.includes_path = this.env.getDefaultPath('includes');
54
55     if(config.PLOTER == 'plotly') {
56         var { PRDC_JSLAB_PLOTER } = require('./jslab-ploter-plotly');
57     } else if(config.PLOTER == 'echarts') {
58         var { PRDC_JSLAB_PLOTER } = require('./jslab-ploter-echarts');
59     }
60     this.ploter = new PRDC_JSLAB_PLOTER(this);
61
62     this.override = new PRDC_JSLAB_OVERRIDE(this);
63
64     this.ready = true;
65 }
66
67 /**
68 * Invoked at the beginning of a code evaluation to set up necessary flags
69 * and state.
70 */
71 onEvaluating() {
72     this.env.codeEvaluating();
73     this.ignore_output = false;
74     this.no_ans = false;
75     this.stop_loop = false;
76     this.env.resetStopLoop();
77     this.env.setStatus('busy', language.string(88));
78 }
79
80 /**
81 * Invoked after code evaluation to finalize the state and update the
82 * environment accordingly.
83 */
84 onEvaluated() {
85     this.env.setWorkspace();
86     this.env.codeEvaluated();
87     this.env.setStatus('ready', language.string(87));
88 }
89
90 /**
91 * Triggers when there are changes in the stats, such as the number of
92 * promises or timeouts, and updates the environment state.
93 */
94 onStatsChange() {
95     this.env.setWorkspace();
96     this.env.setStats({
97         'required_modules': this.required_modules.length,
98         'promises': this.promises_number,
99         'timeouts': this.started_timeouts.length,
100        'immediates': this.started_immediates.length,
101       'intervals': this.started_intervals.length,
102      'animation_frames': this.started_animation_frames.length,
103      'idle_callbacks': this.started_idle_callbacks.length
104    });
105 }
```

```

102 }
103
104 /**
105 * Clears the current workspace, stopping any ongoing operations and
106 * removing any dynamic properties.
107 */
108 clear() {
109   this.doCleanup();
110   this.context.sym.clear();
111   this.context.parallel.terminate();
112
113   this.clearEventListeners();
114   this.clearImmediates();
115   this.clearAnimationFrames();
116   this.clearIntervals();
117   this.clearTimeouts();
118   this.clearIdleCallbacks();
119   this.stopPromises();
120   this.stopSubprocesses();
121   this.unrequireAll();
122   this.onStatsChange();
123
124   var obj = this;
125   this.previous_properties.forEach(function(property) {
126     if(typeof obj.context[property] != 'function' || (typeof obj.context[
127       property] == 'function' && obj.context[property]._jsl_saved)) {
128       delete obj.context[property];
129     }
130   });
131   this.previous_properties = [];
132   this.previous_workspace = {};
133   this.no_ans = true;
134   this.ignore_output = true;
135 }
136
137 /**
138 * Sets the current path in the environment, adjusting how file paths are
139 * resolved.
140 * @param {String} new_path The new path to set as the current working
141 * directory.
142 */
143 setPath(new_path) {
144   this.current_path = this.env.addPathSep(new_path);
145 }
146
147 /**
148 * Saves the properties of the current workspace to restore them later if
149 * needed.
150 */
151 savePreviousWorkspace() {
152   var obj = this;
153   this.previous_properties = this.getWorkspaceProperties();
154   this.previous_properties.forEach(function(property) {
155     if(typeof obj.context[property] == 'function') {
156       obj.context[property]._jsl_saved = true;
157     }
158   });
159 }

```

```

152      }
153      obj.previous_workspace[property] = obj.context[property];
154      delete obj.context[property];
155  });
156}
157
158 /**
159 * Restores the properties of the workspace that were saved by ‘
160 * savePreviousWorkspace’.
161 */
162loadPreviousWorkspace() {
163  var obj = this;
164  var workspace = this.getWorkspaceProperties();
165  workspace.forEach(function(property) {
166    if(obj.previous_properties.includes(property)) {
167      // This property is defined again in new code
168      obj.previous_properties.splice(
169        obj.previous_properties.indexOf(property), 1);
170  });
171
172  // Add saved properties back to global scope
173  this.previous_properties.forEach(function(property) {
174    obj.context[property] = obj.previous_workspace[property];
175  });
176}
177
178 /**
179 * Retrieves a list of custom properties added to the global context since
180 * the initial load of the library.
181 * @returns {Array} A list of property names that have been added to the
182 * global context.
183 */
184getWorkspaceProperties() {
185  var initial_props = this.initial_workspace;
186  var current_prop_list = Object.getOwnPropertyNames(this.context);
187  var workspace = [];
188
189  for(var i = 0, l = current_prop_list.length; i < l; ++i) {
190    var prop_name = current_prop_list[i];
191    if(initial_props.indexOf(prop_name) === -1) {
192      workspace.push(prop_name);
193    }
194  }
195
196 /**
197 * Constructs a detailed view of the current workspace, including the types
198 * and names of properties.
199 * @returns {Array} An array of arrays, each containing the name, type, and
200 * constructor name (if applicable) of each property.
201 */
202getWorkspace() {
203  var initial_props = this.initial_workspace;

```

```

202   var current_prop_list = Object.getOwnPropertyNames(this.context);
203   var workspace = [];
204
205   for(var i = 0, l = current_prop_list.length; i < l; ++i) {
206     var prop_name = current_prop_list[i];
207     if(initial_props && initial_props.indexOf(prop_name) === -1) {
208       var prop = this.context[prop_name];
209       var type = typeof prop;
210       var name = 'none';
211
212       if(type !== 'undefined') {
213         if(prop && prop.constructor) {
214           name = prop.constructor.name;
215         } else {
216           name = '/';
217         }
218       }
219       workspace.push([prop_name, type, name]);
220     }
221   }
222
223   return workspace;
224 }
225
226 /**
227 * Generates the initialization script for the worker environment.
228 * @returns {string} – The worker initialization script as a string.
229 */
230 getWorkerInit() {
231   return `
232     global.app_path = ${JSON.stringify(app_path)};
233     global.is_worker = true;
234     global.debug = ${this.env.debug};
235     global.version = '${this.env.version}';
236     global.platform = '${this.env.platform}';
237
238
239     const helper = require(app_path + '/js/helper.js');
240     require(app_path + '/js/init-config.js');
241
242     importScripts(app_path + '/lib/luxon-3.4.4/luxon-3.4.4.min.js');
243     importScripts(app_path + '/lib/math-11.8.2/math-11.8.2.min.js');
244     importScripts(app_path + '/lib/sprintf-1.1.3/sprintf-1.1.3.min.js');
245     importScripts(app_path + '/lib/sympy-0.26.2/pyodide.js');
246
247     const { PRDC_JSLAB_LIB } = require(app_path + '/js/sandbox/jslab');
248
249     // Global variables
250     var jsl = new PRDC_JSLAB_LIB();
251     jsl.current_path = ${JSON.stringify(this.current_path)};
252     jsl.includes_path = ${JSON.stringify(this.includes_path)};
253     jsl.saved_paths = JSON.parse('${JSON.stringify(this.saved_paths)})');
254   `;
255 }
256

```

```

257  /**
258   * Resolves the absolute path for a file , searching through predefined
259   * directories and optionally using module context .
260   * @param {string} file_path - The file path to resolve .
261   * @param {object} [this_module] - Optional module context for resolution .
262   * @returns {string|boolean} - The resolved path , or 'false' if not found .
263   */
264   pathResolve(file_path , this_module) {
265     var obj = this ;
266     if(typeof file_path == 'string') {
267       if(!this.env.pathIsAbsolute(file_path)) {
268         if(this_module) {
269           var file_path_temp = this.env.pathJoin(this_module.path , file_path);
270           if(this.env.checkFile(file_path_temp)) {
271             return file_path_temp;
272           }
273         }
274         var file_paths = [];
275         var file_path_temp = this.env.pathJoin(this.current_path , file_path);
276         if(this.env.checkFile(file_path_temp)) {
277           file_paths.push(file_path_temp);
278         }
279         file_path_temp = this.env.pathJoin(this.includes_path , file_path);
280         if(this.env.checkFile(file_path_temp)) {
281           file_paths.push(file_path_temp);
282         }
283         this.saved_paths.forEach(function(saved_path) {
284           file_path_temp = obj.env.pathJoin(saved_path , file_path);
285           if(obj.env.checkFile(file_path_temp)) {
286             file_paths.push(file_path_temp);
287           }
288         });
289         if(file_paths.length > 2) {
290           var N = file_paths.length;
291           var str = '@pathResolve: '+language.string(106)+'' + file_paths[0]
292             + ''+language.string(107)+': [\n';
293           for(var i = 1; i < N-1; i++) {
294             str += ''+file_paths[i]+',\n';
295           }
296           str += ''+file_paths[N-1]+'\n';
297           this.env.disp(str);
298         } else if(file_paths.length > 1) {
299           this.env.disp('@pathResolve: '+language.string(106)+'' + file_paths
300             [0]+ ''+language.string(108)+': [\n' + file_paths[1]+ '\n]');
301         } else if(file_paths.length == 0) {
302           try {
303             if(this_module) {
304               return this.override._Module._resolveFilename(file_path ,
305                 this_module , false);
306             } else {
307               return require.resolve(file_path);
308             }
309           } catch(err) {

```

```
307     this._console.log(err);
308     this.env.error('@pathResolve: '+language.string(109)+'' +
309                   file_path + ''+language.string(110)+'.');
310     return false;
311   }
312   file_path = file_paths[0];
313 }
314 else {
315   this.env.error('@pathResolve: '+language.string(111)+'.');
316   return false;
317 }
318 return file_path;
319 }

320 /**
321 * Sets the flag to stop loop execution within the JSLAB environment.
322 * @param {Boolean} data The flag indicating whether to stop loop execution.
323 */
324 setStopLoop(data) {
325   this.stop_loop = data;
326   this.onStopLoop(false);
327   this.onEvaluated();
328   this.env.disp(language.string(90));
329 }

330 /**
331 * Handles the process of stopping loop execution and cleaning up
332 * asynchronous operations.
333 * @param {Boolean} [throw_error=true] Indicates whether to throw an error
334 * when stopping the loop.
335 * @throws {Object} An error object with a custom message if 'throw_error'
336 * is true and the loop needs to be stopped.
337 */
338 onStopLoop(throw_error = true) {
339   if(this.stop_loop) {
340     this.clearEventListeners();
341     this.clearImmediates();
342     this.clearAnimationFrames();
343     this.clearIntervals();
344     this.clearTimeouts();
345     this.clearIdleCallbacks();
346     this.stopPromises();
347     this.stopSubprocesses();
348     this.doCleanup();
349     this.onStatsChange();
350     if(throw_error) {
351       throw {name: 'JslabError', message: language.string(90)};
352     }
353   }
354 }

355 /**
356 * Sets the paths saved in the environment for easier access to files and
357 * modules.
```

```
357     * @param {Array} data An array of paths to be saved for future use.  
358     */  
359     setSavedPaths(data) {  
360         this.saved_paths = data;  
361     }  
362  
363     /**  
364      * Placeholder function for features that have not been implemented.  
365      */  
366     notImplemented() {  
367         obj.env.error(language.string(115));  
368     }  
369  
370     /**  
371      * Clears all modules that have been required during the session.  
372      */  
373     unrequireAll() {  
374         var obj = this;  
375         this.required_modules.forEach(function(module) {  
376             try {  
377                 var name = require.resolve(module);  
378                 if(name) {  
379                     delete require.cache[name];  
380                 }  
381             } catch(err) {  
382                 obj._console.log(err);  
383             }  
384         });  
385         this.required_modules = [];  
386         Object.keys(require.cache).forEach(function(key) { delete require.cache[key]; });  
387     }  
388  
389     /**  
390      * Stops all promises that have been started within the environment.  
391      */  
392     stopPromises() {  
393         var obj = this;  
394         Object.keys(this.started_promises).forEach(function(key) {  
395             obj.started_promises[key].loop_stoped = true;  
396             delete obj.started_promises[key];  
397         });  
398         this.promises_number = 0;  
399     }  
400  
401     /**  
402      * Registers an object for cleanup with a specified cleanup function.  
403      * @param {Object} obj - The object to be registered for cleanup.  
404      * @param {Function} fun - The function to execute during cleanup.  
405      */  
406     addForCleanup(obj, fun) {  
407         this.cleanup_registry.push({obj: obj, fun: fun});  
408     }  
409  
410     /**
```

```
411 * Do cleanup on all registered objects;
412 */
413 doCleanup() {
414     for(var entry of this.cleanup_registry) {
415         if(isFunction(entry.fun)) {
416             try {
417                 entry.fun();
418             } catch {};
419         } else if(isFunction(entry.obj._jslabCleanup)) {
420             try {
421                 entry.obj._jslabCleanup();
422             } catch {};
423         }
424     }
425     this.cleanup_registry = [];
426 }
427
428 /**
429 * Removes a specific promise from the tracking list based on its ID.
430 * @param {Number} id The ID of the promise to remove.
431 */
432 clearPromise(id) {
433     this.promises_number -= 1;
434     if(this.promises_number < 0) {
435         this.promises_number = 0;
436     }
437     this.onStatsChange();
438     delete this.started_promises[id];
439 }
440
441 /**
442 * Clears all timeouts that have been set within the environment.
443 */
444 clearTimeouts() {
445     var obj = this;
446     this.started_timeouts.forEach(function(timeout) {
447         obj._clearTimeout(timeout);
448     });
449     this.started_timeouts = [];
450 }
451
452 /**
453 * Clears all intervals that have been set within the environment.
454 */
455 clearIntervals() {
456     var obj = this;
457     this.started_intervals.forEach(function(interval) {
458         obj._clearInterval(interval);
459     });
460     this.started_intervals = [];
461 }
462
463 /**
464 * Cancels all animation frames that have been requested within the
465 * environment.
```

```
465  */
466 clearAnimationFrames() {
467     var obj = this;
468     this.started_animation_frames.forEach(function(animation_frames) {
469         obj._cancelAnimationFrame(animation_frames);
470     });
471     this.started_animation_frames = [];
472 }
473
474 /**
475 * Cancels all idle callbacks that have been requested within the
476 environment.
477 */
478 clearIdleCallbacks() {
479     var obj = this;
480     this.started_idle_callbacks.forEach(function(idle_callback) {
481         obj._cancelIdleCallback(idle_callback);
482     });
483     this.started_idle_callbacks = [];
484 }
485 /**
486 * Clears all immediate executions that have been set within the environment
487 .
488 */
489 clearImmediates() {
490     var obj = this;
491     this.started_immediates.forEach(function(immediate) {
492         obj._clearImmediate(immediate);
493     });
494     this.started_immediates = [];
495 }
496 /**
497 * Removes all event listeners that have been added to the document body,
498 * effectively clearing any remaining event bindings.
499 */
500 clearEventListeners() {
501     document.body.parentNode.replaceChild(document.body.cloneNode(true),
502                                         document.body);
503 }
504 /**
505 * Lists all subprocesses.
506 */
507 stopSubprocesses() {
508     var pids = this.listSubprocesses();
509     pids.forEach(function(pid) {
510         killProcess(pid);
511     });
512 }
513 /**
514 * Sets the safe stringify depth for the given data.
515 * @param {Object} data - The data object to modify.
```

```

516   * @param {number} depth - The depth level for safe stringification.
517   * @returns {Object} The modified data object with the set depth.
518   */
519   setDepthSafeStringify(data, depth) {
520     data._safeStringifyDepth = depth;
521     return data;
522   }
523
524 /**
525  * Lists all subprocesses.
526  */
527 listSubprocesses() {
528   return this.env.native_module.listSubprocesses(this.env.process_pid);
529 }
530
531 /**
532  * Gets properties missing documentation.
533  * @param {Array|string} [workspace=this.initial_workspace] - Workspace to
534  * check.
535  * @param {boolean} [without_builtin=true] - Exclude built-in properties.
536  * @returns {Array<string>} Missing property names.
537  */
538 _getMissingDocs(workspace = this.initial_workspace, without_builtin = true)
539 {
540   var missing = [];
541   for(var prop of workspace) {
542     if(!prop.startsWith('_') &&
543         (!without_builtin ||
544          !this.builtin_workspace.includes(prop))) {
545       try {
546         help(prop);
547       } catch {
548         missing.push(prop);
549       }
550     }
551   }
552   return missing;
553 }
554 /**
555  * Writes templates for missing docs to a file.
556  * @param {string} path - File path for missing docs.
557  * @param {Array|string} [workspace=this.initial_workspace] - Workspace to
558  * check.
559  */
560 _writeMissingDocsToFile(path, workspace = this.initial_workspace) {
561   var workspace_array = workspace;
562   if(!Array.isArray(workspace)) {
563     workspace_array = Object.keys(workspace);
564   }
565   var missing = this._getMissingDocs(workspace_array);
566   var str = '';
567   for(var prop of missing) {
      var kind = 'function member';

```

```

568     if (!Array.isArray(workspace)) {
569         kind = typeof workspace[prop] === 'function' ? 'function' : 'member';
570     }
571     str += '
572 /**
573 * Description
574 * @name ${prop}
575 * @kind ${kind}
576 * @param {}
577 * @returns {}
578 * @memberof PRDC_JSLAB_LIB_
579 */
580 ';
581     }
582     this.env.writeFileSync(path, str);
583 }
584 /**
585 * Awaits the provided promise if the loop has not been stopped.
586 * @param {Promise} p - The promise to await.
587 * @returns {Promise|undefined} - Resolves if 'p' is awaited; does nothing
588 * if the loop is stopped.
589 */
590 async promiseOrStoped(p) {
591     if (!p.loop_stoped) {
592         await p;
593     }
594 }
595 }
596 }
597
598 exports.PRDC_JSLAB_LIB = PRDC_JSLAB_LIB;

```

Listing 108 - jslab.js

```

1 /**
2 * @file JSLAB test
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 const { PRDC_JSLAB_TESTS } = require('../tester');
9 var tests = new PRDC_JSLAB_TESTS();
10
11 tests.add('Expect tic to be equal to last tic', function() {
12     return tic === jsl.context.last_tic;
13 }
14 );
15
16 tests.add('Joining two paths with path separator', function() {
17     return 'a'+pathSep()+''b' === jsl.env.pathJoin('a', 'b');
18 }
19 );
20

```

```
21 exports.MODULE_TESTS = tests;
```

Listing 109 - jslab.test.js

```

1  /**
2   * @file JSLAB library math submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8
9 /**
10  * Class for JSLAB math submodule.
11 */
12 class PRDC_JSLAB_LIB_MATH {
13
14 /**
15  * Constructs the math submodule, initializing mathematical constants and
16  * functions.
17 */
18 constructor(jsl) {
19   var obj = this;
20   this.jsl = jsl;
21
22  // Additional math constants
23  if(this.jsl.env.math) {
24    /**
25     * Pi number.
26     * @type {number}
27     */
28  this.Pi = this.jsl.env.math.pi;
29
30  /**
31   * Coefficient for converting degrees to radians.
32   * @type {number}
33   */
34  this.d2r = this.jsl.env.math.pi/180;
35
36  /**
37   * Coefficient for converting radians to degrees.
38   * @type {number}
39   */
40  this.r2d = 1/this.d2r;
41 }
42
43 /**
44  * Floating-point relative accuracy
45  * @type {number}
46  */
47 this.eps = 1E-7;
48
49 /**
50  * Floating-point relative accuracy
51  * @type {number}

```

```

52      */
53     this.EPS = this.eps;
54   }
55
56 /**
57 * Seeds the random number generator with the provided arguments.
58 * @param {...any} args - Arguments used to seed the random generator.
59 * @returns {any} The result from the seeded random generator.
60 */
61 seedRandom(...args) {
62   return this.jsl.env.seedRandom(...args);
63 }
64
65 /**
66 * Performs linear interpolation on a set of data points.
67 * @param {Array} x The x-values of the data points.
68 * @param {Array} y The y-values of the data points, corresponding to each x
69 * -value.
70 * @param {Number|Array} xq The x-value(s) for which to interpolate a y-
71 * value.
72 * @param {String} mode The mode of interpolation. Use 'extrap' for
73 * extrapolation.
74 * @returns {Number|Array} The interpolated y-value(s) at xq.
75 */
76 interp(x, y, xq, mode = 'none') {
77   // Helper function for scalar interpolation
78   const interpolate = (xqi) => {
79     // Use lastIndexOf to get the last occurrence of an exact match.
80     const i = x.lastIndexOf(xqi);
81     if(i >= 0) {
82       return y[i];
83     }
84     // Find the closest points for interpolation
85     let x1 = null, x2 = null;
86     for(let v of x) {
87       if(v < xqi && (x1 === null || v > x1)) {
88         x1 = v;
89       }
90       if(v > xqi && (x2 === null || v < x2)) {
91         x2 = v;
92       }
93     }
94     // Handle extrapolation if mode is 'extrap'
95     if(mode === 'extrap') {
96       if(xqi < x[0]) {
97         const gradient = (y[1] - y[0]) / (x[1] - x[0]);
98         return y[0] + (xqi - x[0]) * gradient;
99       } else if(xqi > x[x.length - 1]) {
100         const lastIdx = x.length - 1;
101         const gradient = (y[lastIdx] - y[lastIdx - 1]) / (x[lastIdx] - x[
102           lastIdx - 1]);
103         return y[lastIdx] + (xqi - x[lastIdx]) * gradient;
104       }
105     }
106   }
107 }
```

```

103 }
104
105 // Interpolate between x1 and x2
106 if(x1 !== null && x2 !== null && x1 !== x2) {
107   // Use the last occurrence for x1 and the first occurrence for x2
108   const idx1 = x.lastIndexOf(x1);
109   const idx2 = x.indexOf(x2);
110   const y1 = y[idx1];
111   const y2 = y[idx2];
112   return y1 + (xqi - x1) * (y2 - y1) / (x2 - x1);
113 }
114
115 // Return NaN if no valid interpolation point is found
116 return NaN;
117 };
118
119 // If xq is an array, map over each element
120 return Array.isArray(xq) ? xq.map(interpolate) : interpolate(xq);
121 }
122
123 /**
124 * Computes the gradient of a 2D grid.
125 * @param {Array} x - X coordinates.
126 * @param {Array} y - Y coordinates.
127 * @param {Array[]} z - 2D data array.
128 * @param {number} [N_a=11] - Neighborhood size.
129 * @returns {Array[]} Gradient components [dz_x, dz_y].
130 */
131 gridGradient(x, y, z, N_a = 11) {
132   var obj = this;
133   var hx = x[1] - x[0];
134   var hy = y[1] - y[0];
135   var N_a_c = Math.floor(N_a / 2);
136
137   var n = z.length;
138   var m = z[0].length;
139
140   var dz_x = zeros(n, m);
141   var dz_y = zeros(n, m);
142   var dz_x_m = zeros(n, m);
143   var dz_y_m = zeros(n, m);
144
145   // Compute the gradient at a single point (x,y)
146   // In our function, x is the horizontal index and y is the vertical index.
147   function pointGradient(x, y) {
148     if(x < 0 || x >= m || y < 0 || y >= n) return;
149     // Check if gradients are already calculated
150     if(dz_x[y][x] !== 0 || dz_y[y][x] !== 0) return;
151
152     if(obj.jsl._isNaN(z[y][x])) {
153       // If the current value is NaN or Inf, try using central differences
154       // if available
155       // x component:
156       if(x === 0 || x === m - 1) {
157         dz_x[y][x] = 0;
158       } else {
159         dz_x[y][x] = (z[y][x + 1] - z[y][x - 1]) / (2 * hx);
160       }
161     } else {
162       dz_x[y][x] = (z[y][x + 1] - z[y][x - 1]) / (2 * hx);
163     }
164
165     if(obj.jsl._isNaN(z[y][x])) {
166       // If the current value is NaN or Inf, try using central differences
167       // if available
168       // y component:
169       if(y === 0 || y === n - 1) {
170         dz_y[y][x] = 0;
171       } else {
172         dz_y[y][x] = (z[y + 1][x] - z[y - 1][x]) / (2 * hy);
173       }
174     } else {
175       dz_y[y][x] = (z[y + 1][x] - z[y - 1][x]) / (2 * hy);
176     }
177   }
178
179   // Compute the gradient for the entire grid
180   for(let y = 0; y < n; y++) {
181     for(let x = 0; x < m; x++) {
182       pointGradient(x, y);
183     }
184   }
185
186   return [dz_x, dz_y];
187 }

```

```

157     } else if (!obj.jsl._isNaN(z[y][x - 1]) && !obj.jsl._isNaN(z[y][x + 1]))
158         ) {
159     dz_x[y][x] = (z[y][x + 1] - z[y][x - 1]) / 2;
160 } else {
161     dz_x[y][x] = 0;
162 }
163 // y component:
164 if(y == 0 || y == n - 1) {
165     dz_y[y][x] = 0;
166 } else if(!obj.jsl._isNaN(z[y - 1][x]) && !obj.jsl._isNaN(z[y + 1][x]))
167     ) {
168     dz_y[y][x] = (z[y + 1][x] - z[y - 1][x]) / 2;
169 } else {
170     dz_y[y][x] = 0;
171 }
172 // If the value is valid, use one-sided differences on the boundaries
173 // x component:
174 if(x == 0 || x == m - 1) {
175     if(x == 0 && !obj.jsl._isNaN(z[y][x + 1])) {
176         dz_x[y][x] = z[y][x + 1] - z[y][x];
177     } else if(x == m - 1 && !obj.jsl._isNaN(z[y][x - 1])) {
178         dz_x[y][x] = z[y][x] - z[y][x - 1];
179     } else {
180         dz_x[y][x] = 0;
181     }
182 } else {
183     if(!obj.jsl._isNaN(z[y][x - 1]) && !obj.jsl._isNaN(z[y][x + 1])) {
184         dz_x[y][x] = (z[y][x + 1] - z[y][x - 1]) / 2;
185     } else if(!obj.jsl._isNaN(z[y][x + 1])) {
186         dz_x[y][x] = z[y][x + 1] - z[y][x];
187     } else if(!obj.jsl._isNaN(z[y][x - 1])) {
188         dz_x[y][x] = z[y][x] - z[y][x - 1];
189     } else {
190         dz_x[y][x] = 0;
191     }
192 }
193 // y component:
194 if(y == 0 || y == n - 1) {
195     if(y == 0 && !obj.jsl._isNaN(z[y + 1][x])) {
196         dz_y[y][x] = z[y + 1][x] - z[y][x];
197     } else if(y == n - 1 && !obj.jsl._isNaN(z[y - 1][x])) {
198         dz_y[y][x] = z[y][x] - z[y - 1][x];
199     } else {
200         dz_y[y][x] = 0;
201     }
202 } else {
203     if(!obj.jsl._isNaN(z[y - 1][x]) && !obj.jsl._isNaN(z[y + 1][x])) {
204         dz_y[y][x] = (z[y + 1][x] - z[y - 1][x]) / 2;
205     } else if(!obj.jsl._isNaN(z[y + 1][x])) {
206         dz_y[y][x] = z[y + 1][x] - z[y][x];
207     } else if(!obj.jsl._isNaN(z[y - 1][x])) {
208         dz_y[y][x] = z[y][x] - z[y - 1][x];
209     } else {
210         dz_y[y][x] = 0;
211     }
212 }

```

```

210          }
211      }
212  }
213  // Scale the gradients by the grid spacing
214  dz_x[y][x] /= hx;
215  dz_y[y][x] /= hy;
216 }
217
218
219 // Compute the mean gradient at a point using its neighborhood
220 function meanGradient(x, y) {
221   // Check if already calculated
222   if(dz_x_m[y][x] !== 0 || dz_y_m[y][x] !== 0) return;
223   var sum_x = 0, sum_y = 0, count_x = 0, count_y = 0;
224   for(var p = 0; p < N_a; p++) {
225     for(var q = 0; q < N_a; q++) {
226       var nx = x + q - N_a_c;
227       var ny = y + p - N_a_c;
228       if(!(nx < 0 || nx >= m || ny < 0 || ny >= n)) {
229         if(dz_x[ny][nx] !== 0) {
230           sum_x += dz_x[ny][nx];
231           count_x++;
232         }
233         if(dz_y[ny][nx] !== 0) {
234           sum_y += dz_y[ny][nx];
235           count_y++;
236         }
237       }
238     }
239   }
240   dz_x_m[y][x] = count_x > 1 ? sum_x / count_x : sum_x;
241   dz_y_m[y][x] = count_y > 1 ? sum_y / count_y : sum_y;
242 }
243
244 for(var y = 0; y < n; y++) {
245   for(var x = 0; x < m; x++) {
246     pointGradient(x, y);
247   }
248 }
249 // Calculate mean gradient for all points
250 for(var y = 0; y < n; y++) {
251   for(var x = 0; x < m; x++) {
252     meanGradient(x, y);
253   }
254 }
255
256 return [dz_x_m, dz_y_m];
257 }
258
259 /**
260 * Interpolates grid data using the specified method.
261 * @param {Array} x - X coordinates.
262 * @param {Array} y - Y coordinates.
263 * @param {Array} z - Data values.
264 * @param {Array} xq - Query X coordinates.

```

```

265  * @param {Array} yq - Query Y coordinates.
266  * @param {string} [method="linear"] - Interpolation method.
267  * @param {Object} [opts_in] - Optional settings.
268  * @returns {Array[]} Interpolated grid [xq, yq, zq].
269  */
270 gridData(x, y, z, xq, yq, method = "linear", opts_in) {
271   var opts = {
272     N_a: 11,
273     k_e: 0.05,
274     Ngx: 5,
275     Ngy: 1,
276     extrap: true,
277     ...opts_in
278   };
279
280   function isBetween(A, a, b){
281     var B = [];
282     for(var i = 0; i < A.length; i += 1){
283       if(A[i] >= (a - (b - a) * opts.k_e) &&
284           A[i] <= (b + (b - a) * opts.k_e)) B.push(i);
285     }
286     return B;
287   }
288
289   var q_ids = [];
290   var i_ids = [];
291
292   var limits_x = linspace(xq[0], end(xq), opts.Ngx + 1);
293   var limits_y = linspace(yq[0], end(yq), opts.Ngy + 1);
294
295   for(var i = 0; i < opts.Ngx; i++){
296     var i_idx = isBetween(x, limits_x[i], limits_x[i + 1]);
297     for(var j = 0; j < opts.Ngy; j++){
298       var i_idy = isBetween(y, limits_y[j], limits_y[j + 1]);
299       var i_id = this.jsl.array.arrayIntersect(i_idx, i_idy);
300       i_ids.push(structuredClone(i_id));
301
302       var q_ids_ij = [];
303       for(var k = 0; k < xq.length; k++){
304         for(var m = 0; m < yq.length; m++){
305           if(limits_x[i] <= xq[k] &&
306               xq[k] <= limits_x[i + 1] &&
307               limits_y[j] <= yq[m] &&
308               yq[m] <= limits_y[j + 1]){
309             q_ids_ij.push([k, m]);
310           }
311         }
312       }
313       q_ids.push([...q_ids_ij]);
314     }
315   }
316
317   // Interpolation
318   var zq = this.jsl.array.NaNs(yq.length, xq.length);
319   if(method === 'nearest') {

```

```

320     for (var i = 0; i < q_ids.length; i++) {
321         var gc_q_ids = q_ids[i];
322         var gc_i_ids = i_ids[i];
323         for (var j = 0; j < gc_q_ids.length; j++) {
324             var L_min;
325             for (var k = 0; k < gc_i_ids.length; k++) {
326                 var L = Math.sqrt(Math.pow(x[gc_i_ids[k]] - xq[gc_q_ids[j][0]], 2)
327                                 +
328                                 Math.pow(y[gc_i_ids[k]] - yq[gc_q_ids[j][1]], 2));
329                 if (k == 0 || L < L_min) {
330                     L_min = L;
331                     zq[gc_q_ids[j][1]][gc_q_ids[j][0]] = z[gc_i_ids[k]];
332                 }
333             }
334         }
335     } else if (method == 'linear') {
336         for (var i = 0; i < q_ids.length; i++) {
337             var gc_q_ids = q_ids[i];
338             var gc_i_ids = i_ids[i];
339
340             // Delaunay triangulation
341             var points = [];
342             for (var j = 0; j < gc_i_ids.length; j++) {
343                 points.push([x[gc_i_ids[j]], y[gc_i_ids[j]], z[gc_i_ids[j]])];
344             }
345             var triangles = this.jsl.geometry.delaunayTriangulation(points);
346
347             // Locate point (xq, yq) in triangle and calculate zq
348             for (var j = 0; j < gc_q_ids.length; j++) {
349                 var point = [xq[gc_q_ids[j][0]], yq[gc_q_ids[j][1]], 0];
350                 for (var k = 0; k < triangles.length; k++) {
351                     if (triangles[k].contains(point)) {
352                         zq[gc_q_ids[j][1]][gc_q_ids[j][0]] = triangles[k].valueAt(point);
353                         break;
354                     }
355                 }
356             }
357         }
358
359         if (opts.extrap) {
360             // Extrapolation
361             var zq0 = structuredClone(zq);
362
363             // Calculate dz/dx and dz/dy for grid
364             var [dz_x, dz_y] = this.gridGradient(xq, yq, zq, opts.N_a);
365
366             for (var i = 0; i < q_ids.length; i++) {
367                 var gc_q_ids = q_ids[i];
368
369                 // Find nearest point with dz/dx and dz/dy
370                 for (var j = 0; j < gc_q_ids.length; j++) {
371                     var point = [
372                         xq[gc_q_ids[j][0]],

```

```

373         yq[ gc_q_ids[ j ][ 1 ] ] ,
374         zq[ gc_q_ids[ j ][ 1 ][ gc_q_ids[ j ][ 0 ] ]
375     ];
376     if( this.jsl._isNaN( point[ 2 ] ) ) {
377         var L_min;
378         var k_min = -1;
379         for( var k = 0; k < gc_q_ids.length; k++ ) {
380             if( !this.jsl._isNaN( zq0[ gc_q_ids[ k ][ 1 ][ gc_q_ids[ k ][ 0 ] ] ] ) ) {
381                 var L = Math.sqrt( Math.pow( point[ 0 ] - xq[ gc_q_ids[ k ][ 0 ] ], 2 )
382                               +
383                               Math.pow( point[ 1 ] - yq[ gc_q_ids[ k ][ 1 ] ], 2 ) );
384                 if( k_min === -1 || L < L_min ) {
385                     L_min = L;
386                     k_min = k;
387                 }
388             }
389             if( k_min >= 0 ) {
390                 // Calculate dz and z
391                 var dz_x_i = dz_x[ gc_q_ids[ k_min ][ 1 ][ gc_q_ids[ k_min ][ 0 ] ];
392                 var dz_y_i = dz_y[ gc_q_ids[ k_min ][ 1 ][ gc_q_ids[ k_min ][ 0 ] ];
393
394                 var dx = xq[ gc_q_ids[ k_min ][ 0 ] ] - point[ 0 ];
395                 var dy = yq[ gc_q_ids[ k_min ][ 1 ] ] - point[ 1 ];
396                 zq[ gc_q_ids[ j ][ 1 ][ gc_q_ids[ j ][ 0 ] ] =
397                     zq[ gc_q_ids[ k_min ][ 1 ][ gc_q_ids[ k_min ][ 0 ] ] -
398                     dx * dz_x_i - dy * dz_y_i;
399             }
400         }
401     }
402 }
403 } else {
404     this.jsl.env.error( '@gridData: '+language.string(235) );
405 }
406 return [xq, yq, zq];
407 }

410 /**
411 * Calculates the output of a bilinear function based on input value,
412 * midpoint, and mid-value.
413 * @param {number} x - The input value for the function.
414 * @param {number} midPoint - The midpoint of the function where the slope
415 * changes.
416 * @param {number} midValue - The value of the function at the midpoint.
417 * @returns {number} The output value of the bilinear function.
418 */
419 bilinearFunction( x, midPoint, midValue ) {
420     var sign = 1;
421     if( x < 0 ) {
422         sign = -1;
423     }
424     x = Math.abs( x );
425     if( x <= midPoint ) {

```

```

424     return sign * (midValue / midPoint) * x;
425   } else {
426     return sign * (((1 - midValue) / (1 - midPoint)) * (x - midPoint) +
427                   midValue);
428   }
429
430 /**
431 * Generates a random number between a specified range.
432 * @param {Number} [min=0] The lower bound of the range.
433 * @param {Number} [max] The upper bound of the range.
434 * @returns {Number} A random number within the specified range.
435 */
436 random(min, max) {
437   if(isNaN(Number(max))) return Math.random();
438   if(isNaN(Number(min))) min = 0;
439   return Math.random() * (max - min) + min;
440 }
441
442 /**
443 * Generates a random integer within a specified range.
444 * @param {Number} [min=0] The lower bound of the range.
445 * @param {Number} [max] The upper bound of the range.
446 * @returns {Number} A random integer within the specified range.
447 */
448 randInt(min, max) {
449   if(isNaN(Number(max))) return NaN;
450   if(isNaN(Number(min))) min = 0;
451   return Math.floor(Math.random() * (max - min + 1) + min);
452 }
453
454 /**
455 * Computes the arc cosine of x, with the result in degrees.
456 * @param {Number} x The value to compute the arc cosine for.
457 * @returns {Number} The arc cosine of x in degrees.
458 */
459 acosd(x) {
460   return this.jsl.env.math.dotMultiply(this.jsl.env.math.acos(x), this.r2d);
461 }
462
463 /**
464 * Computes the arc cotangent of x, with the result in degrees.
465 * @param {Number} x The value to compute the arc cotangent for.
466 * @returns {Number} The arc cotangent of x in degrees.
467 */
468 acotd(x) {
469   return this.jsl.env.math.dotMultiply(this.jsl.env.math.acot(x), this.r2d);
470 }
471
472 /**
473 * Computes the arc cosecant of x, with the result in degrees.
474 * @param {Number} x The value to compute the arc cosecant for.
475 * @returns {Number} The arc cosecant of x in degrees.
476 */
477 acscd(x) {

```

```
478     return this.jsl.env.math.dotMultiply(this.jsl.env.math.acsc(x), this.r2d);  
479 }  
480  
481 /**
482 * Computes the arc secant of x, with the result in degrees.  
483 * @param {Number} x The value to compute the arc secant for.  
484 * @returns {Number} The arc secant of x in degrees.  
485 */  
486 asecd(x) {  
    return this.jsl.env.math.dotMultiply(this.jsl.env.math.asec(x), this.r2d);  
}  
487  
488 /**
489 * Computes the arc sine of x, with the result in degrees.  
490 * @param {Number} x The value to compute the arc sine for.  
491 * @returns {Number} The arc sine of x in degrees.  
492 */  
493 asind(x) {  
    return this.jsl.env.math.dotMultiply(this.jsl.env.math.asin(x), this.r2d);  
}  
494  
495 /**
496 * Computes the arc tangent of x, with the result in degrees.  
497 * @param {Number} x The value to compute the arc tangent for.  
498 * @returns {Number} The arc tangent of x in degrees.  
499 */  
500 atan(x) {  
    return this.jsl.env.math.dotMultiply(this.jsl.env.math.atan(x), this.r2d);  
}  
501  
502 /**
503 * Computes the arc tangent of the quotient of its arguments, with the  
504 * result in degrees.  
505 * @param {Number} y The y coordinate.  
506 * @param {Number} x The x coordinate.  
507 * @returns {Number} The arc tangent of y/x in degrees.  
508 */  
509 atan2d(y, x) {  
    return this.jsl.env.math.dotMultiply(this.jsl.env.math.atan2(y, x), this.r2d);  
}  
510  
511 /**
512 * Computes the cosine of x, where x is in degrees.  
513 * @param {Number} x The angle in degrees.  
514 * @returns {Number} The cosine of x in degrees.  
515 */  
516 cosd(x) {  
    return this.jsl.env.math.cos(this.jsl.env.math.dotMultiply(x, this.d2r));  
}  
517  
518 /**
519 * Computes the cotangent of x, where x is in degrees.  
520 * @param {Number} x The angle in degrees.  
521 * @returns {Number} The cotangent of x.  
522 */  
523 cotd(x) {  
    return this.jsl.env.math.dotMultiply(x, this.d2r);  
}  
524  
525 /**
526 * Computes the secant of x, where x is in degrees.  
527 * @param {Number} x The angle in degrees.  
528 * @returns {Number} The secant of x.  
529 */  
530 secd(x) {  
    return this.jsl.env.math.dotMultiply(this.jsl.env.math.sec(x), this.r2d);  
}
```

```

531   */
532   cotd(x) {
533     return this.jsl.env.math.cot(this.jsl.env.math.dotMultiply(x, this.d2r));
534   }
535
536 /**
537 * Computes the cosecant of x, where x is in degrees.
538 * @param {Number} x The angle in degrees.
539 * @returns {Number} The cosecant of x.
540 */
541 cscd(x) {
542   return this.jsl.env.math.csc(this.jsl.env.math.dotMultiply(x, this.d2r));
543 }
544
545 /**
546 * Computes the secant of x, where x is in degrees.
547 * @param {Number} x The angle in degrees.
548 * @returns {Number} The secant of x.
549 */
550 secd(x) {
551   return this.jsl.env.math.sec(this.jsl.env.math.dotMultiply(x, this.d2r));
552 }
553
554 /**
555 * Computes the sine of x, where x is in degrees.
556 * @param {Number} x The angle in degrees.
557 * @returns {Number} The sine of x.
558 */
559 sind(x) {
560   return this.jsl.env.math.sin(this.jsl.env.math.dotMultiply(x, this.d2r));
561 }
562
563 /**
564 * Computes the tangent of x, where x is in degrees.
565 * @param {Number} x The angle in degrees.
566 * @returns {Number} The tangent of x.
567 */
568 tand(x) {
569   return this.jsl.env.math.tan(this.jsl.env.math.dotMultiply(x, this.d2r));
570 }
571
572 /**
573 * Computes the characteristic polynomial of a matrix or the polynomial from
574 * roots.
575 * @param {Array} A - If 'A' is a matrix (2D array), computes its
576 * characteristic polynomial.
577 * If 'A' is an array of roots, computes the polynomial
578 * with those roots.
579 * @returns {Array} - Coefficients of the resulting polynomial.
580 */
581 poly(A) {
582   if(A[0].length) {
583     return charpoly(A);
584   }
585 }
```

```

583     let p = [1];
584
585     for (const root of A) {
586         p = p.map((coef) => coef * -root)
587             .concat(0)
588             .map((coef, index, arr) => coef + (arr[index + 1] || 0));
589     }
590
591     return p;
592 }
593
594 /**
595 * Fits a polynomial of degree n to the given data points and returns the
596 * coefficients (highest degree first).
597 * @param {number[]} x - The array of x-values.
598 * @param {number[]} y - The array of y-values corresponding to each x-value
599 *
600 * @param {number} n - The degree of the polynomial to fit.
601 * @returns {number[]} The coefficients of the fitted polynomial in
602 * descending order.
603 */
604 polyfit(x, y, n) {
605     var p = new this.jsl.env.PolynomialRegression(x, y, n);
606     return p.coefficients.reverse(); // Reverse to match MATLAB's coefficient
607     order
608 }
609
610 /**
611 * Evaluates a polynomial with given coefficients at specified x-values.
612 * @param {number[]} p - The coefficients of the polynomial in descending
613 * order.
614 * @param {number[]} x_in - The array of x-values at which to evaluate the
615 * polynomial.
616 * @returns {number[]} The resulting y-values after evaluating the
617 * polynomial at x_in.
618 */
619 polyval(p, x_in) {
620     var y_out = [];
621     y_out.length = x_in.length;
622     for(var k = 0; k < x_in.length; k++) {
623         var y = p[0];
624         for(var i = 1; i < p.length; i++) {
625             y = y * x_in[k] + p[i];
626         }
627         y_out[k] = y;
628     }
629     return y_out;
630 }
631
632 /**
633 * Computes the roots of a polynomial with the given coefficients.
634 * @param {number[]} p - Array of polynomial coefficients, ordered from
635 * highest degree to constant term.
636 * @returns {number[]} Array of roots (real or complex) of the polynomial.
637 */

```

```

630   roots(p) {
631     return this.jsl.env.native_module.roots(p);
632   }
633
634 /**
635 * Generates a string representation of a polynomial based on the provided
636 coefficients and options.
637 * @param {number[]} p - An array of polynomial coefficients, ordered from
638 highest degree to constant term.
639 * @param {Object} [opts] - Optional settings for the polynomial string.
640 * @param {string} [opts.x_symbol='x'] - The symbol to use for the variable
641 x.
642 * @param {string} [opts.y_symbol='y'] - The symbol to use for the variable
643 y.
644 * @param {number} [opts.precision=7] - The number of decimal places for
645 coefficients.
646 * @param {string} [opts.lang] - The language format for the output ('tex',
647 'c', etc.).
648 * @returns {string} The formatted polynomial string.
649 */
650 polystr(p, opts) {
651   let degree = p.length - 1;
652   let x_symbol = 'x';
653   let y_symbol = 'y';
654   let precision = 7;
655   let lang;
656   if(opts) {
657     if(opts.hasOwnProperty('x_symbol')) {
658       x_symbol = opts.x_symbol;
659     }
660     if(opts.hasOwnProperty('y_symbol')) {
661       y_symbol = opts.y_symbol;
662     }
663   }
664   let str = y_symbol+ '= ';
665
666   for(let i = 0; i < p.length; i++) {
667     let current_power = degree - i;
668     let coef = p[i];
669
670     // Skip terms with zero coefficient
671     if(coef === 0) {
672       continue;
673     }
674
675     // Handle the sign
676     let sign_str = '';
677     if(coef > 0 && i !== 0) {
678       sign_str = '+';
```

```

679     } else if(coef < 0) {
680         sign_str = ', - ';
681         coef = -coef; // Convert to positive for display
682     }
683
684     // Form the term based on the power of x
685     let term_str;
686     if(current_power > 1) {
687         if(lang == 'tex') {
688             term_str = '${sign_str}\\num\\${coef.toExponential(precision)}\\',
689             `${x_symbol}^\\${current_power}`;
690         } else if(lang == 'c') {
691             term_str = `${sign_str}${coef.toExponential(precision)}*pow(${x_symbol},
692             ${current_power})`;
693         } else {
694             term_str = `${sign_str}${coef.toExponential(precision)}*${x_symbol}^
695             ${current_power}`;
696         }
697     } else if(current_power === 1) {
698         if(lang == 'tex') {
699             term_str = `${sign_str}\\num\\${coef.toExponential(precision)}\\`,
700             `${x_symbol}`;
701         } else {
702             term_str = `${sign_str}${coef.toExponential(precision)}*${x_symbol}`;
703         }
704     } else {
705         if(lang == 'tex') {
706             term_str = `${sign_str}\\num\\${coef.toExponential(precision)} `;
707         } else {
708             term_str = `${sign_str}${coef.toExponential(precision)} `;
709         }
710     }
711     str += term_str;
712 }
713
714 return str;
715
716 /**
717 * Generates a C language formatted string representation of a polynomial.
718 * @param {number[]} p - An array of polynomial coefficients, ordered from
719 * highest degree to constant term.
720 * @param {Object} [opts] - Optional settings for the polynomial string.
721 * @returns {string} The polynomial string formatted for C language.
722 */
723 polystrc(p, opts) {
724     if(opts) {
725         opts.lang = 'c';
726     } else {
727         opts = {lang: 'c'};
728     }

```

```

728     return polystr(p, opts);
729 }
730
731 /**
732 * Generates a LaTeX formatted string representation of a polynomial.
733 * @param {number[]} p - An array of polynomial coefficients , ordered from
734 * highest degree to constant term.
735 * @param {Object} [opts] - Optional settings for the polynomial string.
736 * @returns {string} The polynomial string formatted for LaTeX.
737 */
738 polystrtex(p, opts) {
739   if(opts) {
740     opts.lang = 'tex';
741   } else {
742     opts = {lang: 'tex'};
743   }
744   return polystr(p, opts);
745 }
746 /**
747 * Filters out spikes in a sequence by replacing sudden large changes with
748 * the previous value.
749 * @param {number[]} x - The input sequence of numbers.
750 * @param {number} dx_max - The maximum allowed difference between
751 * consecutive elements before considering it a spike.
752 * @param {number} n - The maximum number of consecutive spikes to correct .
753 * @returns {number[]} The filtered sequence with spikes removed.
754 */
755 spikeFilter(x, dx_max, n) {
756   var c = 0;
757   if(x.length > 1) {
758     for(var i = 1; i < x.length; i++) {
759       if(Math.abs(x[i]-x[i-1]) > dx_max && c < n){
760         c = c+1;
761         x[i] = x[i-1];
762       } else {
763         c = 0;
764       }
765     }
766   }
767   return x;
768 }
769 /**
770 * Calculates the magnitude (absolute value) of a complex number or a real
771 * number .
772 * @param {number|Object} num - A real number or an object with 'real' and '
773 * imag' properties.
774 * @returns {number} The magnitude of the number.
775 */
776 magnitude(num) {
777   if(typeof num === 'object' && num.real !== undefined && num.imag !==
778     undefined) {
779     return Math.hypot(num.real, num.imag);

```

```

777     } else {
778         return Math.abs(num);
779     }
780 }
781 /**
782 * Compares two numbers (real or complex) according to Octave's rules.
783 *
784 * @param {number|Object} a - First number to compare.
785 * @param {number|Object} b - Second number to compare.
786 * @returns {number} -1 if a < b, 1 if a > b, 0 if equal.
787 */
788 compareComplex(a, b) {
789     const mag_A = magnitude(a);
790     const mag_B = magnitude(b);
791
792     if(mag_A < mag_B) return -1;
793     if(mag_A > mag_B) return 1;
794
795     // Magnitudes are equal; compare real parts
796     const real_A = (typeof a === 'object') ? a.real : a;
797     const real_B = (typeof b === 'object') ? b.real : b;
798
799     if(real_A < real_B) return -1;
800     if(real_A > real_B) return 1;
801
802     // Real parts are equal; compare imaginary parts
803     const imag_A = (typeof a === 'object') ? a.imag : 0;
804     const imag_B = (typeof b === 'object') ? b.imag : 0;
805
806     if(imag_A < imag_B) return -1;
807     if(imag_A > imag_B) return 1;
808
809     // Numbers are equal
810     return 0;
811 }
812 /**
813 * Finds the minimum value in an array of numbers, which can include both
814 * real numbers and complex numbers.
815 * Complex numbers are represented as objects with 'real' and 'imag'
816 * properties.
817 * The comparison follows Octave's rules:
818 * 1. Compare magnitudes (absolute values) of the numbers.
819 * 2. If magnitudes are equal, compare real parts.
820 * 3. If real parts are equal, compare imaginary parts.
821 *
822 * @param {Array<number|Object>} arr - An array of numbers or complex number
823 * objects.
824 * @returns {number|Object} The minimum value found in the array.
825 * @throws {TypeError} If the input is not an array.
826 */
827 min(arr) {
828     if(!Array.isArray(arr)) {
829         this.jsl.env.error('@min: '+language.string(190));

```

```

829     return;
830   }
831   arr = arr.filter(num => !isNaN(num));
832
833   // Separate real numbers and complex numbers
834   const real_numbers = [];
835   const complex_numbers = [];
836   for(const num of arr) {
837     if(typeof num === 'object' && num.real !== undefined && num.imag !==
838       undefined) {
839       complex_numbers.push(num);
840     } else {
841       real_numbers.push(num);
842     }
843   }
844
845   // Find min among real numbers using existing function
846   let min_real = real_numbers.length > 0 ? this.jsl.env.math.min(
847     real_numbers) : null;
848
849   // Find min among complex numbers
850   let min_complex = complex_numbers.length > 0 ? complex_numbers[0] : null;
851   for(let i = 1; i < complex_numbers.length; i++) {
852     if(compareComplex(complex_numbers[i], min_complex) < 0) {
853       min_complex = complex_numbers[i];
854     }
855   }
856
857   // Compare min_real and min_complex
858   if(min_real !== null && min_complex !== null) {
859     return compareComplex(min_real, min_complex) < 0 ? min_real :
860       min_complex;
861   } else if(min_real !== null) {
862     return min_real;
863   } else {
864     return min_complex;
865   }
866
867   /**
868    * Finds the maximum value in an array of numbers, which can include both
869    * real numbers and complex numbers.
870    * Complex numbers are represented as objects with 'real' and 'imag'
871    * properties.
872    * The comparison follows Octave's rules:
873    * 1. Compare magnitudes (absolute values) of the numbers.
874    * 2. If magnitudes are equal, compare real parts.
875    * 3. If real parts are equal, compare imaginary parts.
876    *
877    * @param {Array<number|Object>} arr - An array of numbers or complex number
878    * objects.
879    * @returns {number|Object} The maximum value found in the array.
880    * @throws {TypeError} If the input is not an array.
881    */
882
883   max(arr) {

```

```

878     if (!Array.isArray(arr)) {
879       this.jsl.env.error(`@max: ${language.string(190)}`);
880       return;
881     }
882     arr = arr.filter(num => !isNaN(num));
883
884     // Separate real numbers and complex numbers
885     const real_numbers = [];
886     const complex_numbers = [];
887     for(const num of arr) {
888       if(typeof num === 'object' && num.real !== undefined && num.imag !==
889         undefined) {
890         complex_numbers.push(num);
891       } else {
892         real_numbers.push(num);
893       }
894     }
895     // Find max among real numbers using existing function
896     let max_real = real_numbers.length > 0 ? this.jsl.env.math.max(
897       real_numbers) : null;
898
899     // Find max among complex numbers
900     let max_complex = complex_numbers.length > 0 ? complex_numbers[0] : null;
901     for(let i = 1; i < complex_numbers.length; i++) {
902       if(compareComplex(complex_numbers[i], max_complex) > 0) {
903         max_complex = complex_numbers[i];
904       }
905     }
906     // Compare max_real and max_complex
907     if(max_real !== null && max_complex !== null) {
908       return compareComplex(max_real, max_complex) > 0 ? max_real :
909         max_complex;
910     } else if(max_real !== null) {
911       return max_real;
912     } else {
913       return max_complex;
914     }
915
916    /**
917     * Extracts the real part of a number or an array of numbers.
918     * Handles mixed inputs containing both real numbers and complex numbers.
919     *
920     * @param {number|Object|Array<number|Object>} input - A number, complex
921     *   number object, or array thereof.
922     * @returns {number|Array<number>} The real part(s) of the input.
923     * @throws {TypeError} If the input is not a number, complex number object,
924     *   or array.
925     */
926     real(input) {
927       if(Array.isArray(input)) {
928         return input.map(real);
929       } else if(typeof input === 'object' && input !== null && 'real' in input)

```

```

928     && 'imag' in input) {
929     return real(input.real);
930   } else if(typeof input === 'number') {
931     return input;
932   } else {
933     this.jsl.env.error('@real: '+language.string(192));
934   }
935 }
936 /**
937 * Extracts the imaginary part of a number or an array of numbers.
938 * Handles mixed inputs containing both real numbers and complex numbers.
939 *
940 * @param {number|Object|Array<number|Object>} input - A number, complex
941 *       number object, or array thereof.
942 * @returns {number|Array<number>} The imaginary part(s) of the input.
943 * @throws {TypeError} If the input is not a number, complex number object,
944 *       or array.
945 */
946 imag(input) {
947   if(Array.isArray(input)) {
948     return input.map(imag);
949   } else if(typeof input === 'object' && input !== null && 'real' in input
950     && 'imag' in input) {
951     return imag(input.imag);
952   } else if(typeof input === 'number') {
953     return 0;
954   } else {
955     this.jsl.env.error('@imag: '+language.string(192));
956   }
957   return false;
958 }
959 /**
960 * Performs cumulative trapezoidal integration on the provided data.
961 * @param {...any} args - Arguments required for cumulative trapezoidal
962 *       integration.
963 * @returns {any} The result of the cumulative trapezoidal integration.
964 */
965 cumtrapz(...args) {
966   return this.jsl.env.native_module.cumtrapz(...args);
967 }
968 /**
969 * Performs trapezoidal integration on the provided data.
970 * @param {...any} args - Arguments required for trapezoidal integration.
971 * @returns {any} The result of the trapezoidal integration.
972 */
973 trapz(...args) {
974   return this.jsl.env.native_module.trapz(...args);
975 }
976 /**
977 * Compute the mean squared error (MSE) between two arrays.
978 * @param {Array<number>} A - The first array.

```

```

978  * @param {Array<number>} B - The second array.
979  * @returns {number} - The mean squared error between A and B.
980  * @throws {Error} - If A and B have different lengths or are not arrays.
981  */
982 mse(A, B) {
983   if(!Array.isArray(A) || !Array.isArray(B)) {
984     throw new Error("Both inputs must be arrays.");
985   }
986
987   if(A.length !== B.length) {
988     throw new Error("Input arrays must have the same length.");
989   }
990
991   const n = A.length;
992   const mse = A.reduce((sum, a, i) => {
993     const diff = a - B[i];
994     return sum + diff * diff;
995   }, 0) / n;
996
997   return mse;
998 }
999
1000 /**
1001 * Calculates the coefficients of the characteristic polynomial of a square
1002 * matrix.
1003 * @param {number[][]} matrix - A square matrix (2D array) for which the
1004 *   characteristic polynomial is computed.
1005 * @returns {number[]} - An array of coefficients of the characteristic
1006 *   polynomial.
1007 * @throws {Error} - Throws an error if the input is not a square matrix or
1008 *   has less than 2 rows/columns.
1009 */
1010 charpoly(matrix) {
1011   const n = matrix.length;
1012   const m = matrix[0].length;
1013
1014   if(n !== m || n < 1) {
1015     throw new Error("Argument 'matrix' must be a square matrix.");
1016   }
1017
1018   let p = ones(n+1);
1019   let a1 = [...matrix];
1020   for(let k = 2; k <= n; k++) {
1021     p[k-1] = -1 * sum(this.jsl.env.math.diag(a1)) / (k - 1);
1022     a1 = this.jsl.env.math.multiply(matrix, plus(a1,
1023       this.jsl.env.math.multiply(p[k-1],
1024       this.jsl.env.math.diag(ones(n)))));
1025   }
1026
1027   p[n] = -1 * sum(this.jsl.env.math.diag(a1)) / n;
1028

```

```

1029     return p;
1030 }
1031 }
1032
1033 exports.PRDC_JSLAB_LIB_MATH = PRDC_JSLAB_LIB_MATH;

```

Listing 110 - math.js

```

1  /**
2  * @file JSLAB library math.js doc.
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9 * Class for JSLAB math.js doc.
10 */
11 class PRDC_JSLAB_MATHJS_DOC {
12
13     constructor() {
14
15         /**
16         * Test whether a value is a Complex number.
17         * @name isComplex
18         * @kind function
19         * @param { * } x - The value to test.
20         * @returns { boolean } Returns true if 'x' is a Complex number, false
21         * otherwise.
22         * @memberof PRDC_JSLAB_MATHJS_DOC
23         */
24
25         /**
26         * Test whether a value is a BigNumber.
27         * @name isBigNumber
28         * @kind function
29         * @param { * } x - The value to test.
30         * @returns { boolean } Returns true if 'x' is a BigNumber, false
31         * otherwise.
32         * @memberof PRDC_JSLAB_MATHJS_DOC
33         */
34
35         /**
36         * Test whether a value is a Fraction.
37         * @name isFraction
38         * @kind function
39         * @param { * } x - The value to test.
40         * @returns { boolean } Returns true if 'x' is a Fraction, false otherwise
41
42         /**
43         * Test whether a value is a Unit.
44         * @name isUnit
45         * @kind function

```

```
46     * @param { * } x - The value to test.
47     * @returns { boolean } Returns true if 'x' is a Unit, false otherwise.
48     * @memberof PRDC_JSLAB_MATHJS_DOC
49     */
50
51 /**
52  * Test whether a value is a Matrix.
53  * @name isMatrix
54  * @kind function
55  * @param { * } x - The value to test.
56  * @returns { boolean } Returns true if 'x' is a Matrix, false otherwise.
57  * @memberof PRDC_JSLAB_MATHJS_DOC
58  */
59
60 /**
61  * Test whether a value is a collection (Array or Matrix).
62  * @name isCollection
63  * @kind function
64  * @param { * } x - The value to test.
65  * @returns { boolean } Returns true if 'x' is an Array or Matrix, false
66  * otherwise.
67  * @memberof PRDC_JSLAB_MATHJS_DOC
68  */
69
70 /**
71  * Test whether a value is a DenseMatrix.
72  * @name isDenseMatrix
73  * @kind function
74  * @param { * } x - The value to test.
75  * @returns { boolean } Returns true if 'x' is a DenseMatrix, false
76  * otherwise.
77  * @memberof PRDC_JSLAB_MATHJS_DOC
78  */
79
80 /**
81  * Test whether a value is a SparseMatrix.
82  * @name isSparseMatrix
83  * @kind function
84  * @param { * } x - The value to test.
85  * @returns { boolean } Returns true if 'x' is a SparseMatrix, false
86  * otherwise.
87  * @memberof PRDC_JSLAB_MATHJS_DOC
88  */
89
90 /**
91  * Test whether a value is a Range.
92  * @name isRange
93  * @kind function
94  * @param { * } x - The value to test.
95  * @returns { boolean } Returns true if 'x' is a Range, false otherwise.
96  * @memberof PRDC_JSLAB_MATHJS_DOC
97  */
98
99 /**
100 * Test whether a value is an Index.
```

```

98 * @name isIndex
99 * @kind function
100 * @param { * } x - The value to test .
101 * @returns { boolean } Returns true if ‘x’ is an Index , false otherwise .
102 * @memberof PRDC_JSLAB_MATHJS_DOC
103 */
104
105 /**
106 * Test whether a value is a boolean.
107 * @name isBoolean
108 * @kind function
109 * @param { * } x - The value to test .
110 * @returns { boolean } Returns true if ‘x’ is a boolean , false otherwise .
111 * @memberof PRDC_JSLAB_MATHJS_DOC
112 */
113
114 /**
115 * Test whether a value is a ResultSet .
116 * @name isResultSet
117 * @kind function
118 * @param { * } x - The value to test .
119 * @returns { boolean } Returns true if ‘x’ is a ResultSet , false
120 * @memberof PRDC_JSLAB_MATHJS_DOC
121 */
122
123 /**
124 * Test whether a value is a Help object .
125 * @name isHelp
126 * @kind function
127 * @param { * } x - The value to test .
128 * @returns { boolean } Returns true if ‘x’ is a Help object , false
129 * @memberof PRDC_JSLAB_MATHJS_DOC
130 */
131
132 /**
133 * Test whether a value is a Date .
134 * @name isDate
135 * @kind function
136 * @param { * } x - The value to test .
137 * @returns { boolean } Returns true if ‘x’ is a Date object , false
138 * @memberof PRDC_JSLAB_MATHJS_DOC
139 */
140
141 /**
142 * Test whether a value is a RegExp .
143 * @name isRegExp
144 * @kind function
145 * @param { * } x - The value to test .
146 * @returns { boolean } Returns true if ‘x’ is a RegExp object , false
147 * @memberof PRDC_JSLAB_MATHJS_DOC
148 */

```

```
149
150  /**
151   * Test whether a value is an AccessorNode.
152   * @name isAccessorNode
153   * @kind function
154   * @param { *} x - The value to test.
155   * @returns { boolean } Returns true if 'x' is an AccessorNode, false
156   * otherwise.
157   * @memberof PRDC_JSLAB_MATHJS_DOC
158   */
159
160 /**
161  * Test whether a value is an ArrayNode.
162  * @name isArrayNode
163  * @kind function
164  * @param { *} x - The value to test.
165  * @returns { boolean } Returns true if 'x' is an ArrayNode, false
166  * otherwise.
167  * @memberof PRDC_JSLAB_MATHJS_DOC
168  */
169
170 /**
171  * Test whether a value is an AssignmentNode.
172  * @name isAssignmentNode
173  * @kind function
174  * @param { *} x - The value to test.
175  * @returns { boolean } Returns true if 'x' is an AssignmentNode, false
176  * otherwise.
177  * @memberof PRDC_JSLAB_MATHJS_DOC
178  */
179
180 /**
181  * Test whether a value is a BlockNode.
182  * @name isBlockNode
183  * @kind function
184  * @param { *} x - The value to test.
185  * @returns { boolean } Returns true if 'x' is a BlockNode, false
186  * otherwise.
187  * @memberof PRDC_JSLAB_MATHJS_DOC
188  */
189
190 /**
191  * Test whether a value is a ConditionalNode.
192  * @name isConditionalNode
193  * @kind function
194  * @param { *} x - The value to test.
195  * @returns { boolean } Returns true if 'x' is a ConditionalNode, false
196  * otherwise.
197  * @memberof PRDC_JSLAB_MATHJS_DOC
198  */
199
200 /**
201  * Test whether a value is a ConstantNode.
202  * @name isConstantNode
203  * @kind function
```

```
199     * @param { *} x - The value to test.
200     * @returns { boolean } Returns true if 'x' is a ConstantNode, false
201     * otherwise.
202     * @memberof PRDC_JSLAB_MATHJS_DOC
203     */
204
205     /**
206      * Test whether a value is a FunctionAssignmentNode.
207      * @name isFunctionAssignmentNode
208      * @kind function
209      * @param { *} x - The value to test.
210      * @returns { boolean } Returns true if 'x' is a FunctionAssignmentNode,
211      * false otherwise.
212      * @memberof PRDC_JSLAB_MATHJS_DOC
213      */
214
215     /**
216      * Test whether a value is a FunctionNode.
217      * @name isFunctionNode
218      * @kind function
219      * @param { *} x - The value to test.
220      * @returns { boolean } Returns true if 'x' is a FunctionNode, false
221      * otherwise.
222      * @memberof PRDC_JSLAB_MATHJS_DOC
223      */
224
225     /**
226      * Test whether a value is an IndexNode.
227      * @name isIndexNode
228      * @kind function
229      * @param { *} x - The value to test.
230      * @returns { boolean } Returns true if 'x' is an IndexNode, false
231      * otherwise.
232      * @memberof PRDC_JSLAB_MATHJS_DOC
233      */
234
235     /**
236      * Test whether a value is a Node.
237      * @name isNode
238      * @kind function
239      * @param { *} x - The value to test.
240      * @returns { boolean } Returns true if 'x' is a Node, false otherwise.
241      * @memberof PRDC_JSLAB_MATHJS_DOC
242      */
243
244     /**
245      * Test whether a value is an ObjectNode.
246      * @name isObjectNode
247      * @kind function
248      * @param { *} x - The value to test.
249      * @returns { boolean } Returns true if 'x' is an ObjectNode, false
250      * otherwise.
251      * @memberof PRDC_JSLAB_MATHJS_DOC
252      */
253
```

```
249  /**
250   * Test whether a value is an OperatorNode.
251   * @name isOperatorNode
252   * @kind function
253   * @param { *} x - The value to test.
254   * @returns { boolean } Returns true if 'x' is an OperatorNode, false
255   * otherwise.
256   * @memberof PRDC_JSLAB_MATHJS_DOC
257   */
258
259 /**
260  * Test whether a value is a ParenthesisNode.
261  * @name isParenthesisNode
262  * @kind function
263  * @param { *} x - The value to test.
264  * @returns { boolean } Returns true if 'x' is a ParenthesisNode, false
265  * otherwise.
266  * @memberof PRDC_JSLAB_MATHJS_DOC
267  */
268
269 /**
270  * Test whether a value is a RangeNode.
271  * @name isRangeNode
272  * @kind function
273  * @param { *} x - The value to test.
274  * @returns { boolean } Returns true if 'x' is a RangeNode, false
275  * otherwise.
276  * @memberof PRDC_JSLAB_MATHJS_DOC
277  */
278
279 /**
280  * Test whether a value is a RelationalNode.
281  * @name isRelationalNode
282  * @kind function
283  * @param { *} x - The value to test.
284  * @returns { boolean } Returns true if 'x' is a RelationalNode, false
285  * otherwise.
286  * @memberof PRDC_JSLAB_MATHJS_DOC
287  */
288
289 /**
290  * Test whether a value is a SymbolNode.
291  * @name isSymbolNode
292  * @kind function
293  * @param { *} x - The value to test.
294  * @returns { boolean } Returns true if 'x' is a SymbolNode, false
295  * otherwise.
296  * @memberof PRDC_JSLAB_MATHJS_DOC
297  */
298
299 /**
300  * Test whether a value is a Chain.
301  * @name isChain
302  * @kind function
303  * @param { *} x - The value to test.
```

```

299   * @returns { boolean } Returns true if 'x' is a Chain, false otherwise.
300   * @memberof PRDC_JSLAB_MATHJS_DOC
301   */
302
303  /**
304   * Subscribe to an event.
305   * @name on
306   * @kind function
307   * @param { string } event - The event name to subscribe to.
308   * @param { function } callback - The callback function to execute when
309   * the event occurs.
310   * @returns { void }
311   * @memberof PRDC_JSLAB_MATHJS_DOC
312   */
313
314  /**
315   * Unsubscribe from an event.
316   * @name off
317   * @kind function
318   * @param { string } event - The event name to unsubscribe from.
319   * @param { function } [callback] - The callback function to remove.
320   * @returns { void }
321   * @memberof PRDC_JSLAB_MATHJS_DOC
322   */
323
324  /**
325   * Subscribe to an event once.
326   * @name once
327   * @kind function
328   * @param { string } event - The event name to subscribe to.
329   * @param { function } callback - The callback function to execute when
330   * the event occurs.
331   * @returns { void }
332   * @memberof PRDC_JSLAB_MATHJS_DOC
333   */
334
335  /**
336   * Emit an event, triggering all bound callbacks.
337   * @name emit
338   * @kind function
339   * @param { string } event - The event name to emit.
340   * @param { * } [data] - The data to pass to the event handlers.
341   * @returns { void }
342   * @memberof PRDC_JSLAB_MATHJS_DOC
343   */
344
345  /**
346   * An object containing functions for parsing and evaluating expressions.
347   * @name expression
348   * @kind member
349   * @memberof PRDC_JSLAB_MATHJS_DOC
350   */
351
352  /**
353   * Import functions or constants into math.js.

```

```
352     * @name import
353     * @kind function
354     * @param { Object | Array } object - An object or array of objects with
355         functions or constants to import.
355     * @param { Object } [options] - Optional import options.
356     * @returns { void }
357     * @memberof PRDC_JSLAB_MATHJS_DOC
358   */
359
360 /**
361 * Create a new, isolated math.js instance.
362 * @name create
363 * @kind function
364 * @param { Object } [config] - Optional configuration options.
365 * @param { Function[] } [factories] - Optional list of factories to
366     include.
366 * @returns { PRDC_JSLAB_MATHJS_DOC } Returns a new instance of math.js.
367 * @memberof PRDC_JSLAB_MATHJS_DOC
368 */
369
370 /**
371 * Factory function to create new functions.
372 * @name factory
373 * @kind function
374 * @param { string } name - The name of the function.
375 * @param { string[] } dependencies - Array of dependency names.
376 * @param { function } create - Function to create the new function.
377 * @returns { function } Returns the created function.
378 * @memberof PRDC_JSLAB_MATHJS_DOC
379 */
380
381 /**
382 * Calculate the absolute value of a number.
383 * @name abs
384 * @kind function
385 * @param { number | BigNumber | Complex | Array | Matrix } x - A number
386     or array with numbers.
386 * @returns { number | BigNumber | Complex | Array | Matrix } Returns the
387     absolute value of 'x'.
388 * @memberof PRDC_JSLAB_MATHJS_DOC
389 */
390
391 /**
392 * A node representing access to a property or index.
393 * @name AccessorNode
394 * @kind function
394 * @param { Node } object - The object being accessed.
395 * @param { IndexNode } index - The index used to access the object.
396 * @returns { AccessorNode } Returns a new AccessorNode.
397 * @memberof PRDC_JSLAB_MATHJS_DOC
398 */
399
400 /**
401 * Calculate the inverse cosine of a value.
402 * @name acos
```

```
403 * @kind function
404 * @param { number | Complex | Array | Matrix } x - Function input.
405 * @returns { number | Complex | Array | Matrix } The arc cosine of 'x'.
406 * @memberof PRDC_JSLAB_MATHJS_DOC
407 */
408
409 /**
410 * Calculate the inverse hyperbolic cosine of a value.
411 * @name acosh
412 * @kind function
413 * @param { number | Complex | Array | Matrix } x - Function input.
414 * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
415 * cosine of 'x'.
416 * @memberof PRDC_JSLAB_MATHJS_DOC
417 */
418 /**
419 * Calculate the inverse cotangent of a value.
420 * @name acot
421 * @kind function
422 * @param { number | Complex | Array | Matrix } x - Function input.
423 * @returns { number | Complex | Array | Matrix } The inverse cotangent of
424 * 'x'.
425 * @memberof PRDC_JSLAB_MATHJS_DOC
426 */
427 /**
428 * Calculate the inverse hyperbolic cotangent of a value.
429 * @name acoth
430 * @kind function
431 * @param { number | Complex | Array | Matrix } x - Function input.
432 * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
433 * cotangent of 'x'.
434 * @memberof PRDC_JSLAB_MATHJS_DOC
435 */
436 /**
437 * Calculate the inverse cosecant of a value.
438 * @name acsc
439 * @kind function
440 * @param { number | Complex | Array | Matrix } x - Function input.
441 * @returns { number | Complex | Array | Matrix } The inverse cosecant of
442 * 'x'.
443 * @memberof PRDC_JSLAB_MATHJS_DOC
444 */
445 /**
446 * Calculate the inverse hyperbolic cosecant of a value.
447 * @name acsch
448 * @kind function
449 * @param { number | Complex | Array | Matrix } x - Function input.
450 * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
451 * cosecant of 'x'.
452 * @memberof PRDC_JSLAB_MATHJS_DOC
453 */
```

```

453
454  /**
455   * Add two scalar values, 'x + y'.
456   * @name addScalar
457   * @kind function
458   * @param { number | BigNumber | Fraction | Complex } x - First value to
459   *   add.
460   * @param { number | BigNumber | Fraction | Complex } y - Second value to
461   *   add.
462   * @returns { number | BigNumber | Fraction | Complex } Sum of 'x' and 'y'
463   *
464  /**
465   * Logical AND. Returns true if both inputs are true.
466   * @name and
467   * @kind function
468   * @param { boolean | Array | Matrix } x - First input.
469   * @param { boolean | Array | Matrix } y - Second input.
470   * @returns { boolean | Array | Matrix } Returns true when both 'x' and 'y'
471   *   are true.
472   * @memberof PRDC_JSLAB_MATHJS_DOC
473   */
474 /**
475  * Apply a function to each entry in a matrix or array.
476  * @name apply
477  * @kind function
478  * @param { Array | Matrix } x - The input array or matrix.
479  * @param { number } dim - The dimension along which to apply the function
480  *
481  * @param { function } callback - The function to apply.
482  * @returns { Array | Matrix } The result of applying the function along
483  *   the specified dimension.
484  * @memberof PRDC_JSLAB_MATHJS_DOC
485  */
486 /**
487  * Calculate the argument of a complex number.
488  * @name arg
489  * @kind function
490  * @param { number | Complex | Array | Matrix } x - Function input.
491  * @returns { number | Array | Matrix } The argument of 'x'.
492  * @memberof PRDC_JSLAB_MATHJS_DOC
493  */
494 /**
495  * An array node representing an array in the expression tree.
496  * @name ArrayNode
497  * @kind function
498  * @param { Node[] } items - An array of nodes.
499  * @returns { ArrayNode } Returns a new ArrayNode.
500  * @memberof PRDC_JSLAB_MATHJS_DOC
501  */

```

```
502
503  /**
504   * Calculate the inverse secant of a value.
505   * @name asec
506   * @kind function
507   * @param { number | Complex | Array | Matrix } x - Function input.
508   * @returns { number | Complex | Array | Matrix } The inverse secant of 'x'
509   *
510   * @memberof PRDC_JSLAB_MATHJS_DOC
511   */
512
513 /**
514  * Calculate the inverse hyperbolic secant of a value.
515  * @name asech
516  * @kind function
517  * @param { number | Complex | Array | Matrix } x - Function input.
518  * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
519  * secant of 'x'.
520  * @memberof PRDC_JSLAB_MATHJS_DOC
521  */
522
523 /**
524  * Calculate the inverse sine of a value.
525  * @name asin
526  * @kind function
527  * @param { number | Complex | Array | Matrix } x - Function input.
528  * @returns { number | Complex | Array | Matrix } The inverse sine of 'x'.
529  * @memberof PRDC_JSLAB_MATHJS_DOC
530  */
531
532 /**
533  * Calculate the inverse hyperbolic sine of a value.
534  * @name asinh
535  * @kind function
536  * @param { number | Complex | Array | Matrix } x - Function input.
537  * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
538  * sine of 'x'.
539  * @memberof PRDC_JSLAB_MATHJS_DOC
540  */
541
542 /**
543  * An assignment node representing variable assignment.
544  * @name AssignmentNode
545  * @kind function
546  * @param { Node } object - The symbol or AccessorNode to assign to.
547  * @param { Node } value - The value to assign.
548  * @returns { AssignmentNode } Returns a new AssignmentNode.
549  * @memberof PRDC_JSLAB_MATHJS_DOC
550  */
551
552 /**
553  * Calculate the inverse tangent of a value.
554  * @name atan
555  * @kind function
556  * @param { number | Complex | Array | Matrix } x - Function input.
```

```

554   * @returns { number | Complex | Array | Matrix } The inverse tangent of 'x'.
555   * @memberof PRDC_JSLAB_MATHJS_DOC
556   */
557
558 /**
559 * Calculate the inverse tangent of 'y/x'.
560 * @name atan2
561 * @kind function
562 * @param { number | BigNumber | Array | Matrix } y - Dividend.
563 * @param { number | BigNumber | Array | Matrix } x - Divisor.
564 * @returns { number | BigNumber | Array | Matrix } The inverse tangent of 'y/x'.
565 * @memberof PRDC_JSLAB_MATHJS_DOC
566 */
567
568 /**
569 * Calculate the inverse hyperbolic tangent of a value.
570 * @name atanh
571 * @kind function
572 * @param { number | Complex | Array | Matrix } x - Function input.
573 * @returns { number | Complex | Array | Matrix } The inverse hyperbolic
574 * tangent of 'x'.
575 * @memberof PRDC_JSLAB_MATHJS_DOC
576 */
577
578 /**
579 * Atomic mass constant , expressed in kg.
580 * @name atomicMass
581 * @kind member
582 * @memberof PRDC_JSLAB_MATHJS_DOC
583 * @type { number }
584 */
585
586 /**
587 * Avogadro 's number , approximately '6.022e23 ' mol-1.
588 * @name avogadro
589 * @kind member
590 * @memberof PRDC_JSLAB_MATHJS_DOC
591 * @type { number }
592 */
593
594 /**
595 * Compute the Bell Numbers , 'B(n)' .
596 * @name bellNumbers
597 * @kind function
598 * @param { number } n - The input value.
599 * @returns { number } Returns the nth Bell number.
600 * @memberof PRDC_JSLAB_MATHJS_DOC
601 */
602
603 /**
604 * BigNumber constructor .
605 * @name BigNumber
606 * @kind function

```

```

606   * @param { number | string | BigNumber } value - The numeric value.
607   * @returns { BigNumber } Returns a new BigNumber instance.
608   * @memberof PRDC_JSLAB_MATHJS_DOC
609   */
610
611 /**
612  * Create a BigNumber with arbitrary precision.
613  * @name bignumber
614  * @kind function
615  * @param { number | string | BigNumber } value - The numeric value.
616  * @returns { BigNumber } Returns a BigNumber instance.
617  * @memberof PRDC_JSLAB_MATHJS_DOC
618  */
619
620 /**
621  * Format a number as binary.
622  * @name bin
623  * @kind function
624  * @param { number | BigNumber } n - The number to format.
625  * @returns { string } The binary representation of 'n'.
626  * @memberof PRDC_JSLAB_MATHJS_DOC
627  */
628
629 /**
630  * Bitwise AND operation.
631  * @name bitAnd
632  * @kind function
633  * @param { number | BigNumber | Array | Matrix } x - First value.
634  * @param { number | BigNumber | Array | Matrix } y - Second value.
635  * @returns { number | BigNumber | Array | Matrix } Result of 'x AND y'.
636  * @memberof PRDC_JSLAB_MATHJS_DOC
637  */
638
639 /**
640  * Bitwise NOT operation.
641  * @name bitNot
642  * @kind function
643  * @param { number | BigNumber | Array | Matrix } x - Input value.
644  * @returns { number | BigNumber | Array | Matrix } Result of 'NOT x'.
645  * @memberof PRDC_JSLAB_MATHJS_DOC
646  */
647
648 /**
649  * Bitwise OR operation.
650  * @name bitOr
651  * @kind function
652  * @param { number | BigNumber | Array | Matrix } x - First value.
653  * @param { number | BigNumber | Array | Matrix } y - Second value.
654  * @returns { number | BigNumber | Array | Matrix } Result of 'x OR y'.
655  * @memberof PRDC_JSLAB_MATHJS_DOC
656  */
657
658 /**
659  * Bitwise XOR operation.
660  * @name bitXor

```

```
661 * @kind function
662 * @param { number | BigNumber | Array | Matrix } x - First value.
663 * @param { number | BigNumber | Array | Matrix } y - Second value.
664 * @returns { number | BigNumber | Array | Matrix } Result of 'x XOR y'.
665 * @memberof PRDC_JSLAB_MATHJS_DOC
666 */
667
668 /**
669 * A node representing a block of statements.
670 * @name BlockNode
671 * @kind function
672 * @param { Object[] } blocks - An array of statements.
673 * @returns { BlockNode } Returns a new BlockNode.
674 * @memberof PRDC_JSLAB_MATHJS_DOC
675 */
676
677 /**
678 * The Bohr magneton in units of 'J/T'.
679 * @name bohrMagneton
680 * @kind member
681 * @memberof PRDC_JSLAB_MATHJS_DOC
682 * @type { number }
683 */
684
685 /**
686 * The Bohr radius in meters.
687 * @name bohrRadius
688 * @kind member
689 * @memberof PRDC_JSLAB_MATHJS_DOC
690 * @type { number }
691 */
692
693 /**
694 * The Boltzmann constant in 'J/K'.
695 * @name boltzmann
696 * @kind member
697 * @memberof PRDC_JSLAB_MATHJS_DOC
698 * @type { number }
699 */
700
701 /**
702 * Parse a value into a boolean.
703 * @name boolean
704 * @kind function
705 * @param { * } x - The value to parse.
706 * @returns { boolean } The parsed boolean value.
707 * @memberof PRDC_JSLAB_MATHJS_DOC
708 */
709
710 /**
711 * The Catalan's constant.
712 * @name catalan
713 * @kind member
714 * @memberof PRDC_JSLAB_MATHJS_DOC
715 * @type { number }
```

```

716     */
717
718 /**
719 * Calculate the cube root of a value.
720 * @name cbrt
721 * @kind function
722 * @param { number | BigNumber | Complex | Array | Matrix } x - Function
723 *           input.
724 * @returns { number | BigNumber | Complex | Array | Matrix } The cube
725 *           root of 'x'.
726 * @memberof PRDC_JSLAB_MATHJS_DOC
727 */
728
729 /**
730 * Round a value towards plus infinity.
731 * @name ceil
732 * @kind function
733 * @param { number | BigNumber | Array | Matrix } x - Input value.
734 * @returns { number | BigNumber | Array | Matrix } The rounded value.
735 * @memberof PRDC_JSLAB_MATHJS_DOC
736 */
737 /**
738 * Create a chained operation , allowing to chain methods.
739 * @name chain
740 * @kind function
741 * @param { * } value - The initial value of the chain.
742 * @returns { Chain } A chain object.
743 * @memberof PRDC_JSLAB_MATHJS_DOC
744 */
745 /**
746 * Chain constructor .
747 * @name Chain
748 * @kind function
749 * @param { * } value - The initial value.
750 * @returns { Chain } Returns a new Chain instance .
751 * @memberof PRDC_JSLAB_MATHJS_DOC
752 */
753
754 /**
755 * Classical electron radius in meters .
756 * @name classicalElectronRadius
757 * @kind member
758 * @memberof PRDC_JSLAB_MATHJS_DOC
759 * @type { number }
760 */
761
762 /**
763 * Calculate the number of combinations of n items taken k at a time .
764 * @name combinations
765 * @kind function
766 * @param { number | BigNumber } n - Total number of items .
767 * @param { number | BigNumber } k - Number of items to choose .
768 * @returns { number | BigNumber } Number of possible combinations .

```

```

769     * @memberof PRDC_JSLAB_MATHJS_DOC
770     */
771
772 /**
773 * Calculate the number of combinations with replacement of n items taken
774 * k at a time.
775 * @name combinationsWithRep
776 * @kind function
777 * @param { number | BigNumber } n - Total number of items.
778 * @param { number | BigNumber } k - Number of items to choose.
779 * @returns { number | BigNumber } Number of possible combinations with
780 * replacement.
781 * @memberof PRDC_JSLAB_MATHJS_DOC
782 */
783
784 /**
785 * Compare two values numerically.
786 * @name compare
787 * @kind function
788 * @param { number | BigNumber | Fraction | Complex | Unit | string |
789 *         Array | Matrix } x - First value to compare.
790 * @param { number | BigNumber | Fraction | Complex | Unit | string |
791 *         Array | Matrix } y - Second value to compare.
792 * @returns { number | BigNumber | Fraction | Complex | Unit | string |
793 *           Array | Matrix } Returns 1 when x > y, -1 when x < y, and 0 when x ==
794 *             y.
795 * @memberof PRDC_JSLAB_MATHJS_DOC
796 */
797
798 /**
799 * Compare two strings using natural order.
800 * @name compareNatural
801 * @kind function
802 * @param { string } x - First string to compare.
803 * @param { string } y - Second string to compare.
804 * @returns { number } Returns 1 when x > y, -1 when x < y, and 0 when x ==
805 *             y.
806 * @memberof PRDC_JSLAB_MATHJS_DOC
807 */
808
809 /**
810 * Compare two strings lexicographically.
811 * @name compareText
812 * @kind function
813 * @param { string } x - First string to compare.
814 * @param { string } y - Second string to compare.
815 * @returns { number } Returns 1 when x > y, -1 when x < y, and 0 when x ==
816 *             y.
817 * @memberof PRDC_JSLAB_MATHJS_DOC
818 */
819
820 /**
821 * Compile an expression into a compiled function for faster evaluation.
822 * @name compile
823 * @kind function

```

```
816     * @param { string | Object } expr - The expression to compile.
817     * @returns { Object } A compiled expression that can be evaluated with `eval`.
818     * @memberof PRDC_JSLAB_MATHJS_DOC
819     */
820
821 /**
822 * Create a complex number.
823 * @name complex
824 * @kind function
825 * @param { number | string | Complex } [re] - Real part or a string
826   representation.
827 * @param { number } [im] - Imaginary part.
828 * @returns { Complex } Returns a Complex number.
829 * @memberof PRDC_JSLAB_MATHJS_DOC
830 */
831 /**
832 * Complex number constructor.
833 * @name Complex
834 * @kind function
835 * @param { number | string | Complex } [re] - Real part or a string
836   representation.
837 * @param { number } [im] - Imaginary part.
838 * @returns { Complex } A new Complex number.
839 * @memberof PRDC_JSLAB_MATHJS_DOC
840 */
841 /**
842 * Calculate the composition count of n into k parts.
843 * @name composition
844 * @kind function
845 * @param { number | BigNumber } n - Total number of items.
846 * @param { number | BigNumber } k - Number of parts.
847 * @returns { number | BigNumber } Number of compositions.
848 * @memberof PRDC_JSLAB_MATHJS_DOC
849 */
850
851 /**
852 * Concatenate matrices or arrays along a specified dimension.
853 * @name concat
854 * @kind function
855 * @param { Array | Matrix } a - First array or matrix.
856 * @param { ...Array | ...Matrix } b - Other arrays or matrices.
857 * @param { number | BigNumber } [dim=0] - Dimension along which to
858   concatenate.
859 * @returns { Array | Matrix } Concatenated array or matrix.
860 * @memberof PRDC_JSLAB_MATHJS_DOC
861 */
862 /**
863 * A node representing a conditional expression.
864 * @name ConditionalNode
865 * @kind function
866 * @param { Node } condition - The condition expression.
```

```
867     * @param { Node } trueExpr - Expression to evaluate when condition is
868     * true.
869     * @param { Node } falseExpr - Expression to evaluate when condition is
870     * false.
871     * @returns { ConditionalNode } A new ConditionalNode instance.
872     * @memberof PRDC_JSLAB_MATHJS_DOC
873     */
874
875 /**
876 * Conductance quantum in Siemens.
877 * @name conductanceQuantum
878 * @kind member
879 * @memberof PRDC_JSLAB_MATHJS_DOC
880 * @type { number }
881 */
882 /**
883 * Compute the complex conjugate of a complex number.
884 * @name conj
885 * @kind function
886 * @param { number | BigNumber | Complex | Array | Matrix } x - Input
887 * value.
888 * @returns { number | BigNumber | Complex | Array | Matrix } The complex
889 * conjugate of x.
890 * @memberof PRDC_JSLAB_MATHJS_DOC
891 */
892 /**
893 * A node representing a constant value.
894 * @name ConstantNode
895 * @kind function
896 * @param { number | string | BigNumber | Fraction } value - The constant
897 * value.
898 * @returns { ConstantNode } A new ConstantNode instance.
899 * @memberof PRDC_JSLAB_MATHJS_DOC
900 */
901 /**
902 * Calculate the cosine of a value.
903 * @name cos
904 * @kind function
905 * @param { number | Complex | Unit | Array | Matrix } x - Function input.
906 * @returns { number | Complex | Array | Matrix } The cosine of x.
907 * @memberof PRDC_JSLAB_MATHJS_DOC
908 */
909 /**
910 * Calculate the hyperbolic cosine of a value.
911 * @name cosh
912 * @kind function
913 * @param { number | Complex | Array | Matrix } x - Function input.
914 * @returns { number | Complex | Array | Matrix } The hyperbolic cosine of
915 * x.
```

```
916
917  /**
918   * Calculate the cotangent of a value.
919   * @name cot
920   * @kind function
921   * @param { number | Complex | Unit | Array | Matrix } x - Function input.
922   * @returns { number | Complex | Array | Matrix } The cotangent of x.
923   * @memberof PRDC_JSLAB_MATHJS_DOC
924   */
925
926 /**
927  * Calculate the hyperbolic cotangent of a value.
928  * @name coth
929  * @kind function
930  * @param { number | Complex | Array | Matrix } x - Function input.
931  * @returns { number | Complex | Array | Matrix } The hyperbolic cotangent
932  * of x.
933  * @memberof PRDC_JSLAB_MATHJS_DOC
934  */
935 /**
936  * Coulomb's constant in Nm^2/C^2.
937  * @name coulomb
938  * @kind member
939  * @memberof PRDC_JSLAB_MATHJS_DOC
940  * @type { number }
941  */
942
943 /**
944  * Count the number of elements in a matrix or array.
945  * @name count
946  * @kind function
947  * @param { Array | Matrix } x - The input array or matrix.
948  * @returns { number } The number of elements.
949  * @memberof PRDC_JSLAB_MATHJS_DOC
950  */
951
952 /**
953  * Create a user-defined unit and register it with the Unit system.
954  * @name createUnit
955  * @kind function
956  * @param { string } name - The name of the new unit.
957  * @param { string | Object } definition - Definition of the unit.
958  * @param { Object } [options] - Configuration options.
959  * @returns { Unit } The created unit.
960  * @memberof PRDC_JSLAB_MATHJS_DOC
961  */
962
963 /**
964  * Calculate the cosecant of a value.
965  * @name csc
966  * @kind function
967  * @param { number | Complex | Unit | Array | Matrix } x - Function input.
968  * @returns { number | Complex | Array | Matrix } The cosecant of x.
969  * @memberof PRDC_JSLAB_MATHJS_DOC
```

```
970 */  
971  
972 /**  
 * Calculate the hyperbolic cosecant of a value.  
 * @name csch  
 * @kind function  
 * @param { number | Complex | Array | Matrix } x - Function input.  
 * @returns { number | Complex | Array | Matrix } The hyperbolic cosecant  
 *          of x.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
980  
981 /**  
 * Compute the conjugate transpose of a matrix.  
 * @name ctranspose  
 * @kind function  
 * @param { Array | Matrix } x - The matrix to transpose.  
 * @returns { Array | Matrix } The conjugate transpose of x.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
988  
989 /**  
 * Compute the cube of a value.  
 * @name cube  
 * @kind function  
 * @param { number | BigNumber | Complex | Unit | Array | Matrix } x -  
 *           Input value.  
 * @returns { number | BigNumber | Complex | Unit | Array | Matrix } The  
 *          cube of x.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
996  
997  
998 /**  
 * Compute the cumulative sum of a matrix or array.  
 * @name cumsum  
 * @kind function  
 * @param { Array | Matrix } x - The input array or matrix.  
 * @param { number | BigNumber } [dim] - Dimension along which to  
 *           calculate.  
 * @returns { Array | Matrix } The cumulative sum.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
1007  
1008  
1009 /**  
 * Test element-wise whether two values are equal.  
 * @name deepEqual  
 * @kind function  
 * @param { * } x - First value to compare.  
 * @param { * } y - Second value to compare.  
 * @returns { boolean } Returns true if x and y are deep equal.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
1017  
1018  
1019 /**  
 * Dense matrix constructor.  
 * @param { number | BigNumber | Complex | Unit | Array | Matrix } x - The  
 *           input array or matrix.  
 * @param { number | BigNumber } [dim] - Dimension along which to  
 *           calculate.  
 * @returns { Matrix } The cumulative sum.  
 * @memberof PRDC_JSLAB_MATHJS_DOC  
 */  
1020
```

```

1021   * @name DenseMatrix
1022   * @kind function
1023   * @param { Array } data - The data for the matrix.
1024   * @returns { DenseMatrix } A new DenseMatrix instance.
1025   * @memberof PRDC_JSLAB_MATHJS_DOC
1026   */
1027
1028 /**
1029  * Take the derivative of an expression.
1030  * @name derivative
1031  * @kind function
1032  * @param { string | Node } expr - The expression to differentiate.
1033  * @param { string | Node } variable - The variable with respect to which
1034  * to differentiate.
1035  * @param { Object } [options] - Optional options object.
1036  * @returns { Node } The derivative of the expression.
1037  * @memberof PRDC_JSLAB_MATHJS_DOC
1038 */
1039 /**
1040  * Deuteron mass in kilograms.
1041  * @name deuteronMass
1042  * @kind member
1043  * @memberof PRDC_JSLAB_MATHJS_DOC
1044  * @type { number }
1045 */
1046
1047 /**
1048  * Calculate the differences between adjacent values in a matrix or array.
1049  * @name diff
1050  * @kind function
1051  * @param { Array | Matrix } x - Input array or matrix.
1052  * @param { number } [dim=0] - Dimension along which to calculate the
1053  * difference.
1054  * @returns { Array | Matrix } The differences.
1055  * @memberof PRDC_JSLAB_MATHJS_DOC
1056 */
1057 /**
1058  * Divide two scalar values, x / y.
1059  * @name divideScalar
1060  * @kind function
1061  * @param { number | BigNumber | Fraction | Complex } x - Numerator.
1062  * @param { number | BigNumber | Fraction | Complex } y - Denominator.
1063  * @returns { number | BigNumber | Fraction | Complex } The result of
1064  * division.
1065  * @memberof PRDC_JSLAB_MATHJS_DOC
1066 */
1067 /**
1068  * Divide two matrices element-wise.
1069  * @name dotDivide
1070  * @kind function
1071  * @param { Array | Matrix } x - Numerator matrix.
1072  * @param { Array | Matrix } y - Denominator matrix.

```

```
1073 * @returns { Array | Matrix } The element-wise division.  
1074 * @memberof PRDC_JSLAB_MATHJS_DOC  
1075 */  
1076  
1077 /**  
1078 * Multiply two matrices element-wise.  
1079 * @name dotMultiply  
1080 * @kind function  
1081 * @param { Array | Matrix } x - First matrix.  
1082 * @param { Array | Matrix } y - Second matrix.  
1083 * @returns { Array | Matrix } The element-wise multiplication.  
1084 * @memberof PRDC_JSLAB_MATHJS_DOC  
1085 */  
1086  
1087 /**  
1088 * Exponentiate two matrices element-wise.  
1089 * @name dotPow  
1090 * @kind function  
1091 * @param { Array | Matrix } x - Base matrix.  
1092 * @param { Array | Matrix } y - Exponent matrix.  
1093 * @returns { Array | Matrix } The element-wise exponentiation.  
1094 * @memberof PRDC_JSLAB_MATHJS_DOC  
1095 */  
1096  
1097 /**  
1098 * Euler's number, the base of natural logarithms.  
1099 * @name e  
1100 * @kind member  
1101 * @memberof PRDC_JSLAB_MATHJS_DOC  
1102 * @type { number }  
1103 */  
1104  
1105 /**  
1106 * Efimov factor.  
1107 * @name efimovFactor  
1108 * @kind member  
1109 * @memberof PRDC_JSLAB_MATHJS_DOC  
1110 * @type { number }  
1111 */  
1112  
1113 /**  
1114 * Calculate eigenvalues and eigenvectors of a matrix.  
1115 * @name eigs  
1116 * @kind function  
1117 * @param { Array | Matrix } x - A square matrix.  
1118 * @returns { Object } An object containing eigenvalues and eigenvectors.  
1119 * @memberof PRDC_JSLAB_MATHJS_DOC  
1120 */  
1121  
1122 /**  
1123 * Electric constant (vacuum permittivity) in F/m.  
1124 * @name electricConstant  
1125 * @kind member  
1126 * @memberof PRDC_JSLAB_MATHJS_DOC  
1127 * @type { number }
```

```
1128 */  
1129  
1130 /**  
1131 * Electron mass in kilograms.  
1132 * @name electronMass  
1133 * @kind member  
1134 * @memberof PRDC_JSLAB_MATHJS_DOC  
1135 * @type { number }  
1136 */  
1137  
1138 /**  
1139 * Elementary charge in coulombs.  
1140 * @name elementaryCharge  
1141 * @kind member  
1142 * @memberof PRDC_JSLAB_MATHJS_DOC  
1143 * @type { number }  
1144 */  
1145  
1146 /**  
1147 * Test whether two values are equal.  
1148 * @name equal  
1149 * @kind function  
1150 * @param { * } x - First value to compare.  
1151 * @param { * } y - Second value to compare.  
1152 * @returns { boolean } Returns true if x equals y.  
1153 * @memberof PRDC_JSLAB_MATHJS_DOC  
1154 */  
1155  
1156 /**  
1157 * Test whether two scalar values are equal.  
1158 * @name equalScalar  
1159 * @kind function  
1160 * @param { number | BigNumber | Fraction | Complex } x - First value.  
1161 * @param { number | BigNumber | Fraction | Complex } y - Second value.  
1162 * @returns { boolean } Returns true if x equals y.  
1163 * @memberof PRDC_JSLAB_MATHJS_DOC  
1164 */  
1165  
1166 /**  
1167 * Test whether two strings are equal.  
1168 * @name equalText  
1169 * @kind function  
1170 * @param { string } x - First string.  
1171 * @param { string } y - Second string.  
1172 * @returns { boolean } Returns true if x equals y.  
1173 * @memberof PRDC_JSLAB_MATHJS_DOC  
1174 */  
1175  
1176 /**  
1177 * Calculate the error function of a value.  
1178 * @name erf  
1179 * @kind function  
1180 * @param { number | BigNumber | Complex | Array | Matrix } x - Input  
value.  
1181 * @returns { number | BigNumber | Complex | Array | Matrix } The error
```

```

    function evaluated at x.
1182  * @memberof PRDC_JSLAB_MATHJS_DOC
1183  */
1184
1185 /**
1186  * Calculate the exponential of a value.
1187  * @name exp
1188  * @kind function
1189  * @param { number | BigNumber | Complex | Array | Matrix } x - Exponent.
1190  * @returns { number | BigNumber | Complex | Array | Matrix } The
1191  *          exponential of x.
1192  * @memberof PRDC_JSLAB_MATHJS_DOC
1193  */
1194
1195 /**
1196  * Calculate  $\exp(x) - 1$ .
1197  * @name expm1
1198  * @kind function
1199  * @param { number | BigNumber | Complex } x - Input value.
1200  * @returns { number | BigNumber | Complex } The result of  $\exp(x) - 1$ .
1201  * @memberof PRDC_JSLAB_MATHJS_DOC
1202  */
1203
1204 /**
1205  * Calculate the factorial of a value.
1206  * @name factorial
1207  * @kind function
1208  * @param { number | BigNumber | Array | Matrix } n - A non-negative
1209  *          integer.
1210  * @returns { number | BigNumber | Array | Matrix } The factorial of n.
1211  * @memberof PRDC_JSLAB_MATHJS_DOC
1212  */
1213
1214 /**
1215  * Boolean value false.
1216  * @name false
1217  * @kind member
1218  * @memberof PRDC_JSLAB_MATHJS_DOC
1219  * @type { boolean }
1220  */
1221
1222 /**
1223  * Faraday constant in C/mol.
1224  * @name faraday
1225  * @kind member
1226  * @memberof PRDC_JSLAB_MATHJS_DOC
1227  * @type { number }
1228  */
1229
1230 /**
1231  * Fermi coupling constant in  $\text{GeV}^{-2}$ .
1232  * @name fermiCoupling
1233  * @kind member
1234  * @memberof PRDC_JSLAB_MATHJS_DOC
1235  * @type { number }

```

```
1234 */  
1235  
1236 /**  
1237 * Compute the Fast Fourier Transform of a matrix or array.  
1238 * @name fft  
1239 * @kind function  
1240 * @param { Array | Matrix } x - Input array or matrix.  
1241 * @returns { Array | Matrix } The FFT of x.  
1242 * @memberof PRDC_JSLAB_MATHJS_DOC  
1243 */  
1244  
1245 /**  
1246 * Fibonacci heap data structure.  
1247 * @name FibonacciHeap  
1248 * @kind function  
1249 * @param { function } [compare] - Comparison function.  
1250 * @returns { FibonacciHeap } A new FibonacciHeap instance.  
1251 * @memberof PRDC_JSLAB_MATHJS_DOC  
1252 */  
1253  
1254 /**  
1255 * Filter the items in an array or matrix.  
1256 * @name filter  
1257 * @kind function  
1258 * @param { Array | Matrix } x - The input array or matrix.  
1259 * @param { function } test - The test function.  
1260 * @returns { Array | Matrix } The filtered array or matrix.  
1261 * @memberof PRDC_JSLAB_MATHJS_DOC  
1262 */  
1263  
1264 /**  
1265 * Fine-structure constant.  
1266 * @name fineStructure  
1267 * @kind member  
1268 * @memberof PRDC_JSLAB_MATHJS_DOC  
1269 * @type { number }  
1270 */  
1271  
1272 /**  
1273 * First radiation constant in Wm^2.  
1274 * @name firstRadiation  
1275 * @kind member  
1276 * @memberof PRDC_JSLAB_MATHJS_DOC  
1277 * @type { number }  
1278 */  
1279  
1280 /**  
1281 * Round a value towards zero.  
1282 * @name fix  
1283 * @kind function  
1284 * @param { number | BigNumber | Array | Matrix } x - Input value.  
1285 * @returns { number | BigNumber | Array | Matrix } The rounded value.  
1286 * @memberof PRDC_JSLAB_MATHJS_DOC  
1287 */  
1288
```



```
1289 /**
1290 * Flatten a multi-dimensional array or matrix into a single dimension.
1291 * @name flatten
1292 * @kind function
1293 * @param { Array | Matrix } x - The input array or matrix.
1294 * @returns { Array | Matrix } The flattened array or matrix.
1295 * @memberof PRDC_JSLAB_MATHJS_DOC
1296 */
1297
1298 /**
1299 * Round a value towards negative infinity.
1300 * @name floor
1301 * @kind function
1302 * @param { number | BigNumber | Array | Matrix } x - Input value.
1303 * @returns { number | BigNumber | Array | Matrix } The rounded value.
1304 * @memberof PRDC_JSLAB_MATHJS_DOC
1305 */
1306
1307 /**
1308 * Iterate over each element of a matrix or array.
1309 * @name forEach
1310 * @kind function
1311 * @param { Array | Matrix } x - The input array or matrix.
1312 * @param { function } callback - The function to execute on each element.
1313 * @returns { void }
1314 * @memberof PRDC_JSLAB_MATHJS_DOC
1315 */
1316
1317 /**
1318 * Format a value for display.
1319 * @name format
1320 * @kind function
1321 * @param { * } value - The value to format.
1322 * @param { Object | function } [options] - Formatting options or custom
1323 * function.
1324 * @returns { string } The formatted value.
1325 * @memberof PRDC_JSLAB_MATHJS_DOC
1326 */
1327
1328 /**
1329 * Create a fraction.
1330 * @name fraction
1331 * @kind function
1332 * @param { number | string | Fraction } numerator - Numerator.
1333 * @param { number | string } [denominator] - Denominator.
1334 * @returns { Fraction } A new Fraction instance.
1335 * @memberof PRDC_JSLAB_MATHJS_DOC
1336 */
1337
1338 /**
1339 * Fraction constructor.
1340 * @name Fraction
1341 * @kind function
1342 * @param { number | string | Fraction } numerator - Numerator.
1343 * @param { number | string } [denominator] - Denominator.
```

```

1343   * @returns { Fraction } A new Fraction instance.
1344   * @memberof PRDC_JSLAB_MATHJS_DOC
1345   */
1346
1347 /**
1348 * A node representing a function assignment in the expression tree.
1349 * @name FunctionAssignmentNode
1350 * @kind function
1351 * @param { string } name - The name of the function being assigned.
1352 * @param { string[] } params - An array of parameter names.
1353 * @param { Node } expr - The function expression.
1354 * @returns { FunctionAssignmentNode } A new FunctionAssignmentNode
1355     instance.
1356 * @memberof PRDC_JSLAB_MATHJS_DOC
1357 */
1358
1359 /**
1360 * A node representing a function call in the expression tree.
1361 * @name FunctionNode
1362 * @kind function
1363 * @param { string | SymbolNode } name - The name of the function or a
1364     SymbolNode.
1365 * @param { Node[] } args - An array of argument nodes.
1366 * @returns { FunctionNode } A new FunctionNode instance.
1367 * @memberof PRDC_JSLAB_MATHJS_DOC
1368 */
1369
1370 /**
1371 * Calculate the gamma function of a value.
1372 * @name gamma
1373 * @kind function
1374 * @param { number | BigNumber | Complex | Array | Matrix } n - The input
1375     value.
1376 * @returns { number | BigNumber | Complex | Array | Matrix } The gamma of
1377     n.
1378 * @memberof PRDC_JSLAB_MATHJS_DOC
1379 */
1380
1381 /**
1382 * The molar gas constant, in units of J/(molK).
1383 * @name gasConstant
1384 * @kind member
1385 * @memberof PRDC_JSLAB_MATHJS_DOC
1386 * @type { number }
1387 */
1388
1389 /**
1390 * Compute the greatest common divisor of two or more values.
1391 * @name gcd
1392 * @kind function
1393 * @param { ...number | ...BigNumber | Array | Matrix } args - Two or more
1394     integer numbers.
1395 * @returns { number | BigNumber | Array | Matrix } The greatest common
1396     divisor.
1397 * @memberof PRDC_JSLAB_MATHJS_DOC

```

```
1392      */
1393
1394  /**
1395   * Get the data type of a matrix.
1396   * @name getMatrixDataType
1397   * @kind function
1398   * @param { Matrix } matrix - The input matrix.
1399   * @returns { string } The data type of the matrix elements.
1400   * @memberof PRDC_JSLAB_MATHJS_DOC
1401   */
1402
1403 /**
1404  * Newtonian constant of gravitation , in m^3/(kg s^2) .
1405  * @name gravitationConstant
1406  * @kind member
1407  * @memberof PRDC_JSLAB_MATHJS_DOC
1408  * @type { number }
1409  */
1410
1411 /**
1412  * Acceleration due to gravity on Earth , in m/s ^2.
1413  * @name gravity
1414  * @kind member
1415  * @memberof PRDC_JSLAB_MATHJS_DOC
1416  * @type { number }
1417  */
1418
1419 /**
1420  * Hartree energy , in joules .
1421  * @name hartreeEnergy
1422  * @kind member
1423  * @memberof PRDC_JSLAB_MATHJS_DOC
1424  * @type { number }
1425  */
1426
1427 /**
1428  * Test whether a value is a numeric value .
1429  * @name hasNumericValue
1430  * @kind function
1431  * @param { * } x - The value to test .
1432  * @returns { boolean } Returns true if x is numeric .
1433  * @memberof PRDC_JSLAB_MATHJS_DOC
1434  */
1435
1436 /**
1437  * Help object constructor .
1438  * @name Help
1439  * @kind function
1440  * @param { * } value - The function or object to get help for .
1441  * @returns { Help } A new Help instance .
1442  * @memberof PRDC_JSLAB_MATHJS_DOC
1443  */
1444
1445 /**
1446  * Format a number as hexadecimal .
```

```
1447 * @name hex
1448 * @kind function
1449 * @param { number | BigNumber } value - The value to format.
1450 * @returns { string } The hexadecimal representation.
1451 * @memberof PRDC_JSLAB_MATHJS_DOC
1452 */
1453
1454 /**
1455 * Calculate the hypotenuse of a list of values.
1456 * @name hypot
1457 * @kind function
1458 * @param { ...number | ...BigNumber | Array | Matrix } args - The input
1459 * values.
1460 * @returns { number | BigNumber | Array | Matrix } The hypotenuse.
1461 * @memberof PRDC_JSLAB_MATHJS_DOC
1462 */
1463 /**
1464 * The imaginary unit 'i'.
1465 * @name i
1466 * @kind member
1467 * @memberof PRDC_JSLAB_MATHJS_DOC
1468 * @type { Complex }
1469 */
1470
1471 /**
1472 * Create an identity matrix.
1473 * @name identity
1474 * @kind function
1475 * @param { number | Array } size - The size of the matrix.
1476 * @param { string } [format] - The matrix format.
1477 * @returns { Matrix } An identity matrix.
1478 * @memberof PRDC_JSLAB_MATHJS_DOC
1479 */
1480
1481 /**
1482 * Compute the inverse Fast Fourier Transform.
1483 * @name ifft
1484 * @kind function
1485 * @param { Array | Matrix } x - Input array or matrix.
1486 * @returns { Array | Matrix } The inverse FFT of x.
1487 * @memberof PRDC_JSLAB_MATHJS_DOC
1488 */
1489
1490 /**
1491 * Get the imaginary part of a complex number.
1492 * @name im
1493 * @kind function
1494 * @param { number | BigNumber | Complex | Array | Matrix } x - Input
1495 * value.
1496 * @returns { number | BigNumber | Array | Matrix } The imaginary part of
1497 * x.
1498 * @memberof PRDC_JSLAB_MATHJS_DOC
1499 */
1500
```

```
1499  /**
1500   * Immutable dense matrix constructor.
1501   * @name ImmutableDenseMatrix
1502   * @kind function
1503   * @param { Array } data - The data for the matrix.
1504   * @returns { ImmutableDenseMatrix } A new ImmutableDenseMatrix instance.
1505   * @memberof PRDC_JSLAB_MATHJS_DOC
1506   */
1507
1508 /**
1509  * Index constructor for matrices.
1510  * @name Index
1511  * @kind function
1512  * @param { ...Range | ...number } ranges - Ranges or indices.
1513  * @returns { Index } A new Index instance.
1514  * @memberof PRDC_JSLAB_MATHJS_DOC
1515  */
1516
1517 /**
1518  * A node representing an index operation in the expression tree.
1519  * @name IndexNode
1520  * @kind function
1521  * @param { Node[] } dimensions - The indices.
1522  * @returns { IndexNode } A new IndexNode instance.
1523  * @memberof PRDC_JSLAB_MATHJS_DOC
1524  */
1525
1526 /**
1527  * Compute the intersection of two sets.
1528  * @name intersect
1529  * @kind function
1530  * @param { Array | Matrix } a - First set.
1531  * @param { Array | Matrix } b - Second set.
1532  * @returns { Array | Matrix } The intersection of a and b.
1533  * @memberof PRDC_JSLAB_MATHJS_DOC
1534  */
1535
1536 /**
1537  * Inverse conductance quantum, in ohms.
1538  * @name inverseConductanceQuantum
1539  * @kind member
1540  * @memberof PRDC_JSLAB_MATHJS_DOC
1541  * @type { number }
1542  */
1543
1544 /**
1545  * Calculate the modular inverse of a value.
1546  * @name invmod
1547  * @kind function
1548  * @param { number | BigNumber } a - The value.
1549  * @param { number | BigNumber } m - The modulus.
1550  * @returns { number | BigNumber } The modular inverse.
1551  * @memberof PRDC_JSLAB_MATHJS_DOC
1552  */
1553
```

```
1554 /**
1555 * Test whether a value is an integer.
1556 * @name isInteger
1557 * @kind function
1558 * @param { number | BigNumber } x - The value to test.
1559 * @returns { boolean } Returns true if x is an integer.
1560 * @memberof PRDC_JSLAB_MATHJS_DOC
1561 */
1562
1563 /**
1564 * Test whether a value is negative.
1565 * @name isNegative
1566 * @kind function
1567 * @param { number | BigNumber } x - The value to test.
1568 * @returns { boolean } Returns true if x is negative.
1569 * @memberof PRDC_JSLAB_MATHJS_DOC
1570 */
1571
1572 /**
1573 * Test whether a value is positive.
1574 * @name isPositive
1575 * @kind function
1576 * @param { number | BigNumber } x - The value to test.
1577 * @returns { boolean } Returns true if x is positive.
1578 * @memberof PRDC_JSLAB_MATHJS_DOC
1579 */
1580
1581 /**
1582 * Test whether a number is prime.
1583 * @name isPrime
1584 * @kind function
1585 * @param { number | BigNumber } x - The value to test.
1586 * @returns { boolean } Returns true if x is prime.
1587 * @memberof PRDC_JSLAB_MATHJS_DOC
1588 */
1589
1590 /**
1591 * Test whether a value is zero.
1592 * @name isZero
1593 * @kind function
1594 * @param { number | BigNumber } x - The value to test.
1595 * @returns { boolean } Returns true if x is zero.
1596 * @memberof PRDC_JSLAB_MATHJS_DOC
1597 */
1598
1599 /**
1600 * Calculate the Kullback-Leibler divergence between two distributions.
1601 * @name kldivergence
1602 * @kind function
1603 * @param { Array | Matrix } p - First probability distribution.
1604 * @param { Array | Matrix } q - Second probability distribution.
1605 * @returns { number } The KL divergence  $D_{KL}(p || q)$ .
1606 * @memberof PRDC_JSLAB_MATHJS_DOC
1607 */
1608
```

```
1609  /**
1610   * Von Klitzing constant , in ohms.
1611   * @name klitzing
1612   * @kind member
1613   * @memberof PRDC_JSLAB_MATHJS_DOC
1614   * @type { number }
1615   */
1616
1617 /**
1618  * Compute the Kronecker product of two matrices.
1619  * @name kron
1620  * @kind function
1621  * @param { Array | Matrix } A - First matrix.
1622  * @param { Array | Matrix } B - Second matrix.
1623  * @returns { Matrix } The Kronecker product of A and B.
1624  * @memberof PRDC_JSLAB_MATHJS_DOC
1625  */
1626
1627 /**
1628  * Natural logarithm of 10.
1629  * @name LN10
1630  * @kind member
1631  * @memberof PRDC_JSLAB_MATHJS_DOC
1632  * @type { number }
1633  */
1634
1635 /**
1636  * Natural logarithm of 2.
1637  * @name LN2
1638  * @kind member
1639  * @memberof PRDC_JSLAB_MATHJS_DOC
1640  * @type { number }
1641  */
1642
1643 /**
1644  * Base 10 logarithm of e.
1645  * @name LOG10E
1646  * @kind member
1647  * @memberof PRDC_JSLAB_MATHJS_DOC
1648  * @type { number }
1649  */
1650
1651 /**
1652  * Base 2 logarithm of e.
1653  * @name LOG2E
1654  * @kind member
1655  * @memberof PRDC_JSLAB_MATHJS_DOC
1656  * @type { number }
1657  */
1658
1659 /**
1660  * Test whether value x is larger than y.
1661  * @name larger
1662  * @kind function
1663  * @param { number | BigNumber | Fraction | Complex | Unit | string | }
```

```

1664     Array | Matrix } x - First value.
1665     * @param { number | BigNumber | Fraction | Complex | Unit | string |
1666       Array | Matrix } y - Second value.
1667     * @returns { boolean | Array | Matrix } Returns true if x > y.
1668     * @memberof PRDC_JSLAB_MATHJS_DOC
1669   */
1670
1671 /**
1672  * Test whether value x is larger than or equal to y.
1673  * @name largerEq
1674  * @kind function
1675  * @param { number | BigNumber | Fraction | Complex | Unit | string |
1676    Array | Matrix } x - First value.
1677  * @param { number | BigNumber | Fraction | Complex | Unit | string |
1678    Array | Matrix } y - Second value.
1679  * @returns { boolean | Array | Matrix } Returns true if x >= y.
1680  * @memberof PRDC_JSLAB_MATHJS_DOC
1681 */
1682
1683 /**
1684  * Compute the least common multiple of two or more values.
1685  * @name lcm
1686  * @kind function
1687  * @param { ...number | ...BigNumber | Array | Matrix } args - Two or more
1688    integer numbers.
1689  * @returns { number | BigNumber | Array | Matrix } The least common
1690    multiple.
1691  * @memberof PRDC_JSLAB_MATHJS_DOC
1692 */
1693
1694 /**
1695  * Count the number of leaf nodes in an expression tree.
1696  * @name leafCount
1697  * @kind function
1698  * @param { Node } node - The root node of the expression tree.
1699  * @returns { number } The number of leaf nodes.
1700  * @memberof PRDC_JSLAB_MATHJS_DOC
1701 */
1702
1703 /**
1704  * Bitwise left shift operation.
1705  * @name leftShift
1706  * @kind function
1707  * @param { number | BigNumber | Array | Matrix } x - Value to be shifted.
1708  * @param { number | BigNumber | Array | Matrix } y - Amount of bits to
1709    shift.
1710  * @returns { number | BigNumber | Array | Matrix } The shifted value.
1711  * @memberof PRDC_JSLAB_MATHJS_DOC
1712 */
1713
1714 /**
1715  * Compute the natural logarithm of the gamma function.
1716  * @name lgamma
1717  * @kind function
1718  * @param { number } n - The input value.
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769

```

```
1712 * @returns { number } The natural logarithm of the gamma function at n.
1713 * @memberof PRDC_JSLAB_MATHJS_DOC
1714 */
1715
1716 /**
1717 * Calculate the natural logarithm of a value.
1718 * @name log
1719 * @kind function
1720 * @param { number | BigNumber | Complex | Array | Matrix } x - Value for
1721 * which to calculate the logarithm.
1722 * @param { number | BigNumber | Complex } [base=e] - Base of the
1723 * logarithm.
1724 * @returns { number | BigNumber | Complex | Array | Matrix } The
1725 * logarithm of x.
1726 * @memberof PRDC_JSLAB_MATHJS_DOC
1727 */
1728
1729 /**
1730 * Calculate the base-10 logarithm of a value.
1731 * @name log10
1732 * @kind function
1733 * @param { number | BigNumber | Complex | Array | Matrix } x - Value for
1734 * which to calculate the logarithm.
1735 * @returns { number | BigNumber | Complex | Array | Matrix } The base-10
1736 * logarithm of x.
1737 * @memberof PRDC_JSLAB_MATHJS_DOC
1738 */
1739
1740 /**
1741 * Calculate the natural logarithm of 1 plus a value.
1742 * @name log1p
1743 * @kind function
1744 * @param { number | BigNumber } x - Input value.
1745 * @returns { number | BigNumber } The result of  $\log(1 + x)$ .
1746 * @memberof PRDC_JSLAB_MATHJS_DOC
1747 */
1748
1749 /**
1750 * Calculate the base-2 logarithm of a value.
1751 * @name log2
1752 * @kind function
1753 * @param { number | BigNumber | Complex | Array | Matrix } x - Value for
1754 * which to calculate the logarithm.
1755 * @returns { number | BigNumber | Complex | Array | Matrix } The base-2
1756 * logarithm of x.
1757 * @memberof PRDC_JSLAB_MATHJS_DOC
1758 */
1759
1760 /**
1761 * Loschmidt constant at 0°C and 1 atm, in  $m^{-3}$ .
1762 * @name loschmidt
1763 * @kind member
1764 * @memberof PRDC_JSLAB_MATHJS_DOC
1765 * @type { number }
1766 */
1767
```

```

1760
1761 /**
1762 * Solve a linear system A * x = b where A is a lower triangular matrix.
1763 * @name lsolve
1764 * @kind function
1765 * @param { Matrix | Array } L - A lower triangular matrix.
1766 * @param { Matrix | Array } b - A column vector.
1767 * @returns { Matrix | Array } The solution vector x.
1768 * @memberof PRDC_JSLAB_MATHJS_DOC
1769 */
1770
1771 /**
1772 * Find all solutions of a linear system A * x = b where A is a lower
1773 * triangular matrix.
1774 * @name lsolveAll
1775 * @kind function
1776 * @param { Matrix | Array } L - A lower triangular matrix.
1777 * @param { Matrix | Array } b - A column vector.
1778 * @returns { Array } An array of solutions.
1779 * @memberof PRDC_JSLAB_MATHJS_DOC
1780 */
1781 /**
1782 * Compute the LU decomposition with partial pivoting.
1783 * @name lup
1784 * @kind function
1785 * @param { Matrix | Array } A - A square matrix.
1786 * @param { number } [threshold=1e-10] - Tolerance threshold.
1787 * @returns { Object } An object containing L, U, and P matrices.
1788 * @memberof PRDC_JSLAB_MATHJS_DOC
1789 */
1790
1791 /**
1792 * Solve a linear system using LU decomposition.
1793 * @name lusolve
1794 * @kind function
1795 * @param { Matrix | Array } A - Coefficient matrix.
1796 * @param { Matrix | Array } b - Right-hand side vector or matrix.
1797 * @param { string } [order] - Matrix storage order.
1798 * @returns { Matrix | Array } The solution vector or matrix.
1799 * @memberof PRDC_JSLAB_MATHJS_DOC
1800 */
1801
1802 /**
1803 * Solve the Lyapunov equation A*X + X*A' = Q.
1804 * @name lyap
1805 * @kind function
1806 * @param { Matrix | Array } A - A square matrix.
1807 * @param { Matrix | Array } Q - A square matrix.
1808 * @returns { Matrix } Solution X of the Lyapunov equation.
1809 * @memberof PRDC_JSLAB_MATHJS_DOC
1810 */
1811
1812 /**
1813 * Compute the median absolute deviation of a set of values.

```

```

1814     * @name mad
1815     * @kind function
1816     * @param { Array | Matrix } array - Input array.
1817     * @param { number | BigNumber } [dim] - Dimension along which to compute.
1818     * @returns { number | BigNumber | Array | Matrix } The median absolute
1819         deviation.
1820     * @memberof PRDC_JSLAB_MATHJS_DOC
1821     */
1822
1823 /**
1824 * Magnetic constant (vacuum permeability), in N/A^2.
1825 * @name magneticConstant
1826 * @kind member
1827 * @memberof PRDC_JSLAB_MATHJS_DOC
1828 * @type { number }
1829 */
1830
1831 /**
1832 * Magnetic flux quantum, in Wb.
1833 * @name magneticFluxQuantum
1834 * @kind member
1835 * @memberof PRDC_JSLAB_MATHJS_DOC
1836 * @type { number }
1837 */
1838 /**
1839 * Map a function over the elements of a matrix or array.
1840 * @name map
1841 * @kind function
1842 * @param { Array | Matrix } x - The input array or matrix.
1843 * @param { function } callback - The function to apply.
1844 * @returns { Array | Matrix } The result after applying the callback.
1845 * @memberof PRDC_JSLAB_MATHJS_DOC
1846 */
1847
1848 /**
1849 * Create a matrix.
1850 * @name matrix
1851 * @kind function
1852 * @param { Array | Matrix } [data] - The data for the matrix.
1853 * @param { string } [format] - The matrix format.
1854 * @returns { Matrix } A new Matrix instance.
1855 * @memberof PRDC_JSLAB_MATHJS_DOC
1856 */
1857
1858 /**
1859 * Matrix constructor.
1860 * @name Matrix
1861 * @kind function
1862 * @param { Array | Matrix } [data] - The data for the matrix.
1863 * @param { string } [format] - The matrix format.
1864 * @returns { Matrix } A new Matrix instance.
1865 * @memberof PRDC_JSLAB_MATHJS_DOC
1866 */
1867

```

```
1868 /**
1869 * Create a matrix from given columns.
1870 * @name matrixFromColumns
1871 * @kind function
1872 * @param { ...Array | ...Matrix } columns - Columns to construct the
1873 * matrix.
1874 * @returns { Matrix } The constructed matrix.
1875 * @memberof PRDC_JSLAB_MATHJS_DOC
1876 */
1877 /**
1878 * Create a matrix using a provided function.
1879 * @name matrixFromFunction
1880 * @kind function
1881 * @param { Array } size - The size of the matrix.
1882 * @param { function } callback - Function to generate matrix entries.
1883 * @returns { Matrix } The generated matrix.
1884 * @memberof PRDC_JSLAB_MATHJS_DOC
1885 */
1886 /**
1887 * Create a matrix from given rows.
1888 * @name matrixFromRows
1889 * @kind function
1890 * @param { ...Array | ...Matrix } rows - Rows to construct the matrix.
1891 * @returns { Matrix } The constructed matrix.
1892 * @memberof PRDC_JSLAB_MATHJS_DOC
1893 */
1894 /**
1895 * Compute the arithmetic mean of a set of values.
1896 * @name mean
1897 * @kind function
1898 * @param { Array | Matrix } array - Input array.
1899 * @param { number | BigNumber } [dim] - Dimension along which to compute.
1900 * @returns { number | BigNumber | Array | Matrix } The mean value.
1901 * @memberof PRDC_JSLAB_MATHJS_DOC
1902 */
1903 /**
1904 * Compute the median of a set of values.
1905 * @name median
1906 * @kind function
1907 * @param { Array | Matrix } array - Input array.
1908 * @param { number | BigNumber } [dim] - Dimension along which to compute.
1909 * @returns { number | BigNumber | Array | Matrix } The median value.
1910 * @memberof PRDC_JSLAB_MATHJS_DOC
1911 */
1912 /**
1913 * Calculate the modulus of two numbers.
1914 * @name mod
1915 * @kind function
1916 * @param { number | BigNumber | Array | Matrix } x - Dividend.
1917 * @param { number | BigNumber | Array | Matrix } y - Divisor.
```

```
1922 * @returns { number | BigNumber | Array | Matrix } The remainder after
1923   division.
1924 * @memberof PRDC_JSLAB_MATHJS_DOC
1925 */
1926 /**
1927 * Compute the mode of a set of values.
1928 * @name mode
1929 * @kind function
1930 * @param { Array | Matrix } array - Input array.
1931 * @returns { Array | Matrix } The mode(s) of the array.
1932 * @memberof PRDC_JSLAB_MATHJS_DOC
1933 */
1934 /**
1935 * Molar mass constant , in kg/mol.
1936 * @name molarMass
1937 * @kind member
1938 * @memberof PRDC_JSLAB_MATHJS_DOC
1939 * @type { number }
1940 */
1941 /**
1942 * Molar mass of carbon-12, in kg/mol.
1943 * @name molarMassC12
1944 * @kind member
1945 * @memberof PRDC_JSLAB_MATHJS_DOC
1946 * @type { number }
1947 */
1948 /**
1949 * Molar Planck constant , in J s/mol.
1950 * @name molarPlanckConstant
1951 * @kind member
1952 * @memberof PRDC_JSLAB_MATHJS_DOC
1953 * @type { number }
1954 */
1955 /**
1956 * Molar volume of an ideal gas at 1 atm and 0°C, in m /mol.
1957 * @name molarVolume
1958 * @kind member
1959 * @memberof PRDC_JSLAB_MATHJS_DOC
1960 * @type { number }
1961 */
1962 /**
1963 * Compute the multinomial coefficient of a list of integers.
1964 * @name multinomial
1965 * @kind function
1966 * @param { ...number | Array } args - Integer numbers.
1967 * @returns { number } The multinomial coefficient .
1968 * @memberof PRDC_JSLAB_MATHJS_DOC
1969 */
1970 /**
1971 * Molar mass constant , in kg/mol.
1972 * @name molarMass
1973 * @kind member
1974 * @memberof PRDC_JSLAB_MATHJS_DOC
1975 */
```

```

1976 /**
1977 * Multiply two scalar values , x * y.
1978 * @name multiplyScalar
1979 * @kind function
1980 * @param { number | BigNumber | Fraction | Complex } x - First value.
1981 * @param { number | BigNumber | Fraction | Complex } y - Second value.
1982 * @returns { number | BigNumber | Fraction | Complex } The product of x
1983 *         and y.
1984 * @memberof PRDC_JSLAB_MATHJS_DOC
1985 */
1986 /**
1987 * Neutron mass , in kilograms .
1988 * @name neutronMass
1989 * @kind member
1990 * @memberof PRDC_JSLAB_MATHJS_DOC
1991 * @type { number }
1992 */
1993 /**
1994 * Logical NOT operation .
1995 * @name not
1996 * @kind function
1997 * @param { boolean | Array | Matrix } x - Input value .
1998 * @returns { boolean | Array | Matrix } The logical negation of x .
1999 * @memberof PRDC_JSLAB_MATHJS_DOC
2000 */
2001 /**
2002 /**
2003 * Calculate the nth root of a value .
2004 * @name nthRoot
2005 * @kind function
2006 * @param { number | BigNumber | Complex | Array | Matrix } a - The value .
2007 * @param { number | BigNumber } [root=2] - The root .
2008 * @returns { number | BigNumber | Complex | Array | Matrix } The nth root
2009 *         of a .
2010 * @memberof PRDC_JSLAB_MATHJS_DOC
2011 */
2012 /**
2013 * Calculate the nth roots of a complex number .
2014 * @name nthRoots
2015 * @kind function
2016 * @param { number | Complex } a - The value .
2017 * @param { number | BigNumber } n - The degree of the root .
2018 * @returns { Array } An array of the n roots .
2019 * @memberof PRDC_JSLAB_MATHJS_DOC
2020 */
2021 /**
2022 /**
2023 * Nuclear magneton , in J/T .
2024 * @name nuclearMagneton
2025 * @kind member
2026 * @memberof PRDC_JSLAB_MATHJS_DOC
2027 * @type { number }
2028 */

```



```
2029 */  
2030  
2031 /**  
2032 * JavaScript null value.  
2033 * @name null  
2034 * @kind member  
2035 * @memberof PRDC_JSLAB_MATHJS_DOC  
2036 * @type { null }  
2037 */  
2038  
2039 /**  
2040 * Parse a value into a number.  
2041 * @name number  
2042 * @kind function  
2043 * @param { * } value - The value to parse.  
2044 * @returns { number } The numeric value.  
2045 * @memberof PRDC_JSLAB_MATHJS_DOC  
2046 */  
2047  
2048 /**  
2049 * Convert a math.js data type to a numeric type.  
2050 * @name numeric  
2051 * @kind function  
2052 * @param { * } value - The value to convert.  
2053 * @returns { number | BigNumber | Complex } The numeric representation.  
2054 * @memberof PRDC_JSLAB_MATHJS_DOC  
2055 */  
2056  
2057 /**  
2058 * A node representing an object in the expression tree.  
2059 * @name ObjectNode  
2060 * @kind function  
2061 * @param { Object } properties - Object properties as nodes.  
2062 * @returns { ObjectNode } A new ObjectNode instance.  
2063 * @memberof PRDC_JSLAB_MATHJS_DOC  
2064 */  
2065  
2066 /**  
2067 * Format a number as octal.  
2068 * @name oct  
2069 * @kind function  
2070 * @param { number | BigNumber } value - The value to format.  
2071 * @returns { string } The octal representation.  
2072 * @memberof PRDC_JSLAB_MATHJS_DOC  
2073 */  
2074  
2075 /**  
2076 * A node representing an operator in the expression tree.  
2077 * @name OperatorNode  
2078 * @kind function  
2079 * @param { string } op - The operator symbol.  
2080 * @param { string } fn - The function name.  
2081 * @param { Node[] } args - An array of argument nodes.  
2082 * @param { boolean } [implicit=false] - Is the operator implicit?  
2083 * @returns { OperatorNode } A new OperatorNode instance.
```

```
2084 * @memberof PRDC_JSLAB_MATHJS_DOC
2085 */
2086
2087 /**
2088 * Logical OR operation.
2089 * @name or
2090 * @kind function
2091 * @param { boolean | Array | Matrix } x - First value.
2092 * @param { boolean | Array | Matrix } y - Second value.
2093 * @returns { boolean | Array | Matrix } The logical OR of x and y.
2094 * @memberof PRDC_JSLAB_MATHJS_DOC
2095 */
2096
2097 /**
2098 * A node representing parentheses in the expression tree.
2099 * @name ParenthesisNode
2100 * @kind function
2101 * @param { Node } content - The node encapsulated by the parentheses.
2102 * @returns { ParenthesisNode } A new ParenthesisNode instance.
2103 * @memberof PRDC_JSLAB_MATHJS_DOC
2104 */
2105
2106 /**
2107 * Parse and evaluate an expression.
2108 * @name parse
2109 * @kind function
2110 * @param { string | Object } expr - The expression to parse.
2111 * @returns { Node | Node[] } The parsed expression node(s).
2112 * @memberof PRDC_JSLAB_MATHJS_DOC
2113 */
2114
2115 /**
2116 * Create a parser with its own scope and functions.
2117 * @name parser
2118 * @kind function
2119 * @returns { Parser } A new Parser instance.
2120 * @memberof PRDC_JSLAB_MATHJS_DOC
2121 */
2122
2123 /**
2124 * Parser constructor.
2125 * @name Parser
2126 * @kind function
2127 * @returns { Parser } A new Parser instance.
2128 * @memberof PRDC_JSLAB_MATHJS_DOC
2129 */
2130
2131 /**
2132 * Select an element from a matrix or array based on its sorted position.
2133 * @name partitionSelect
2134 * @kind function
2135 * @param { Array | Matrix } x - The input array or matrix.
2136 * @param { number } k - The kth smallest value to select.
2137 * @param { function } [compare] - Optional comparison function.
2138 * @returns { number | BigNumber | Fraction | Complex } The selected
```



```
        element .
2139 * @memberof PRDC_JSLAB_MATHJS_DOC
2140 */
2141
2142 /**
2143 * Calculate the number of permutations of n items taken k at a time.
2144 * @name permutations
2145 * @kind function
2146 * @param { number | BigNumber } n - Total number of items.
2147 * @param { number | BigNumber } [k] - Number of items to choose.
2148 * @returns { number | BigNumber } Number of possible permutations.
2149 * @memberof PRDC_JSLAB_MATHJS_DOC
2150 */
2151
2152 /**
2153 * The golden ratio , approximately 1.618.
2154 * @name phi
2155 * @kind member
2156 * @memberof PRDC_JSLAB_MATHJS_DOC
2157 * @type { number }
2158 */
2159
2160 /**
2161 * The mathematical constant pi.
2162 * @name pi
2163 * @kind member
2164 * @memberof PRDC_JSLAB_MATHJS_DOC
2165 * @type { number }
2166 */
2167
2168 /**
2169 * Randomly pick one or more values from a list .
2170 * @name pickRandom
2171 * @kind function
2172 * @param { Array } array - Array to pick values from.
2173 * @param { number } [number] - Number of values to pick .
2174 * @returns { * | Array } Picked value(s) .
2175 * @memberof PRDC_JSLAB_MATHJS_DOC
2176 */
2177
2178 /**
2179 * Compute the pseudoinverse of a matrix .
2180 * @name pinv
2181 * @kind function
2182 * @param { Array | Matrix } x - A matrix .
2183 * @returns { Matrix } The pseudoinverse of x .
2184 * @memberof PRDC_JSLAB_MATHJS_DOC
2185 */
2186
2187 /**
2188 * Planck charge , in coulombs .
2189 * @name planckCharge
2190 * @kind member
2191 * @memberof PRDC_JSLAB_MATHJS_DOC
2192 * @type { number }
```



```
2193    */
2194
2195 /**
2196 * Planck constant , in joule seconds .
2197 * @name planckConstant
2198 * @kind member
2199 * @memberof PRDC_JSLAB_MATHJS_DOC
2200 * @type { number }
2201 */
2202
2203 /**
2204 * Planck length , in meters .
2205 * @name planckLength
2206 * @kind member
2207 * @memberof PRDC_JSLAB_MATHJS_DOC
2208 * @type { number }
2209 */
2210
2211 /**
2212 * Planck mass , in kilograms .
2213 * @name planckMass
2214 * @kind member
2215 * @memberof PRDC_JSLAB_MATHJS_DOC
2216 * @type { number }
2217 */
2218
2219 /**
2220 * Planck temperature , in kelvin .
2221 * @name planckTemperature
2222 * @kind member
2223 * @memberof PRDC_JSLAB_MATHJS_DOC
2224 * @type { number }
2225 */
2226
2227 /**
2228 * Planck time , in seconds .
2229 * @name planckTime
2230 * @kind member
2231 * @memberof PRDC_JSLAB_MATHJS_DOC
2232 * @type { number }
2233 */
2234
2235 /**
2236 * Find roots of a univariate polynomial .
2237 * @name polynomialRoot
2238 * @kind function
2239 * @param { Array } coefficients - Coefficients of the polynomial .
2240 * @returns { Array } An array of roots .
2241 * @memberof PRDC_JSLAB_MATHJS_DOC
2242 */
2243
2244 /**
2245 * Calculate the power of x to y , x^y .
2246 * @name pow
2247 * @kind function
```

```

2248   * @param { number | BigNumber | Complex | Array | Matrix } x - Base.
2249   * @param { number | BigNumber | Complex | Array | Matrix } y - Exponent.
2250   * @returns { number | BigNumber | Complex | Array | Matrix } x raised to
2251     the power y.
2252   * @memberof PRDC_JSLAB_MATHJS_DOC
2253   */
2254
2255 /**
2256  * Compute the product of a set of values.
2257  * @name prod
2258  * @kind function
2259  * @param { Array | Matrix } array - Input array.
2260  * @param { number | BigNumber } [dim] - Dimension along which to compute
2261    the product.
2262  * @returns { number | BigNumber | Array | Matrix } The product of all
2263    values.
2264  * @memberof PRDC_JSLAB_MATHJS_DOC
2265  */
2266
2267 /**
2268  * Proton mass, in kilograms.
2269  * @name protonMass
2270  * @kind member
2271  * @memberof PRDC_JSLAB_MATHJS_DOC
2272  * @type { number }
2273  */
2274
2275 /**
2276  * Compute the QR decomposition of a matrix.
2277  * @name qr
2278  * @kind function
2279  * @param { Matrix | Array } x - A matrix.
2280  * @returns { Object } An object containing Q and R matrices.
2281  * @memberof PRDC_JSLAB_MATHJS_DOC
2282  */
2283
2284 /**
2285  * Compute the quantile of a sequence.
2286  * @name quantileSeq
2287  * @kind function
2288  * @param { Array | Matrix } data - Input data.
2289  * @param { number | Array } prob - Probability or array of probabilities.
2290  * @param { boolean } [sorted=false] - Is data sorted?
2291  * @returns { number | Array } The quantile(s).
2292  * @memberof PRDC_JSLAB_MATHJS_DOC
2293  */
2294
2295 /**
2296  * Quantum of circulation, in m^2/s.
2297  * @name quantumOfCirculation
2298  * @kind member
2299  * @memberof PRDC_JSLAB_MATHJS_DOC
2299  * @type { number }
2299 */

```



```
2300  /**
2301   * Generate a random integer between min and max.
2302   * @name randomInt
2303   * @kind function
2304   * @param { number | BigNumber } [min] - Minimum value, inclusive.
2305   * @param { number | BigNumber } max - Maximum value, inclusive.
2306   * @returns { number | BigNumber } A random integer.
2307   * @memberof PRDC_JSLAB_MATHJS_DOC
2308   */
2309
2310 /**
2311  * Create a range.
2312  * @name Range
2313  * @kind function
2314  * @param { * } start - Start of the range.
2315  * @param { * } end - End of the range.
2316  * @param { * } [step=1] - Step size.
2317  * @returns { Range } A new Range instance.
2318  * @memberof PRDC_JSLAB_MATHJS_DOC
2319 */
2320
2321 /**
2322  * A node representing a range in the expression tree.
2323  * @name RangeNode
2324  * @kind function
2325  * @param { Node } start - Start node.
2326  * @param { Node } end - End node.
2327  * @param { Node } [step] - Step node.
2328  * @returns { RangeNode } A new RangeNode instance.
2329  * @memberof PRDC_JSLAB_MATHJS_DOC
2330 */
2331
2332 /**
2333  * Rationalize an expression.
2334  * @name rationalize
2335  * @kind function
2336  * @param { string | Node } expr - The expression to rationalize.
2337  * @param { Object } [scope] - Scope of variables.
2338  * @param { Object } [options] - Options object.
2339  * @returns { Object } An object with expression and denominator.
2340  * @memberof PRDC_JSLAB_MATHJS_DOC
2341 */
2342
2343 /**
2344  * Get the real part of a complex number.
2345  * @name re
2346  * @kind function
2347  * @param { number | BigNumber | Complex | Array | Matrix } x - Input
2348  * value.
2349  * @returns { number | BigNumber | Array | Matrix } The real part of x.
2350  * @memberof PRDC_JSLAB_MATHJS_DOC
2351 */
2352 /**
2353  * Reduced Planck constant, in joule seconds.
```

```

2354     * @name reducedPlanckConstant
2355     * @kind member
2356     * @memberof PRDC_JSLAB_MATHJS_DOC
2357     * @type { number }
2358     */
2359
2360 /**
2361 * A node representing a relational operation.
2362 * @name RelationalNode
2363 * @kind function
2364 * @param { string } condition - Relational condition.
2365 * @param { Node } params - Parameters.
2366 * @returns { RelationalNode } A new RelationalNode instance.
2367 * @memberof PRDC_JSLAB_MATHJS_DOC
2368 */
2369
2370 /**
2371 * Replacer function for JSON serialization.
2372 * @name replacer
2373 * @kind function
2374 * @param { string } key - Property name.
2375 * @param { * } value - Property value.
2376 * @returns { * } The transformed value.
2377 * @memberof PRDC_JSLAB_MATHJS_DOC
2378 */
2379
2380 /**
2381 * Resize a matrix.
2382 * @name resize
2383 * @kind function
2384 * @param { Array | Matrix } x - The input matrix.
2385 * @param { Array } size - The new size.
2386 * @param { * } [defaultValue=0] - Default value for new entries.
2387 * @returns { Array | Matrix } The resized matrix.
2388 * @memberof PRDC_JSLAB_MATHJS_DOC
2389 */
2390
2391 /**
2392 * Resolve the value of a symbol or function.
2393 * @name resolve
2394 * @kind function
2395 * @param { string } name - The name to resolve.
2396 * @returns { * } The resolved value.
2397 * @memberof PRDC_JSLAB_MATHJS_DOC
2398 */
2399
2400 /**
2401 * ResultSet constructor.
2402 * @name ResultSet
2403 * @kind function
2404 * @param { Array } entries - The entries in the result set.
2405 * @returns { ResultSet } A new ResultSet instance.
2406 * @memberof PRDC_JSLAB_MATHJS_DOC
2407 */
2408

```

```

2409 /**
2410 * Reviver function for JSON deserialization.
2411 * @name reviver
2412 * @kind function
2413 * @param { string } key - Property name.
2414 * @param { * } value - Property value.
2415 * @returns { * } The transformed value.
2416 * @memberof PRDC_JSLAB_MATHJS_DOC
2417 */
2418
2419 /**
2420 * Bitwise right arithmetic shift operation.
2421 * @name rightArithShift
2422 * @kind function
2423 * @param { number | BigNumber | Array | Matrix } x - The value to shift.
2424 * @param { number | BigNumber | Array | Matrix } y - The amount to shift.
2425 * @returns { number | BigNumber | Array | Matrix } The shifted value.
2426 * @memberof PRDC_JSLAB_MATHJS_DOC
2427 */
2428
2429 /**
2430 * Bitwise right logical shift operation.
2431 * @name rightLogShift
2432 * @kind function
2433 * @param { number | BigNumber | Array | Matrix } x - The value to shift.
2434 * @param { number | BigNumber | Array | Matrix } y - The amount to shift.
2435 * @returns { number | BigNumber | Array | Matrix } The shifted value.
2436 * @memberof PRDC_JSLAB_MATHJS_DOC
2437 */
2438
2439 /**
2440 * Rotate the elements of a matrix.
2441 * @name rotate
2442 * @kind function
2443 * @param { Array | Matrix } x - The input matrix.
2444 * @param { number | BigNumber } [turns=1] - Number of 90-degree rotations
2445 * @returns { Array | Matrix } The rotated matrix.
2446 * @memberof PRDC_JSLAB_MATHJS_DOC
2447 */
2448
2449 /**
2450 * Create a 2D rotation matrix.
2451 * @name rotationMatrix
2452 * @kind function
2453 * @param { number | BigNumber } theta - Rotation angle in radians.
2454 * @returns { Matrix } The rotation matrix.
2455 * @memberof PRDC_JSLAB_MATHJS_DOC
2456 */
2457
2458 /**
2459 * Rydberg constant, in m^-1.
2460 * @name rydberg
2461 * @kind member
2462 * @memberof PRDC_JSLAB_MATHJS_DOC

```

```

2463     * @type { number }
2464     */
2465
2466 /**
2467 * Square root of 1/2.
2468 * @name SQRT1_2
2469 * @kind member
2470 * @memberof PRDC_JSLAB_MATHJS_DOC
2471 * @type { number }
2472 */
2473
2474 /**
2475 * Square root of 2.
2476 * @name SQRT2
2477 * @kind member
2478 * @memberof PRDC_JSLAB_MATHJS_DOC
2479 * @type { number }
2480 */
2481
2482 /**
2483 * Sackur-Tetrode constant at 1 atm, in J/K.
2484 * @name sackurTetrode
2485 * @kind member
2486 * @memberof PRDC_JSLAB_MATHJS_DOC
2487 * @type { number }
2488 */
2489
2490 /**
2491 * Compute the Schur decomposition of a matrix.
2492 * @name schur
2493 * @kind function
2494 * @param { Matrix | Array } A - A square matrix.
2495 * @returns { Object } An object containing matrices U and T.
2496 * @memberof PRDC_JSLAB_MATHJS_DOC
2497 */
2498
2499 /**
2500 * Calculate the secant of a value.
2501 * @name sec
2502 * @kind function
2503 * @param { number | Complex | Unit | Array | Matrix } x - Function input.
2504 * @returns { number | Complex | Array | Matrix } The secant of x.
2505 * @memberof PRDC_JSLAB_MATHJS_DOC
2506 */
2507
2508 /**
2509 * Calculate the hyperbolic secant of a value.
2510 * @name sech
2511 * @kind function
2512 * @param { number | Complex | Array | Matrix } x - Function input.
2513 * @returns { number | Complex | Array | Matrix } The hyperbolic secant of
2514 *         x.
2515 * @memberof PRDC_JSLAB_MATHJS_DOC
2516 */

```

```
2517  /**
2518   * Second radiation constant , in m K .
2519   * @name secondRadiation
2520   * @kind member
2521   * @memberof PRDC_JSLAB_MATHJS_DOC
2522   * @type { number }
2523   */
2524
2525 /**
2526   * Create the Cartesian product of two or more sets .
2527   * @name setCartesian
2528   * @kind function
2529   * @param { ...Array | ...Matrix } sets - The sets to compute the product
2530   * of .
2531   * @returns { Array | Matrix } The Cartesian product .
2532   * @memberof PRDC_JSLAB_MATHJS_DOC
2533   */
2534
2535 /**
2536   * Compute the difference between two sets .
2537   * @name setDifference
2538   * @kind function
2539   * @param { Array | Matrix } a - First set .
2540   * @param { Array | Matrix } b - Second set .
2541   * @returns { Array | Matrix } The difference a \ b .
2542   * @memberof PRDC_JSLAB_MATHJS_DOC
2543   */
2544
2545 /**
2546   * Remove duplicate elements from a set .
2547   * @name setDistinct
2548   * @kind function
2549   * @param { Array | Matrix } a - Input set .
2550   * @returns { Array | Matrix } A set with distinct elements .
2551   * @memberof PRDC_JSLAB_MATHJS_DOC
2552   */
2553
2554 /**
2555   * Compute the intersection of two or more sets .
2556   * @name setIntersect
2557   * @kind function
2558   * @param { Array | Matrix } a - First set .
2559   * @param { Array | Matrix } b - Second set .
2560   * @param { ...Array | ...Matrix } [others] - Additional sets .
2561   * @returns { Array | Matrix } The intersection of the sets .
2562   * @memberof PRDC_JSLAB_MATHJS_DOC
2563   */
2564
2565 /**
2566   * Test whether a set is a subset of another set .
2567   * @name setIsSubset
2568   * @kind function
2569   * @param { Array | Matrix } a - Potential subset .
2570   * @param { Array | Matrix } b - Superset .
2571   * @returns { boolean } Returns true if a is a subset of b .
```

```
2571 * @memberof PRDC_JSLAB_MATHJS_DOC
2572 */
2573
2574 /**
2575 * Count the multiplicity of an element in a multiset.
2576 * @name setMultiplicity
2577 * @kind function
2578 * @param { *} e - Element to count.
2579 * @param { Array | Matrix } multiset - The multiset.
2580 * @returns { number } The multiplicity of e.
2581 * @memberof PRDC_JSLAB_MATHJS_DOC
2582 */
2583
2584 /**
2585 * Compute the power set of a set.
2586 * @name setPowerset
2587 * @kind function
2588 * @param { Array | Matrix } set - The input set.
2589 * @returns { Array | Matrix } The power set.
2590 * @memberof PRDC_JSLAB_MATHJS_DOC
2591 */
2592
2593 /**
2594 * Get the size of a set.
2595 * @name setSize
2596 * @kind function
2597 * @param { Array | Matrix } set - The input set.
2598 * @returns { number } The number of elements.
2599 * @memberof PRDC_JSLAB_MATHJS_DOC
2600 */
2601
2602 /**
2603 * Compute the symmetric difference of two sets.
2604 * @name setSymDifference
2605 * @kind function
2606 * @param { Array | Matrix } a - First set.
2607 * @param { Array | Matrix } b - Second set.
2608 * @returns { Array | Matrix } The symmetric difference.
2609 * @memberof PRDC_JSLAB_MATHJS_DOC
2610 */
2611
2612 /**
2613 * Compute the union of two or more sets.
2614 * @name setUnion
2615 * @kind function
2616 * @param { Array | Matrix } a - First set.
2617 * @param { Array | Matrix } b - Second set.
2618 * @param { ...Array | ...Matrix } [others] - Additional sets.
2619 * @returns { Array | Matrix } The union of the sets.
2620 * @memberof PRDC_JSLAB_MATHJS_DOC
2621 */
2622
2623 /**
2624 * Compute the sign of a number.
2625 * @name sign
```

```

2626     * @kind function
2627     * @param { number | BigNumber | Fraction | Complex | Array | Matrix } x -
2628         Input value.
2629     * @returns { number | BigNumber | Fraction | Complex | Array | Matrix }
2630         The sign of x.
2631     * @memberof PRDC_JSLAB_MATHJS_DOC
2632     */
2633
2634 /**
2635     * Simplify an expression.
2636     * @name simplify
2637     * @kind function
2638     * @param { string | Node } expr - The expression to simplify.
2639     * @param { Object | Array | Function } [rules] - Simplification rules.
2640     * @param { Object } [scope] - Scope for variables.
2641     * @returns { Node } The simplified expression.
2642     * @memberof PRDC_JSLAB_MATHJS_DOC
2643     */
2644
2645 /**
2646     * Simplify an expression containing constants.
2647     * @name simplifyConstant
2648     * @kind function
2649     * @param { Node } expr - The expression to simplify.
2650     * @param { Object } [options] - Simplification options.
2651     * @returns { Node } The simplified expression.
2652     * @memberof PRDC_JSLAB_MATHJS_DOC
2653     */
2654
2655 /**
2656     * Perform core simplifications on an expression.
2657     * @name simplifyCore
2658     * @kind function
2659     * @param { Node } expr - The expression to simplify.
2660     * @returns { Node } The simplified expression.
2661     * @memberof PRDC_JSLAB_MATHJS_DOC
2662     */
2663
2664 /**
2665     * Calculate the sine of a value.
2666     * @name sin
2667     * @kind function
2668     * @param { number | Complex | Unit | Array | Matrix } x - Function input.
2669     * @returns { number | Complex | Array | Matrix } The sine of x.
2670     * @memberof PRDC_JSLAB_MATHJS_DOC
2671     */
2672
2673 /**
2674     * Calculate the hyperbolic sine of a value.
2675     * @name sinh
2676     * @kind function
2677     * @param { number | Complex | Array | Matrix } x - Function input.
2678     * @returns { number | Complex | Array | Matrix } The hyperbolic sine of x
2679     * @memberof PRDC_JSLAB_MATHJS_DOC
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4297
4298
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4497
4498
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4
```

```
2678 */  
2679  
2680 /**  
2681 * Sparse Linear Solver using LU decomposition.  
2682 * @name slu  
2683 * @kind function  
2684 * @param { Matrix | Array } A - A sparse matrix.  
2685 * @param { number } order - Ordering and analysis control parameter.  
2686 * @param { number } threshold - Partial pivoting threshold.  
2687 * @returns { Object } The LU decomposition.  
2688 * @memberof PRDC_JSLAB_MATHJS_DOC  
2689 */  
2690  
2691 /**  
2692 * Test whether value x is smaller than y.  
2693 * @name smaller  
2694 * @kind function  
2695 * @param { number | BigNumber | Fraction | Complex | Unit | string |  
2696 *         Array | Matrix } x - First value.  
2697 * @param { number | BigNumber | Fraction | Complex | Unit | string |  
2698 *         Array | Matrix } y - Second value.  
2699 * @returns { boolean | Array | Matrix } Returns true if x < y.  
2700 * @memberof PRDC_JSLAB_MATHJS_DOC  
2701 */  
2702 /**  
2703 * Test whether value x is smaller than or equal to y.  
2704 * @name smallerEq  
2705 * @kind function  
2706 * @param { number | BigNumber | Fraction | Complex | Unit | string |  
2707 *         Array | Matrix } x - First value.  
2708 * @param { number | BigNumber | Fraction | Complex | Unit | string |  
2709 *         Array | Matrix } y - Second value.  
2710 * @returns { boolean | Array | Matrix } Returns true if x <= y.  
2711 * @memberof PRDC_JSLAB_MATHJS_DOC  
2712 */  
2713 * Compute the Sparse LU decomposition of a matrix.  
2714 * @name Spa  
2715 * @kind function  
2716 * @param { Matrix | Array } A - A sparse matrix.  
2717 * @returns { Object } The LU decomposition.  
2718 * @memberof PRDC_JSLAB_MATHJS_DOC  
2719 */  
2720 /**  
2721 * Create a sparse matrix.  
2722 * @name sparse  
2723 * @kind function  
2724 * @param { Array | Matrix } [data] - The data for the matrix.  
2725 * @param { string } [datatype] - The data type.  
2726 * @returns { SparseMatrix } A new SparseMatrix instance.  
2727 * @memberof PRDC_JSLAB_MATHJS_DOC  
2728 */
```

```

2729
2730 /**
2731 * Sparse matrix constructor.
2732 * @name SparseMatrix
2733 * @kind function
2734 * @param { Object } [data] - The data for the matrix.
2735 * @returns { SparseMatrix } A new SparseMatrix instance.
2736 * @memberof PRDC_JSLAB_MATHJS_DOC
2737 */
2738
2739 /**
2740 * Speed of light in vacuum, in m/s.
2741 * @name speedOfLight
2742 * @kind member
2743 * @memberof PRDC_JSLAB_MATHJS_DOC
2744 * @type { number }
2745 */
2746
2747 /**
2748 * Split a unit into its numeric value and unit string.
2749 * @name splitUnit
2750 * @kind function
2751 * @param { Unit } unit - The unit to split.
2752 * @param { Array } parts - Array of units to split into.
2753 * @returns { Array } Array of units.
2754 * @memberof PRDC_JSLAB_MATHJS_DOC
2755 */
2756
2757 /**
2758 * Calculate the square root of a value.
2759 * @name sqrt
2760 * @kind function
2761 * @param { number | BigNumber | Complex | Unit | Array | Matrix } x -
2762 *     Value for which to calculate the square root.
2763 * @returns { number | BigNumber | Complex | Unit | Array | Matrix } The
2764 *     square root of x.
2765 * @memberof PRDC_JSLAB_MATHJS_DOC
2766 */
2767
2768 /**
2769 * Calculate the principal square root of a matrix.
2770 * @name sqrtm
2771 * @kind function
2772 * @param { Array | Matrix } x - A square matrix.
2773 * @returns { Matrix } The principal square root of x.
2774 * @memberof PRDC_JSLAB_MATHJS_DOC
2775 */
2776
2777 /**
2778 * Compute the square of a value.
2779 * @name square
2780 * @kind function
2781 * @param { number | BigNumber | Complex | Unit | Array | Matrix } x -
2782 *     Input value.
2783 * @returns { number | BigNumber | Complex | Unit | Array | Matrix } The

```

```
    square of x.  
2781 * @memberof PRDC_JSLAB_MATHJS_DOC  
2782 */  
2783  
2784 /**  
2785 * Remove singleton dimensions from an array or matrix.  
2786 * @name squeeze  
2787 * @kind function  
2788 * @param { Array | Matrix } x - The input array or matrix.  
2789 * @returns { Array | Matrix } The squeezed array or matrix.  
2790 * @memberof PRDC_JSLAB_MATHJS_DOC  
2791 */  
2792  
2793 /**  
2794 * Compute the standard deviation of a set of values.  
2795 * @name std  
2796 * @kind function  
2797 * @param { Array | Matrix } array - Input array.  
2798 * @param { string } [normalization='unbiased'] - Normalization mode.  
2799 * @param { number | BigNumber } [dim] - Dimension along which to compute.  
2800 * @returns { number | BigNumber | Array | Matrix } The standard deviation  
2801  
2802 * @memberof PRDC_JSLAB_MATHJS_DOC  
2803 */  
2804  
2805 /**  
2806 * Stefan-Boltzmann constant , in W/(m^2K^4) .  
2807 * @name stefanBoltzmann  
2808 * @kind member  
2809 * @memberof PRDC_JSLAB_MATHJS_DOC  
2810 * @type { number }  
2811 */  
2812  
2813 /**  
2814 * Stirling numbers of the second kind.  
2815 * @name stirlingS2  
2816 * @kind function  
2817 * @param { number | BigNumber } n - Total number of objects.  
2818 * @param { number | BigNumber } k - Number of non-empty subsets.  
2819 * @returns { number | BigNumber } The Stirling number.  
2820 * @memberof PRDC_JSLAB_MATHJS_DOC  
2821 */  
2822  
2823 /**  
2824 * Parse a value into a string.  
2825 * @name string  
2826 * @kind function  
2827 * @param { * } value - The value to convert.  
2828 * @returns { string } The string representation of value.  
2829 * @memberof PRDC_JSLAB_MATHJS_DOC  
2830 */  
2831  
2832 /**  
2833 * Get or set a subset of a matrix or array.  
2834 * @name subset
```

```

2834  * @kind function
2835  * @param { Array | Matrix } x - The input matrix.
2836  * @param { Index } index - The index.
2837  * @param { * } [replacement] - The replacement value.
2838  * @param { boolean } [defaultValue] - Default value for missing entries.
2839  * @returns { * } The subset or updated matrix.
2840  * @memberof PRDC_JSLAB_MATHJS_DOC
2841  */
2842
2843 /**
2844 * Solve the Sylvester equation A*X + X*B = C.
2845 * @name sylvester
2846 * @kind function
2847 * @param { Matrix | Array } A - A square matrix.
2848 * @param { Matrix | Array } B - A square matrix.
2849 * @param { Matrix | Array } C - A matrix.
2850 * @returns { Matrix } Solution X of the Sylvester equation.
2851 * @memberof PRDC_JSLAB_MATHJS_DOC
2852 */
2853
2854 /**
2855 * A node representing a symbol in the expression tree.
2856 * @name SymbolNode
2857 * @kind function
2858 * @param { string } name - The symbol name.
2859 * @returns { SymbolNode } A new SymbolNode instance.
2860 * @memberof PRDC_JSLAB_MATHJS_DOC
2861 */
2862
2863 /**
2864 * Test whether two expressions are symbolically equal.
2865 * @name symbolicEqual
2866 * @kind function
2867 * @param { Node } expr1 - First expression.
2868 * @param { Node } expr2 - Second expression.
2869 * @param { Object } [options] - Comparison options.
2870 * @returns { boolean } Returns true if expressions are symbolically equal
2871
2872 * @memberof PRDC_JSLAB_MATHJS_DOC
2873 */
2874 /**
2875 * Calculate the tangent of a value.
2876 * @name tan
2877 * @kind function
2878 * @param { number | Complex | Unit | Array | Matrix } x - Function input.
2879 * @returns { number | Complex | Array | Matrix } The tangent of x.
2880 * @memberof PRDC_JSLAB_MATHJS_DOC
2881 */
2882
2883 /**
2884 * Calculate the hyperbolic tangent of a value.
2885 * @name tanh
2886 * @kind function
2887 * @param { number | Complex | Array | Matrix } x - Function input.

```

```
2888 * @returns { number | Complex | Array | Matrix } The hyperbolic tangent
2889   of x.
2890 * @memberof PRDC_JSLAB_MATHJS_DOC
2891 */
2892 /**
2893 * The constant tau, equal to 2pi.
2894 * @name tau
2895 * @kind member
2896 * @memberof PRDC_JSLAB_MATHJS_DOC
2897 * @type { number }
2898 */
2899
2900 /**
2901 * Thomson cross section, in m^2.
2902 * @name thomsonCrossSection
2903 * @kind member
2904 * @memberof PRDC_JSLAB_MATHJS_DOC
2905 * @type { number }
2906 */
2907
2908 /**
2909 * Convert a unit to another unit.
2910 * @name to
2911 * @kind function
2912 * @param { Unit } x - The unit to be converted.
2913 * @param { Unit | string } unit - Target unit.
2914 * @returns { Unit } The converted unit.
2915 * @memberof PRDC_JSLAB_MATHJS_DOC
2916 */
2917
2918 /**
2919 * Boolean value true.
2920 * @name true
2921 * @kind member
2922 * @memberof PRDC_JSLAB_MATHJS_DOC
2923 * @type { boolean }
2924 */
2925
2926 /**
2927 * Get the type of a variable.
2928 * @name typeOf
2929 * @kind function
2930 * @param { * } x - The variable.
2931 * @returns { string } The type of x.
2932 * @memberof PRDC_JSLAB_MATHJS_DOC
2933 */
2934
2935 /**
2936 * Create a typed-function.
2937 * @name typed
2938 * @kind function
2939 * @param { string } name - Function name.
2940 * @param { Object } signatures - Function signatures.
2941 * @returns { function } The typed function.
```

```
2942     * @memberof PRDC_JSLAB_MATHJS_DOC
2943     */
2944
2945     /**
2946      * Invert the sign of a value.
2947      * @name unaryMinus
2948      * @kind function
2949      * @param { number | BigNumber | Fraction | Complex | Unit | Array | Matrix } x - Input value.
2950      * @returns { number | BigNumber | Fraction | Complex | Unit | Array | Matrix } The negated value.
2951      * @memberof PRDC_JSLAB_MATHJS_DOC
2952     */
2953
2954     /**
2955      * Unary plus operation.
2956      * @name unaryPlus
2957      * @kind function
2958      * @param { number | BigNumber | Fraction | Complex | Unit | Array | Matrix } x - Input value.
2959      * @returns { number | BigNumber | Fraction | Complex | Unit | Array | Matrix } The input value.
2960      * @memberof PRDC_JSLAB_MATHJS_DOC
2961     */
2962
2963     /**
2964      * Test whether two values are unequal.
2965      * @name unequal
2966      * @kind function
2967      * @param { * } x - First value.
2968      * @param { * } y - Second value.
2969      * @returns { boolean | Array | Matrix } Returns true if x is not equal to y.
2970      * @memberof PRDC_JSLAB_MATHJS_DOC
2971     */
2972
2973     /**
2974      * Unit constructor.
2975      * @name Unit
2976      * @kind function
2977      * @param { number | BigNumber } value - The numeric value.
2978      * @param { string | Unit } unit - The unit string or Unit object.
2979      * @returns { Unit } A new Unit instance.
2980      * @memberof PRDC_JSLAB_MATHJS_DOC
2981     */
2982
2983     /**
2984      * Create a unit.
2985      * @name unit
2986      * @kind function
2987      * @param { number | BigNumber } value - The numeric value.
2988      * @param { string | Unit } unit - The unit string or Unit object.
2989      * @returns { Unit } A new Unit instance.
2990      * @memberof PRDC_JSLAB_MATHJS_DOC
2991     */
```

```

2992
2993 /**
2994 * The base of natural logarithms , e .
2995 * @name E
2996 * @kind member
2997 * @memberof PRDC_JSLAB_MATHJS_DOC
2998 * @type { number }
2999 */
3000
3001 /**
3002 * The mathematical constant pi .
3003 * @name PI
3004 * @kind member
3005 * @memberof PRDC_JSLAB_MATHJS_DOC
3006 * @type { number }
3007 */
3008
3009 /**
3010 * Solve a linear system A * x = b where A is an upper triangular matrix .
3011 * @name usolve
3012 * @kind function
3013 * @param { Matrix | Array } U - An upper triangular matrix .
3014 * @param { Matrix | Array } b - A column vector .
3015 * @returns { Matrix | Array } The solution vector x .
3016 * @memberof PRDC_JSLAB_MATHJS_DOC
3017 */
3018
3019 /**
3020 * Find all solutions of a linear system A * x = b where A is an upper
3021 * triangular matrix .
3022 * @name usolveAll
3023 * @kind function
3024 * @param { Matrix | Array } U - An upper triangular matrix .
3025 * @param { Matrix | Array } b - A column vector .
3026 * @returns { Array } An array of solutions .
3027 * @memberof PRDC_JSLAB_MATHJS_DOC
3028 */
3029
3030 /**
3031 * Vacuum impedance , in ohms .
3032 * @name vacuumImpedance
3033 * @kind member
3034 * @memberof PRDC_JSLAB_MATHJS_DOC
3035 * @type { number }
3036 */
3037
3038 /**
3039 * Compute the variance of a set of values .
3040 * @name variance
3041 * @kind function
3042 * @param { Array | Matrix } array - Input array .
3043 * @param { string } [ normalization='unbiased' ] - Normalization mode .
3044 * @param { number | BigNumber } [ dim ] - Dimension along which to compute .
3045 * @returns { number | BigNumber | Array | Matrix } The variance .
3046 * @memberof PRDC_JSLAB_MATHJS_DOC

```

```

3046     */
3047
3048 /**
3049 * Weak mixing angle.
3050 * @name weakMixingAngle
3051 * @kind member
3052 * @memberof PRDC_JSLAB_MATHJS_DOC
3053 * @type { number }
3054 */
3055
3056 /**
3057 * Wien displacement constant , in m K .
3058 * @name wienDisplacement
3059 * @kind member
3060 * @memberof PRDC_JSLAB_MATHJS_DOC
3061 * @type { number }
3062 */
3063
3064 /**
3065 * Extended greatest common divisor for integers.
3066 * @name xgcd
3067 * @kind function
3068 * @param { number | BigNumber } a - An integer.
3069 * @param { number | BigNumber } b - An integer.
3070 * @returns { Array } An array [ gcd , x , y ] satisfying gcd = a*x + b*y .
3071 * @memberof PRDC_JSLAB_MATHJS_DOC
3072 */
3073
3074 /**
3075 * Logical XOR operation .
3076 * @name xor
3077 * @kind function
3078 * @param { boolean | Array | Matrix } x - First value .
3079 * @param { boolean | Array | Matrix } y - Second value .
3080 * @returns { boolean | Array | Matrix } The logical XOR of x and y .
3081 * @memberof PRDC_JSLAB_MATHJS_DOC
3082 */
3083
3084 /**
3085 * Error thrown when an incorrect number of arguments is passed .
3086 * @name ArgumentsError
3087 * @kind function
3088 * @param { string } fn - Function name .
3089 * @param { number } count - Actual argument count .
3090 * @param { number } min - Minimum required arguments .
3091 * @param { number } [max] - Maximum allowed arguments .
3092 * @returns { ArgumentsError } A new ArgumentsError instance .
3093 * @memberof PRDC_JSLAB_MATHJS_DOC
3094 */
3095
3096 /**
3097 * Error thrown when matrix dimensions mismatch .
3098 * @name DimensionError
3099 * @kind function
3100 * @param { number } actual - Actual dimension .

```



```
3101     * @param { number } expected - Expected dimension.
3102     * @param { string } [relation] - Relation between dimensions.
3103     * @returns { DimensionError } A new DimensionError instance.
3104     * @memberof PRDC_JSLAB_MATHJS_DOC
3105     */
3106
3107    /**
3108     * Error thrown when an index is out of range.
3109     * @name IndexError
3110     * @kind function
3111     * @param { number } index - The invalid index.
3112     * @param { number } min - Minimum allowed index.
3113     * @param { number } max - Maximum allowed index.
3114     * @returns { IndexError } A new IndexError instance.
3115     * @memberof PRDC_JSLAB_MATHJS_DOC
3116     */
3117 }
3118 }
```

Listing 111 - mathjs-doc.js

```
1  /**
2   * @file JSLAB library matrix math submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB matrix math submodule.
10  */
11 class PRDC_JSLAB_MATRIX_MATH {
12
13  /**
14   * Constructs a matrix math submodule object with access to JSLAB's math
15   * functions.
16   * @constructor
17   * @param {Object} jsl - Reference to the main JSLAB object.
18   */
19 constructor(jsl) {
20     this.jsl = jsl;
21 }
22 /**
23  * Creates a new matrix.
24  * @param {Array} A - The matrix data.
25  * @param {number} rows - Number of rows.
26  * @param {number} cols - Number of columns.
27  * @returns {PRDC_JSLAB_MATRIX} A new matrix instance.
28  */
29 new(A, rows, cols) {
30   return new PRDC_JSLAB_MATRIX(this.jsl, A, rows, cols);
31 }
32 /**
33  * Creates a matrix filled with a specific value.
```

```
35     * @param {number} v - The value to fill the matrix with.
36     * @param {number} rows - Number of rows.
37     * @param {number} [cols=rows] - Number of columns.
38     * @returns {PRDC_JSLAB_MATRIX} The filled matrix.
39     */
40    fill(v, rows, cols) {
41      if(!cols) {
42        cols = rows;
43      }
44      return new PRDC_JSLAB_MATRIX(this.jsl,
45        this.jsl.array.createFilledArray(rows * cols, v), rows, cols);
46    }
47
48    /**
49     * Creates a matrix filled with NaN values.
50     * @param {number} rows - Number of rows.
51     * @param {number} [cols=rows] - Number of columns.
52     * @returns {PRDC_JSLAB_MATRIX} The NaN-filled matrix.
53     */
54    NaNs(rows, cols) {
55      if(!cols) {
56        cols = rows;
57      }
58      return this.fill(NaN, rows, cols);
59    }
60
61    /**
62     * Creates a matrix filled with ones.
63     * @param {number} rows - Number of rows.
64     * @param {number} [cols=rows] - Number of columns.
65     * @returns {PRDC_JSLAB_MATRIX} The ones-filled matrix.
66     */
67    ones(rows, cols) {
68      if(!cols) {
69        cols = rows;
70      }
71      return this.fill(1, rows, cols);
72    }
73
74    /**
75     * Creates a matrix filled with zeros.
76     * @param {number} rows - Number of rows.
77     * @param {number} [cols=rows] - Number of columns.
78     * @returns {PRDC_JSLAB_MATRIX} The zeros-filled matrix.
79     */
80    zeros(rows, cols) {
81      if(!cols) {
82        cols = rows;
83      }
84      return this.fill(0, rows, cols);
85    }
86
87    /**
88     * Creates a diagonal matrix from an array.
89     * @param {Array} A - The array to create the diagonal matrix from.
```

```

90  * @returns {PRDC_JSLAB_MATRIX} The diagonal matrix.
91  */
92 diag(A) {
93   return new PRDC_JSLAB_MATRIX(this.jsl,
94     this.jsl.array.diag(A, A.length), A.length, A.length);
95 }
96
97 /**
98 * Creates an identity matrix of a given size.
99 * @param {number} size - The size of the identity matrix.
100 * @returns {PRDC_JSLAB_MATRIX} The identity matrix.
101 */
102 eye(size) {
103   return new PRDC_JSLAB_MATRIX(this.jsl,
104     this.jsl.array.diag(this.jsl.array.ones(size), size), size, size);
105 }
106
107 /**
108 * Concatenates multiple matrices vertically (row-wise).
109 * @param {...PRDC_JSLAB_MATRIX} args - Matrices to concatenate.
110 * @returns {PRDC_JSLAB_MATRIX} The concatenated matrix.
111 */
112 concatRow(...args) {
113   var N = args.length;
114   var cols = args[0].cols;
115   var rows = args.reduce((a, e) => a += e.rows, 0);
116   var A = new Array(cols * rows).fill(0);
117
118   var p = 0;
119   for(var j = 0; j < cols; j++) {
120     for(var k = 0; k < N; k++) {
121       var P = args[k].data.length / cols;
122       for(var i = 0; i < P; i++) {
123         A[p++] = args[k].data[j * P + i];
124       }
125     }
126   }
127   return this.new(A, rows, cols);
128 }
129
130 /**
131 * Concatenates multiple matrices horizontally (column-wise).
132 * @param {...PRDC_JSLAB_MATRIX} args - Matrices to concatenate.
133 * @returns {PRDC_JSLAB_MATRIX} The concatenated matrix.
134 */
135 concatCol(...args) {
136   var rows = args[0].rows;
137   var A = args.map((a) => a.data).flat();
138   return this.new(A, rows, A.length / rows);
139 }
140
141 /**
142 * Checks if the provided object is a matrix.
143 * @param {Object} A - The object to check.
144 * @returns {boolean} True if A is a matrix, else false.

```



```
145     */
146     isMatrix(A) {
147         return A instanceof PRDC_JSLAB_MATRIX;
148     }
149 }
150
151 exports.PRDC_JSLAB_MATRIX_MATH = PRDC_JSLAB_MATRIX_MATH;
152
153 /**
154 * Class for JSLAB matrix.
155 */
156 class PRDC_JSLAB_MATRIX {
157
158     #jsl;
159     #rows;
160     #cols;
161
162     /**
163      * Constructs a JSLAB matrix.
164      * @constructor
165      * @param {Object} js1 - Reference to the main JSLAB object .
166      * @param {Array} A - The matrix data.
167      * @param {number} rows - Number of rows.
168      * @param {number} cols - Number of columns.
169      */
170     constructor(js1, A, rows, cols) {
171         this.#jsl = js1;
172         this._set(A, rows, cols);
173     }
174
175     /**
176      * Sets the matrix data and dimensions.
177      * @param {Array} A - The matrix data.
178      * @param {number} rows - Number of rows.
179      * @param {number} cols - Number of columns.
180      */
181     _set(A, rows, cols) {
182         this.rows = rows;
183         this.cols = cols;
184         if(!rows) {
185             this.rows = A.length;
186         }
187         if(!cols) {
188             this.cols = A[0].length;
189         }
190         if(Array.isArray(A[0])) {
191             this.data = this.#jsl.array.transpose(A.flat(), this.rows, this.cols);
192         } else {
193             this.data = [...A];
194         }
195     }
196
197     /**
198      * Extracts a specific column from a matrix.
199      * @param {number} index - The index of the column to extract.
```

```
200     * @returns {Array} The extracted column as an array.  
201     */  
202     column(index) {  
203         return this.#jsl.array.column(this.toArray(), index);  
204     }  
205  
206     /**
207      * Extracts a specific row from a matrix.  
208      * @param {number} index - The index of the row to extract.  
209      * @returns {Array} The extracted row as an array.  
210      */  
211     row(index) {  
212         return this.#jsl.array.row(this.toArray(), index);  
213     }  
214  
215     /**
216      * Gets the length of the matrix along a specified dimension.  
217      * @param {number} dim - The dimension (0 for rows, 1 for columns).  
218      * @returns {number} The length along the specified dimension.  
219      */  
220     length(dim) {  
221         return this.size(dim);  
222     }  
223  
224     /**
225      * Gets the number of elements in the matrix.  
226      * @returns {number} The number of elements.  
227      */  
228     numel() {  
229         return this.rows * this.cols;  
230     }  
231  
232     /**
233      * Gets the size of the matrix along a specified dimension.  
234      * @param {number} dim - The dimension (0 for rows, 1 for columns).  
235      * @returns {number} The size along the specified dimension.  
236      */  
237     size(dim) {  
238         var s = [this.rows, this.cols];  
239         if(typeof dim == 'undefined') {  
240             return s;  
241         }  
242         return s[dim];  
243     }  
244  
245     /**
246      * Reshapes the matrix to the specified dimensions.  
247      * @param {number} rows - New number of rows.  
248      * @param {number} cols - New number of columns.  
249      * @returns {PRDC_JSLAB_MATRIX} The reshaped matrix.  
250      */  
251     reshape(rows, cols) {  
252         return this.#jsl.mat.new(this.#jsl.array.reshape(this.data,  
253             rows, cols), rows, cols);  
254     }
```



```
255
256 /**
257 * Replicates the matrix a specified number of times.
258 * @param {number} rowReps - Number of row repetitions.
259 * @param {number} colReps - Number of column repetitions.
260 * @returns {PRDC_JSLAB_MATRIX} The replicated matrix.
261 */
262 repmat(rowReps, colReps) {
263     var cols;
264     if(colReps > 1) {
265         cols = this.#jsl.array.createFilledArray(colReps, this);
266         cols = this.#jsl.mat.concatCol(...cols);
267     } else {
268         cols = this;
269     }
270
271     var A;
272     if(rowReps > 1) {
273         var rows = this.#jsl.array.createFilledArray(rowReps, cols);
274         A = this.#jsl.mat.concatRow(...rows);
275     } else {
276         A = cols;
277     }
278
279     return A;
280 }
281
282 /**
283 * Transposes the matrix.
284 * @returns {PRDC_JSLAB_MATRIX} The transposed matrix.
285 */
286 transpose() {
287     return this.#jsl.mat.new(this.#jsl.array.transpose(this.data, this.rows,
288             this.cols),
289             this.cols, this.rows);
290 }
291
292 /**
293 * Computes the inverse of the matrix.
294 * @returns {PRDC_JSLAB_MATRIX} The inverse matrix.
295 */
296 inv() {
297     return this.#jsl.mat.new(this.#jsl.env.math.inv(this.toArray()));
298 }
299
300 /**
301 * Computes the determinant of the matrix.
302 * @returns {number} The determinant.
303 */
304 det() {
305     return this.#jsl.env.math.det(this.toArray());
306 }
307
308 /**
309 * Computes the trace of the matrix (sum of diagonal elements).
```

```
309     * @returns {number} The trace of the matrix.  
310     */  
311   trace() {  
312     return this.#jsl.env.math.trace(this.toArray());  
313   }  
314  
315   /**
316    * Computes the Frobenius norm of the matrix.  
317    * @returns {number} The Frobenius norm.  
318    */  
319   norm(p = 2) {  
320     return this.#jsl.env.math.norm(this.toArray(), p);  
321   }  
322  
323   /**
324    * Raises the matrix to a power.  
325    * @param {number} p - The exponent.  
326    * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.  
327    */  
328   powm(p) {  
329     return this.#jsl.mat.new(this.#jsl.env.math.pow(this.toArray(), p));  
330   }  
331  
332   /**
333    * Computes the matrix exponential.  
334    * @returns {PRDC_JSLAB_MATRIX} The exponential matrix.  
335    */  
336   expm() {  
337     return this.#jsl.mat.new(this.#jsl.env.math.expm(this.toArray())._data);  
338   }  
339  
340   /**
341    * Adds two matrices.  
342    * @param {PRDC_JSLAB_MATRIX} A - The matrix to add.  
343    * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.  
344    */  
345   add(A) {  
346     return this.#jsl.mat.new(this.#jsl.array.plus(this.data, A.data),  
347       this.rows, this.cols);  
348   }  
349  
350   /**
351    * Adds two matrices (alias for add).  
352    * @param {PRDC_JSLAB_MATRIX} A - The matrix to add.  
353    * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.  
354    */  
355   plus(A) {  
356     return this.add(A);  
357   }  
358  
359   /**
360    * Subtracts matrix A from the current matrix.  
361    * @param {PRDC_JSLAB_MATRIX} A - The matrix to subtract.  
362    * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.  
363    */
```

```

364  subtract(A) {
365      return this.#jsl.mat.new(this.#jsl.array.minus(this.data, A.data),
366          this.rows, this.cols);
367  }
368
369  /**
370   * Subtracts matrix A from the current matrix (alias for subtract).
371   * @param {PRDC_JSLAB_MATRIX} A - The matrix to subtract.
372   * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
373   */
374  minus(A) {
375      return this.subtract(A);
376  }
377
378  /**
379   * Multiplies two matrices.
380   * @param {PRDC_JSLAB_MATRIX} A - The matrix to multiply with.
381   * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
382   */
383  multiply(A) {
384      return this.#jsl.mat.new(this.#jsl.array.multiply(this.data, A.data, this.
385          rows, this.cols, A.cols), this.rows, A.cols);
386  }
387
388  /**
389   * Solves a linear system.
390   * @param {PRDC_JSLAB_MATRIX} B - The right-hand side matrix.
391   * @returns {PRDC_JSLAB_MATRIX} The solution matrix.
392   */
393  linsolve(B) {
394      return this.#jsl.mat.new(this.#jsl.array.linsolve(this.data, B.data, this.
395          cols), this.rows, 1);
396  }
397
398  /**
399   * Divides each element by another matrix or scalar.
400   * @param {PRDC_JSLAB_MATRIX|number} A - The matrix or scalar to divide by.
401   * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
402   */
403  divideEl(A) {
404      if(this.#jsl.mat.isMatrix(A)) {
405          return this.#jsl.mat.new(this.#jsl.array.divideEl(this.data, A.data),
406              this.rows, this.cols);
407      } else {
408          return this.#jsl.mat.new(this.#jsl.array.scale(this.data, 1 / A),
409              this.rows, this.cols);
410      }
411
412  /**
413   * Multiplies each element by another matrix or scalar.
414   * @param {PRDC_JSLAB_MATRIX|number} A - The matrix or scalar to multiply by
415   * .
416   * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
417   */

```

```

416 multiplyEl(A) {
417     if(this.#jsl.mat.isMatrix(A)) {
418         return this.#jsl.mat.new(this.#jsl.array.multiplyEl(this.data, A.data),
419             this.rows, this.cols);
420     } else {
421         return this.#jsl.mat.new(this.#jsl.array.scale(this.data, A),
422             this.rows, this.cols);
423     }
424 }
425
426 /**
427 * Raises each element to a power.
428 * @param {number} p - The exponent.
429 * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
430 */
431 powEl(p) {
432     return this.#jsl.mat.new(this.#jsl.array.powEl(this.data, p),
433         this.rows, this.cols);
434 }
435
436 /**
437 * Applies a function to each element of the matrix.
438 * @param {function} func - The function to apply.
439 * @returns {PRDC_JSLAB_MATRIX} The resulting matrix.
440 */
441 elementWise(func) {
442     return this.#jsl.mat.new(this.#jsl.array.elementWise((a) => func(a), this.
443         data),
444         this.rows, this.cols);
445 }
446
447 /**
448 * Computes the reciprocal of each element in the matrix.
449 * @returns {PRDC_JSLAB_MATRIX} The matrix with reciprocals.
450 */
451 reciprocal() {
452     return this.#jsl.mat.new(this.#jsl.array.reciprocal(this.data, this.rows *
453         this.cols),
454         this.rows, this.cols);
455 }
456
457 /**
458 * Computes the sum of all elements in the matrix.
459 * @returns {number} The sum of all elements.
460 */
461 sum() {
462     return this.#jsl.env.math.sum(this.toArray());
463 }
464
465 /**
466 * Sorts the elements of the matrix.
467 * @param {string} [order='asc'] - The order of sorting ('asc' or 'desc').
468 * @returns {PRDC_JSLAB_MATRIX} The sorted matrix.
469 */
470 sort() {

```



```
469     return this.#jsl.env.math.sort(this.data);
470 }
471
472 /**
473 * Finds the minimum element in the matrix.
474 * @returns {number} The minimum value.
475 */
476 min() {
477     return this.#jsl.env.math.min(this.toArray());
478 }
479
480 /**
481 * Finds the maximum element in the matrix.
482 * @returns {number} The maximum value.
483 */
484 max() {
485     return this.#jsl.env.math.max(this.toArray());
486 }
487
488 /**
489 * Creates a clone of the current matrix.
490 * @returns {PRDC_JSLAB_MATRIX} A cloned matrix instance.
491 */
492 clone() {
493     return this.#jsl.mat.new(this.data, this.rows, this.cols);
494 }
495
496 /**
497 * Retrieves elements based on row and column indices.
498 * @param {...(number|_)} args - Row and column indices.
499 * @returns {Array} The selected elements.
500 */
501 index(rows_in, cols_in) {
502     var rows;
503     var cols;
504     if (rows_in === _) {
505         rows = range(0, this.rows - 1);
506     } else {
507         rows = rows_in;
508     }
509     if (cols_in === _) {
510         cols = range(0, this.cols - 1);
511     } else {
512         cols = cols_in;
513     }
514     if (!Array.isArray(rows)) {
515         rows = [rows];
516     }
517     if (!Array.isArray(cols)) {
518         cols = [cols];
519     }
520     var A = zeros(rows.length * cols.length);
521
522     var k = 0;
523     for (var j = 0; j < cols.length; j++) {
```

```

524     for (var i = 0; i < rows.length; i++) {
525         A[k++] = rows[i] + cols[j] * this.rows;
526     }
527 }
528
529     return A;
530 }
531
532 /**
533 * Sets a subset of the matrix elements.
534 * @param {...} args - Indices and values to set.
535 */
536 setSub(...args) {
537     var A = args.map(function(a) {
538         if (!Array.isArray(a) && a !== _) {
539             a = [a];
540         }
541         return a;
542     });
543
544     var indices;
545     var B;
546     if (A.length === 3) {
547         indices = this.index(A[0], A[1]);
548         B = A[2];
549     } else {
550         indices = A[0];
551         if (indices === _) {
552             indices = range(0, this.rows * this.cols - 1);
553         }
554         B = A[1];
555     }
556
557     var j = 0;
558     for (var i = 0; i < indices.length; i++) {
559         if (this.#jsl.mat.isMatrix(B[0])) {
560             this.data[indices[i]] = B[0].data[j++];
561         } else if (Array.isArray(B) && B.length === indices.length) {
562             this.data[indices[i]] = B[j++];
563         } else {
564             this.data[indices[i]] = B[0];
565         }
566     }
567 }
568
569 /**
570 * Gets a subset of the matrix elements.
571 * @param {...} args - Indices to retrieve.
572 * @returns {PRDC_JSLAB_MATRIX} The subset matrix.
573 */
574 getSub(...args) {
575     var indices;
576     var rows;
577     var cols;
578     var A = args.map(function(a) {

```

```

579     if (!Array.isArray(a) && a !== _) {
580         a = [a];
581     }
582     return a;
583 });
584
585     if (A.length === 1) {
586         indices = A[0];
587         rows = indices.length;
588         cols = 1;
589     } else {
590         if (A[0] === _) {
591             rows = this.rows;
592         } else {
593             rows = A[0].length;
594         }
595         if (A[1] === _) {
596             cols = this.cols;
597         } else {
598             cols = A[1].length;
599         }
600         indices = this.index(A[0], A[1]);
601     }
602
603     var B = this.#jsl.array.createFilledArray(indices.length, 0);
604     var j = 0;
605     for(var i = 0; i < indices.length; i++) {
606         B[j++] = this.data[indices[i]];
607     }
608     return this.#jsl.mat.new(B, rows, cols);
609 }
610
611 /**
612 * Converts the matrix to a two-dimensional array.
613 * @returns {Array} The matrix data as a two-dimensional array.
614 */
615 toArray() {
616     return this.#jsl.array.reshape(this.data, this.rows, this.cols);
617 }
618
619 /**
620 * Converts the matrix to a one-dimensional array.
621 * @returns {Array} The matrix data as a one-dimensional array.
622 */
623 toFlatArray() {
624     return this.data;
625 }
626
627 /**
628 * Returns a string representation of the matrix.
629 * @returns {string} The string representation of the matrix.
630 */
631 toString() {
632     var str = 'Matrix([ \n';
633     for(var i = 0; i < this.rows; i++) {

```

```

634         str += ' [ ';
635         for(var j = 0; j < this.cols; j++) {
636             str += this.data[j * this.rows + i] + ', ';
637         }
638         str = str.slice(0, -2);
639         str += '],\n';
640     }
641     str = str.slice(0, -2);
642     str += '\n])';
643     return str;
644 }
645
646 /**
647 * Returns a JSON representation of the matrix.
648 * @returns {Array} The matrix data as a two-dimensional array.
649 */
650 toJSON() {
651     return this.toArray();
652 }
653
654 /**
655 * Returns a safe JSON representation of the matrix.
656 * @returns {Array} The matrix data as a two-dimensional array.
657 */
658 toSafeJSON() {
659     return this.toJSON();
660 }
661
662 /**
663 * Returns a pretty string representation of the matrix.
664 * @returns {string} The pretty string representation of the matrix.
665 */
666 toPrettyString() {
667     return this.toString();
668 }
669 }
```

Listing 112 - matrix-math.js

```

1 /**
2 * @file JSLAB library mechanics submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB mechanics submodule.
10 */
11 class PRDC_JSLAB_LIB_MECHANICS {
12
13 /**
14 * Constructs a mechanics submodule object with access to JSLAB's device
15 * functions.
16 * @constructor
17 * @param {Object} js1 - Reference to the main JSLAB object.
18 }
```

```

17  /*
18   * constructor(jsl) {
19     var obj = this;
20     this.jsl = jsl;
21   }
22
23 /**
24  * Plots beam diagrams.
25  * @param {Array} data Array of objects with x, y, title, xlabel, ylabel
26  * @param {Object} [opts_in] Extra plotting options
27  * @returns {Promise<{extrems: String[], context: Object}>}
28 */
29 async plotBeamDiagrams(data, opts_in = {}) {
30   var context = await this.windows.openPlotlyjs();
31   context.setTitle('Beam Diagrams');
32   var plot_config = {
33     responsive: true,
34     scrollZoom: true,
35     modeBarButtonsToAdd: [
36       name: 'Download plot as a svg',
37       icon: context.Plotly.Icons.camera,
38       click: function(gd) {
39         context.Plotly.downloadImage(gd, {format: 'svg'});
40       }
41     ]
42   };
43
44   var opts = {
45     n: 50,
46     digits: 2,
47     font: {
48       family: 'Roboto',
49       size: 18
50     },
51     ...opts_in
52   };
53   var extrems = [];
54
55   var n = data.length;
56   var traces = [];
57   for(var i = 0; i < n; i++) {
58     var [y_max, I_max] = maxi(data[i].y);
59     extrems.push(`${data[i].title}: ${data[i].ylabel[0]}_max = ${y_max}.
60    toFixed(opts.digits)} ${data[i].ylabel[1]}, ${data[i].xlabel[0]} = ${data[i].x[I_max].toFixed(opts.digits)} ${data[i].xlabel[1]}`);
61
62     var trace = {
63       x: data[i].x,
64       y: data[i].y,
65       cliponaxis: false,
66       mode: 'lines+markers',
67       line: {
68         color: "#000",
69         width: 2
70       },
71     },
72
73   }
74
75   var layout = {
76     title: 'Beam Diagrams',
77     xaxis: {
78       title: 'X-axis',
79       tick0: 0,
80       tickStep: 10
81     },
82     yaxis: {
83       title: 'Y-axis',
84       tick0: 0,
85       tickStep: 10
86     }
87   };
88
89   context.layout = layout;
90   context.addTraces(traces);
91   context.addExtrems(extrems);
92 }
```

```

70      xaxis: 'x' + (i > 0 ? (i+1) : ''),
71      yaxis: 'y' + (i > 0 ? (i+1) : ''),
72      showlegend: false
73  };
74  traces.push(trace);
75
76  var xq = this.jsl.array.linspace(data[i].x[0], end(data[i].x), opts.n);
77  var yq = this.jsl.math.interp(data[i].x, data[i].y, xq);
78  var stem_trace = {
79    x: xq.flatMap(x => [x, x, null]),
80    y: yq.flatMap(y => [0, y, null]),
81    cliponaxis: false,
82    mode: 'lines',
83    line: {
84      color: "#000",
85      width: 1
86    },
87    xaxis: 'x' + (i > 0 ? (i+1) : ''),
88    yaxis: 'y' + (i > 0 ? (i+1) : ''),
89    showlegend: false
90  };
91  traces.push(stem_trace);
92}
93
94 var plot_layout = {
95   grid: {
96     rows: n,
97     columns: 1,
98     pattern: 'independent',
99     ygap: 0.25
100   },
101   title: {},
102   autosize: true,
103   plot_bgcolor: "#fff",
104   paper_bgcolor: "#fff",
105   showlegend: false,
106   font: opts.font,
107   margin: {
108     l: 60,
109     r: 15,
110     b: 60,
111     t: 15,
112     pad: 0
113   }
114 };
115
116 for(var i = 0; i < n; i++) {
117   plot_layout['xaxis' + (i > 0 ? (i+1) : '')] = {
118     showgrid: true,
119     zeroline: true,
120     showline: true,
121     automargin: true,
122     mirror: 'ticks',
123     ticks: 'inside',
124     ticklen: 8,

```

```

125     tickwidth: 0.5,
126     tickcolor: '#000',
127     linecolor: '#000',
128     linewidth: 0.5,
129     layer: "below traces",
130     exponentformat: 'power',
131   };
132   if(i == (n-1)) {
133     plot_layout['xaxis' + (i > 0 ? (i+1) : '')].title = {
134       text: '${data[i].xlabel[0]} ${data[i].xlabel[1]}',
135       standoff: 0
136     };
137   }
138
139   plot_layout['yaxis' + (i > 0 ? (i+1) : '')] = {
140     showgrid: true,
141     zeroline: true,
142     showline: true,
143     automargin: true,
144     mirror: 'ticks',
145     ticks: 'inside',
146     ticklen: 8,
147     tickwidth: 0.5,
148     tickcolor: '#000',
149     linecolor: '#000',
150     linewidth: 0.5,
151     layer: "below traces",
152     title: {
153       text: '${data[i].ylabel[0]} ${data[i].ylabel[1]}',
154     },
155     exponentformat: 'power'
156   };
157 }
158 context.Plotly.newPlot(context.plot_cont, traces, plot_layout, plot_config
  );
159 return {extrems, context};
160 }
161 }
162
163 exports.PRDC_JSLAB_LIB_MECHANICS = PRDC_JSLAB_LIB_MECHANICS;

```

Listing 113 - mechanics.js

```

1 /**
2  * @file JSLAB library networking TCP submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9  * Class representing a TCP client for handling network communications.
10 */
11 class PRDC_JSLAB_TCP_CLIENT {
12
13 /**

```

```
14 * Creates an instance of the TCP client.
15 * @param {Object} js1 Reference to the main JSLAB object.
16 * @param {string} host - The host IP address or hostname to connect to.
17 * @param {number} port - The port number to connect to.
18 * @param {function} onConnectCallback - A callback to execute upon
19 *         successful connection.
20 */
21 constructor(js1, host, port, onConnectCallback) {
22     var obj = this;
23     this.js1 = js1;
24     this.host = host;
25     this.port = port;
26     this.onConnectCallback = onConnectCallback;
27     this.onDataCallback;
28     this.onErrorCallback;
29
30     this.buffer = [];
31     this.active = false;
32
33     this._data_callback = false;
34
35     this.com = this.js1.env.net.createConnection(port, host);
36     this.com.setTimeout(0);
37     this.com.on('connect', function() {
38         obj._onConnect();
39     });
40     this.com.on('data', function(data) {
41         obj._onData(data);
42     });
43     this.com.on('error', function(err) {
44         obj._onError(err);
45     });
46     this.com.on('close', function(err) {
47         if(err) {
48             obj._onError(err);
49         }
50     });
51     this.com.on('end', function() {
52         obj.active = false;
53     });
54
55     this.js1.addForCleanup(this, function() {
56         obj.close();
57     });
58
59 /**
60 * Handles successful connection establishment by setting the client's
61 * active status to true.
62 */
63 _onConnect() {
64     this.active = true;
65     if(this.js1.format.isFunction(this.onConnectCallback)) {
66         this.onConnectCallback();
67     }
68 }
```

```
67     }
68
69     /**
70      * Handles errors by setting the client's active status to false and
71      * possibly logging the error.
72      * @param {Error} err - The error object that was thrown.
73      */
74     _onError(err) {
75       this.active = false;
76       if(this.jsl.format.isFunction(this.onErrorCallback)) {
77         this.onErrorCallback(err);
78       }
79
80     /**
81      * Handles incoming data by appending it to the buffer.
82      * @param {Buffer} data - The incoming data buffer.
83      */
84     _onData(data) {
85       if(this._data_callback) {
86         this.onDataCallback(data);
87       } else {
88         this.buffer.push(...data);
89       }
90     }
91
92     /**
93      * Sets the callback function to handle incoming data events.
94      * @param {Function} callback - The function to be called when data is
95      * received.
96      */
97     setOnData(callback) {
98       if(this.jsl.format.isFunction(callback)) {
99         this.buffer = [];
100        this.onDataCallback = callback;
101        this._data_callback = true;
102      }
103
104    /**
105      * Sets the callback function to handle error events.
106      * @param {Function} callback - The function to be called when an error
107      * occurs.
108      */
109     setOnError(callback) {
110       if(this.jsl.format.isFunction(callback)) {
111         this.onErrorCallback = callback;
112       }
113
114    /**
115      * Enables or disables keep-alive functionality on the underlying socket.
116      * @param {boolean} [enable=true] - Whether to enable keep-alive.
117      * @param {number} [initialDelay=0] - Delay in milliseconds before the first
keep-alive probe is sent.
```

```

118     */
119     setKeepAlive(... args) {
120         this.com.setKeepAlive(... args);
121     }
122
123     /**
124      * Disables the Nagle algorithm, allowing data to be sent immediately.
125      * @param {boolean} [noDelay=true] - Whether to disable the Nagle algorithm.
126      */
127     setNoDelay(... args) {
128         this.com.setNoDelay(... args);
129     }
130
131     /**
132      * Sets the socket timeout for inactivity.
133      * @param {number} timeout - Timeout in milliseconds.
134      * @param {Function} [callback] - Optional callback triggered on timeout.
135      */
136     setTimeout(... args) {
137         this.com.setTimeout(... args);
138     }
139
140     /**
141      * Reads a specified number of bytes from the buffer.
142      * @param {number} [N=Infinity] - The number of bytes to read. Reads all
143      *       available bytes if not specified.
144      * @returns {Buffer} The data read from the buffer.
145      */
146     read(N = Infinity) {
147         N = Math.min(N, this.buffer.length);
148         return this.buffer.splice(0, N);
149     }
150
151     /**
152      * Writes data to the TCP connection if the client is active.
153      * @param {Buffer|string} data - The data to send over the TCP connection.
154      */
155     write(data) {
156         if(this.active) {
157             this.com.write(data);
158         }
159     }
160
161     /**
162      * Returns the number of bytes available in the buffer.
163      * @returns {number} The number of available bytes.
164      */
165     availableBytes() {
166         return this.buffer.length;
167     }
168
169     /**
170      * Closes the TCP connection and cleans up resources.
171      */
172     close() {

```

```
172     this.active = false;
173     if(this.com) {
174         this.com.destroy();
175     }
176 }
177 }
178
179 exports.PRDC_JSLAB_TCP_CLIENT = PRDC_JSLAB_TCP_CLIENT;
180
181 /**
182 * Class representing a TCP server for handling network communications.
183 */
184 class PRDC_JSLAB_TCP_SERVER {
185
186 /**
187 * Creates an instance of the TCP server.
188 * @param {Object} jsl - Reference to the main JSLAB object.
189 * @param {string} host - The host IP address or hostname to connect to.
190 * @param {number} port - The port number to connect to.
191 * @param {Function} onConnectCallback - Callback executed upon successful
192   connection.
193 */
194 constructor(jsl, host, port, onConnectCallback) {
195     const obj = this;
196     this.jsl = jsl;
197     this.host = host;
198     this.port = port;
199     this.sockets = {};
200     this.sid = 0;
201
202     this.onConnectCallback = onConnectCallback;
203     this.onDataCallback = null;
204     this.onErrorCallback = null;
205     this.onDisconnectCallback = null;
206
207     this.buffer = [];
208     this.active = false;
209
210     this._data_callback = false;
211
212     this.server = this.jsl.env.net.createServer(function(socket) {
213         obj._onConnect(socket);
214
215         socket.on('data', function(data) {
216             obj._onData(socket, data);
217         });
218
219         socket.on('end', function() {
220             obj._onDisconnect(socket);
221         });
222
223         socket.on('error', function(err) {
224             obj._onError(socket, err);
225         });
226     });
227 }
```

```

226
227     this.server.listen(this.port, this.host, function() {
228         obj.active = true;
229     });
230
231     this.jsl.addForCleanup(this, function() {
232         obj.close();
233     });
234 }
235
236 /**
237 * Handles new client connections.
238 * @param {Object} socket - The connected socket.
239 */
240 _onConnect(socket) {
241     this.sid += 1;
242     this.sockets[this.sid] = socket;
243     socket.sid = this.sid;
244
245     if(this.jsl.format.isFunction(this.onConnectCallback)) {
246         this.onConnectCallback(socket);
247     }
248 }
249
250 /**
251 * Handles socket errors.
252 * @param {Object} socket - The socket that encountered an error.
253 * @param {Error} err - The error object.
254 */
255 _onError(socket, err) {
256     if(this.jsl.format.isFunction(this.onErrorCallback)) {
257         this.onErrorCallback(socket, err);
258     }
259 }
260
261 /**
262 * Handles incoming data from sockets.
263 * @param {Object} socket - The socket that received data.
264 * @param {Buffer|string} data - The received data.
265 */
266 _onData(socket, data) {
267     if(this.jsl.format.isFunction(this.onDataCallback)) {
268         this.onDataCallback(socket, data);
269     }
270 }
271
272 /**
273 * Handles socket disconnections.
274 * @param {Object} socket - The socket that disconnected.
275 */
276 _onDisconnect(socket) {
277     if(this.jsl.format.isFunction(this.onDisconnectCallback)) {
278         this.onDisconnectCallback(socket);
279     }
280 }
```

```
281
282  /**
283   * Sets the callback function to handle incoming data events.
284   * @param {Function} callback - Function called when data is received.
285   */
286 setOnData(callback) {
287   if(this.jsl.format.isFunction(callback)) {
288     this.onDataCallback = callback;
289   }
290 }
291
292 /**
293  * Sets the callback function to handle error events.
294  * @param {Function} callback - Function called when an error occurs.
295  */
296 setError(callback) {
297   if(this.jsl.format.isFunction(callback)) {
298     this.onErrorCallback = callback;
299   }
300 }
301
302 /**
303  * Sets the callback function to handle disconnection events.
304  * @param {Function} callback - Function called when a socket disconnects.
305  */
306 setOnDisconnect(callback) {
307   if(this.jsl.format.isFunction(callback)) {
308     this.onDisconnectCallback = callback;
309   }
310 }
311
312 /**
313  * Writes data to a specific TCP connection.
314  * @param {Object} socket - The socket to write data to.
315  * @param {Buffer|string} data - The data to send over the TCP connection.
316  */
317 write(socket, data) {
318   if(this.active && this.sockets[socket.sid]) {
319     this.sockets[socket.sid].write(data);
320   }
321 }
322
323 /**
324  * Closes the TCP server and all active connections.
325  */
326 close() {
327   this.active = false;
328   if(this.server) {
329     this.server.close();
330   }
331   for(const sid in this.sockets) {
332     this.sockets[sid].destroy();
333   }
334   this.sockets = {};
335 }
```

```
336 }
337
338 exports.PRDC_JSLAB_TCP_SERVER = PRDC_JSLAB_TCP_SERVER;
```

Listing 114 - networking-tcp.js

```
1 /**
2  * @file JSLAB library networking UDP submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class representing a UDP client for handling network communications.
10 */
11 class PRDC_JSLAB_UDP {
12
13 /**
14  * Creates an instance of the UDP client.
15  * @param {Object} jsl Reference to the main JSLAB object.
16  * @param {string} host - The host IP address or hostname to connect to.
17  * @param {number} port - The port number to connect to.
18 */
19 constructor(jsl, host, port) {
20     var obj = this;
21     this.jsl = jsl;
22     this.host = host;
23     this.port = port;
24
25     this.active = false;
26
27     this.com = this.jsl.env.udp.createSocket('udp4');
28     this.com.connect(port, host, function(err) {
29         if(err) {
30             obj._onError(err);
31         } else {
32             obj._onConnect();
33         }
34     });
35 }
36
37 /**
38  * Handles successful connection establishment by setting the client's
39  * active status to true.
40 */
41 _onConnect() {
42     this.active = true;
43 }
44
45 /**
46  * Handles errors by setting the client's active status to false and
47  * possibly logging the error.
48  * @param {Error} err - The error object that was thrown.
49 */
50 _onError() {
```

```
49     this.active = false;
50 }
51
52 /**
53 * Sends data over the UDP connection if the client is active.
54 * @param {Buffer|string} data - The data to send over the UDP connection.
55 */
56 write(data) {
57     if(this.active) {
58         this.com.write(data);
59     }
60 }
61
62 /**
63 * Closes the UDP connection and cleans up resources.
64 */
65 close() {
66     var obj = this;
67     this.active = false;
68     this.com.close(function() {
69         delete obj.com;
70     });
71 }
72 }
73
74 exports.PRDC_JSLAB_UDP = PRDC_JSLAB_UDP;
75
76 /**
77 * Represents a UDP server that listens on a specific port.
78 */
79 class PRDC_JSLAB_UDP_SERVER {
80
81 /**
82 * Initializes a UDP server that binds to the specified port and listens for
83 * incoming messages.
84 * @param {Object} jsl Reference to the main JSLAB object.
85 * @param {number} port - The port number on which the server will listen
86 * for incoming UDP packets.
87 */
88 constructor(jsl, port) {
89     var obj = this;
90     this.jsl = jsl;
91     this.port = port;
92
93     this.buffer = [];
94
95     this._data_callback = false;
96
97     this.com = this.jsl.env.udp.createSocket('udp4');
98
99     this.com.on('message', function(msg) {
100         obj._onData(msg);
101     });
102
103     this.com.bind(port);
```

```
102      this.jsl.addForCleanup(this, function() {
103        obj.close();
104      });
105    }
106  }
107
108 /**
109 * Called when data is received. Buffers the incoming data for later
110 * retrieval.
111 * @param {Buffer} data - The received data buffer.
112 */
113 onData(data) {
114   if(this._data_callback) {
115     this.onDataCallback(data);
116   } else {
117     this.buffer.push(...data);
118   }
119 }
120 /**
121 * Sets the callback function to handle incoming data events.
122 * @param {Function} callback - The function to be called when data is
123 * received.
124 */
125 setOnData(callback) {
126   if(this.jsl.format.isFunction(callback)) {
127     this.buffer = [];
128     this.onDataCallback = callback;
129     this._data_callback = true;
130   }
131 }
132 /**
133 * Reads a specified number of bytes from the buffer.
134 * @param {number} [N=Infinity] - The maximum number of bytes to read. Reads
135 *       all available bytes by default.
136 * @returns {Array} An array containing the first N bytes of buffered data.
137 */
138 read(N = Infinity) {
139   N = Math.min(N, this.buffer.length);
140   return this.buffer.splice(0, N);
141 }
142 /**
143 * Returns the number of bytes available in the buffer.
144 * @returns {number} The number of bytes currently stored in the buffer.
145 */
146 availableBytes() {
147   return this.buffer.length;
148 }
149
150 /**
151 * Closes the UDP server and releases any resources.
152 */
153 close() {
```

```

154     var obj = this;
155     this.com.close(function() {
156         delete obj.com;
157     });
158 }
159 }
160
161 exports.PRDC_JSLAB_UDP_SERVER = PRDC_JSLAB_UDP_SERVER;

```

Listing 115 - networking-udp.js

```

1 /**
2  * @file JSLAB library networking video call submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class representing a video call.
10 */
11 class PRDC_JSLAB_VIDEOCALL {
12
13 /**
14  * Creates a new video call instance.
15  * @param {string} type - The call type ('server' or 'client').
16  * @param {string} video_source_type - The video source type ('webcam' or 'desktop').
17  * @param {string} video_id - The video device or source ID.
18  * @param {string} mic_id - The microphone device ID.
19  * @param {string} tcp_host - The TCP host address.
20  * @param {number} tcp_port - The TCP port number.
21  * @param {object} opts - Additional configuration options.
22 */
23 constructor(jsl, type, video_source_type, video_id, mic_id, tcp_host,
24             tcp_port, opts) {
25     var obj = this;
26     this.jsl = jsl;
27     this.type = type;
28     this.video_source_type = video_source_type;
29     this.video_id = video_id;
30     this.mic_id = mic_id;
31     this.host = tcp_host;
32     this.port = tcp_port;
33     this.opts = opts || {};
34     this.timeout = this.opts.timeout || 15000;
35
36     this.is_initiator = (this.type === 'server');
37     this.peer_connection;
38     this.local_stream;
39     this.remote_stream;
40     this.incoming_buffer = '';
41
42     this.jsl.addForCleanup(this, function() {
43         obj.endCall();
44     });
45 }

```

```

44
45     this._init();
46 }
47
48 /**
49 * Attempt to connect the client socket
50 */
51 _connectClient() {
52     var obj = this;
53     this.tcp_client = this.jsl.networking.tcp(this.host, this.port, function()
54     {
55         obj._setUpClientHandlers();
56     });
57
58 /**
59 * Start a timeout waiting for signaling messages.
60 */
61 _startConnectionTimeout() {
62     var obj = this;
63
64     clearTimeoutIf(this.connection_timeout);
65     this.connection_timeout = setTimeout(function() {
66         obj._reconnectClient();
67     }, this.timeout);
68 }
69
70 /**
71 * Attempt to reconnect the client by closing the current connection and
72 * starting a new one.
73 */
74 _reconnectClient() {
75     if(this.tcp_client) {
76         try {
77             this.tcp_client.close();
78         } catch {}}
79     this._connectClient();
80 }
81
82 /**
83 * Initializes the video call by opening a window, setting up the UI,
84 * obtaining media, and creating the peer connection.
85 */
86 async _init() {
87     var obj = this;
88
89     this.win = await openWindowBlank();
90     this.win.setTitle('Video call');
91     this.win.document.body.innerHTML += '<video id="video"></video>';
92     this.dom = this.win.document.getElementById('video');
93     Object.assign(this.dom.style, {
94         position: 'absolute',
95         top: '50%',
```

```

96      transform: 'translate(-50%, -50%)',
97      width: '100%',
98      height: '100%',
99      objectFit: 'contain',
100     background: 'url("../img/no-video.svg") center / 30% no-repeat',
101   });
102   await this._getLocalMedia();
103   this._createPeerConnection();
104
105  if(this.is_initiator) {
106    this.tcp_server = this.jsl.networking.tcpServer(this.host, this.port,
107      function(socket) {
108        obj._setupServerSocketHandlers(socket);
109      });
110    } else {
111      this._connectClient();
112    }
113  }
114 /**
115 * Sets up handlers for the server-side socket once a client connects.
116 * @param {Object} socket - The connected socket.
117 */
118 _setupServerSocketHandlers(socket) {
119   var obj = this;
120   this.server_socket = socket;
121   this.tcp_server.setOnData(function(socket, data) {
122     if(socket.sid == obj.server_socket.sid) {
123       obj._processIncomingData(data);
124     }
125   });
126 }
127 /**
128 * Set up handlers for the client side once connected to server.
129 */
130 _setupClientHandlers() {
131   var obj = this;
132   this.tcp_client.setOnData(function(data) {
133     obj._processIncomingData(data);
134   });
135
136   this._sendSignalingMessage({ type: 'request-offer' });
137   this._startConnectionTimeout();
138 }
139
140 /**
141 * Processes incoming data over TCP.
142 * @param {string} data - The incoming data as a string.
143 */
144 _processIncomingData(data) {
145   this.incoming_buffer += data.toString('utf8');
146   let lines = this.incoming_buffer.split('\0');
147   while(lines.length > 1) {
148     let line = lines.shift().trim();

```

```

150     if(line) {
151         try {
152             let msg = JSON.parse(line);
153             this._handleSignalingMessage(msg);
154         } catch {}}
155     }
156 }
157
158     this.incoming_buffer = lines[0];
159 }
160
161 /**
162 * Sends a signaling message as JSON via TCP.
163 * @param {Object} msg - The signaling message to send.
164 */
165 _sendSignalingMessage(msg) {
166     const json_str = JSON.stringify(msg) + '\0';
167     if(this.is_initiator) {
168         if(this.server_socket) {
169             this.tcp_server.write(this.server_socket, json_str);
170         }
171     } else {
172         this.tcp_client.write(json_str);
173     }
174 }
175
176 /**
177 * Creates the PeerConnection and sets up handlers.
178 */
179 _createPeerConnection() {
180     var obj = this;
181     this.peer_connection = new RTCPeerConnection({});
182
183     if(this.local_stream) {
184         this.local_stream.getTracks().forEach(function(track) {
185             obj.peer_connection.addTrack(track, obj.local_stream);
186         });
187     }
188
189     this.peer_connection.ontrack = function(e) {
190         obj.dom.srcObject = e.streams[0];
191         obj.dom.play();
192     };
193
194     this.peer_connection.onicecandidate = function(e) {
195         if(e.candidate) {
196             obj._sendSignalingMessage({ type: 'candidate', candidate: e.candidate });
197         }
198     };
199 }
200
201 /**
202 * Obtains the local media stream (video/audio) based on the provided
203 * options.

```

```

203  */
204  async _getLocalMedia() {
205    var video_opts = this.opts.video || {};
206    var mic_opts = this.opts.mic || {};
207
208    var constraints = {};
209
210    if (!this.video_source_type || !this.video_id) {
211      constraints.video = false;
212    } else if (this.video_source_type === 'webcam') {
213      constraints.video = {
214        deviceId: { exact: this.video_id },
215        ...video_opts
216      };
217    } else if (this.video_source_type === 'desktop') {
218      constraints.video = {
219        mandatory: {
220          chromeMediaSource: 'desktop',
221          chromeMediaSourceId: this.video_id,
222          ...video_opts
223        }
224      };
225    }
226
227    if (!this.mic_id) {
228      constraints.audio = false;
229    } else {
230      constraints.audio = {
231        deviceId: { exact: this.mic_id },
232        ...mic_opts
233      };
234    }
235
236    try {
237      if (constraints.audio || constraints.video) {
238        this.local_stream = await this.jsl.env.navigator.mediaDevices.
239          getUserMedia(constraints);
240      } else {
241        this.local_stream = new MediaStream();
242      }
243    } catch {
244      this.jsl.env.error(`@videocall: ${language.string(222)})`);
245    }
246
247 /**
248 * Starts the call by creating and sending an offer if the instance is the
249 * initiator.
250 */
251 async _startCall() {
252   if (!this.is_initiator) return;
253   const offer = await this.peer_connection.createOffer({offerToReceiveVideo:
254     true, offerToReceiveAudio: true});
255   await this.peer_connection.setLocalDescription(offer);
256   this._sendSignalingMessage({ type: 'offer', sdp: this.peer_connection.

```

```

255           localDescription });
256     }
257
258     /**
259      * Answers an incoming offer by creating and sending an answer if the
260      * instance is not the initiator.
261     */
262     async _answerCall() {
263       const answer = await this.peer_connection.createAnswer();
264       await this.peer_connection.setLocalDescription(answer);
265       this._sendSignalingMessage({ type: 'answer', sdp: this.peer_connection.
266         localDescription });
267     }
268
269     /**
270      * Handles incoming signaling messages based on their type.
271      * @param {Object} message - The signaling message received.
272      */
273     async _handleSignalingMessage(message) {
274       if(!this.is_initiator) {
275         this.connection_timeout = clearTimeoutIf(this.connection_timeout);
276       }
277
278       switch(message.type) {
279         case 'request-offer':
280           if(this.is_initiator) {
281             await this._startCall();
282           }
283           break;
284         case 'offer':
285           if(!this.is_initiator) {
286             await this.peer_connection.setRemoteDescription(message.sdp);
287             await this._answerCall();
288           }
289           break;
290         case 'answer':
291           if(this.is_initiator) {
292             await this.peer_connection.setRemoteDescription(message.sdp);
293           }
294           break;
295         case 'candidate':
296           this.peer_connection.addIceCandidate(message.candidate);
297           break;
298         case 'message':
299           if(typeof this._onMessage == 'function') {
300             this._onMessage(message.data);
301           } else {
302             disp(message.data);
303           }
304           break;
305         }
306       }
307
308     /**
309      * Sets a callback function to handle incoming messages.
310    
```

```

307   * @param {Function} callback - The function to call when a message is
308   * received.
309   */
310 setOnMessage(callback) {
311   if(typeof callback == 'function') {
312     this._onMessage = callback;
313   }
314 }
315 /**
316  * Sends a message to the connected peer.
317  * @param {any} data - The data to send.
318  */
319 sendMessage(data) {
320   this._sendSignalingMessage({ type: 'message', data: data });
321 }
322 /**
323  * Toggles the local audio track on or off.
324  * @param {boolean} mute - If true, mutes the audio; otherwise, unmutes.
325  */
326 toggleAudio(mute) {
327   if(this.local_stream) {
328     this.local_stream.getAudioTracks().forEach((track) => track.enabled = !mute);
329   }
330 }
331 /**
332  * Toggles the local video track on or off.
333  * @param {boolean} disable - If true, disables the video; otherwise,
334  * enables it.
335  */
336 toggleVideo(disable) {
337   if(this.local_stream) {
338     this.local_stream.getVideoTracks().forEach((track) => track.enabled = !disable);
339   }
340 }
341 /**
342  * Ends the call by closing peer connections and media streams.
343  */
344 endCall() {
345   if(!this.is_initiator) {
346     this.connection_timeout = clearTimeoutIf(this.connection_timeout);
347   }
348   if(this.peer_connection) {
349     this.peer_connection.close();
350   }
351   if(this.local_stream) {
352     this.local_stream.getTracks().forEach((track) => track.stop());
353   }
354   if(this.remote_stream) {
355 
```

```

358     this.remote_stream.getTracks().forEach((track) => track.stop());
359   }
360
361   if(this.tcp_server) {
362     this.tcp_server.close();
363   }
364   if(this.tcp_client) {
365     this.tcp_client.close();
366   }
367 }
368 }
369
370 exports.PRDC_JSLAB_VIDEOCALL = PRDC_JSLAB_VIDEOCALL;

```

Listing 116 - networking-videoocall.js

```

1  /**
2  * @file JSLAB library networking submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 var { PRDC_JSLAB_TCP_CLIENT, PRDC_JSLAB_TCP_SERVER } = require('./networking-
  tcp');
9 var { PRDC_JSLAB_UDP, PRDC_JSLAB_UDP_SERVER } = require('./networking-udp');
10 var { PRDC_JSLAB_VIDEOCALL } = require('./networking-videoocall');
11
12 /**
13 * Class for JSLAB networking submodule.
14 */
15 class PRDC_JSLAB_LIB_NETWORKING {
16
17 /**
18 * Create submodule object.
19 * @param {Object} js1 Reference to the main JSLAB object.
20 */
21 constructor(js1) {
22   var obj = this;
23   this.js1 = js1;
24
25 /**
26 * XMODEM CRC table
27 * @type {Array}
28 */
29 this.CRC_TABLE_XMODEM = [
30   0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
31   0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
32   0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
33   0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
34   0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
35   0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
36   0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
37   0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
38   0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
39   0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,

```

```

40      0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0xa71, 0xa50, 0x3a33, 0x2a12,
41      0xdbfd, 0xcbdc, 0xfbff, 0xeb9e, 0x9b79, 0xb58, 0xbb3b, 0xab1a,
42      0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0xc60, 0x1c41,
43      0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
44      0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
45      0xff9f, 0xefbe, 0xdfdd, 0cffc, 0xbfb1, 0xaf3a, 0x9f59, 0x8f78,
46      0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
47      0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
48      0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
49      0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
50      0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
51      0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
52      0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
53      0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
54      0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
55      0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
56      0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
57      0x4a75, 0x5a54, 0x6a37, 0x7a16, 0xaaf1, 0x1ad0, 0x2ab3, 0x3a92,
58      0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
59      0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
60      0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
61      0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0,
62      ];
63  }
64
65  /**
66   * Checks if the environment is currently online.
67   * @returns {boolean} 'true' if online, otherwise 'false'.
68   */
69  isOnline() {
70    return this.jsl.env.online;
71  }
72
73  /**
74   * Calculates the CRC-16/XMODEM checksum of a byte array.
75   * @param {Uint8Array} byte_array - The input data as a byte array.
76   * @returns {number} The CRC-16/XMODEM checksum as a numeric value.
77   */
78  crc16xmodem(byte_array) {
79    let crc = 0x0000;
80    for(let i = 0; i < byte_array.length; i++) {
81      crc = (crc << 8) ^ this.CRC_TABLE_XMODEM[((crc >> 8) ^ byte_array[i]) &
82          0xFF];
83    }
84    return crc & 0xFFFF;
85  }
86
87  /**
88   * Retrieves the primary IPv4 address of the current machine.
89   * @returns {string} The IP address if found, otherwise an empty string.
90   */
91  getIP() {
92    return Object.values(this.jsl.env.os.networkInterfaces())
93      .reduce(function(r, list) { return r.concat(
94        list.reduce(function(rr, i) { return rr.concat(

```

```

94     i.family === 'IPv4' && !i.internal && i.address || []); }, []));}, []
95   }
96
97  /**
98   * Attempts to establish a TCP connection to the specified host and port to
99   * check reachability.
100  * @param {string} host - The IP address or hostname to ping.
101  * @param {number} port - The port number to use for the connection.
102  * @param {number} [timeout=1000] - The timeout in milliseconds before the
103   * attempt is considered failed.
104  * @returns {Promise<boolean>} A promise that resolves to 'true' if the
105   * connection is successful, 'false' otherwise.
106  */
107 async pingAddressTCP(host, port, timeout = 1000) {
108   var obj = this;
109   return new Promise(function(resolve) {
110     const socket = obj.jsl.env.net.createConnection(port, host);
111     socket.setTimeout(timeout);
112     socket.on('connect', function() {
113       socket.end();
114       resolve(true);
115     });
116     socket.on('timeout', function() {
117       socket.destroy();
118       resolve(false);
119     });
120     socket.on('error', function() {
121       socket.destroy();
122       resolve(false);
123     });
124   })
125   /**
126    * Executes a ping command to check if an IP address is reachable.
127    * @param {string} host - The IP address or hostname to ping.
128    * @param {number} timeout - The timeout in milliseconds for the ping
129    * command.
130    */
131   async pingAddress(host, timeout) {
132     return new Promise((resolve) => {
133       exec(`ping -n 1 -w ${timeout} ${host}`, function(error, stdout, stderr) {
134         if(error || stderr) {
135           resolve(false);
136         } else {
137           resolve(stdout.includes('Reply from'));
138         }
139       });
140     })
141   /**
142    * Executes a ping command synchronized to check if an IP address is
143    * reachable.

```

```

143  * @param {string} host - The IP address or hostname to ping.
144  * @param {number} timeout - The timeout in milliseconds for the ping
145  * command.
146  */
147 pingAddressSync(host, timeout) {
148   try {
149     var stdout = execSync(`ping -n 1 -w ${timeout} ${host}`);
150     return stdout.includes('Reply from');
151   } catch {
152     return false;
153   }
154 }
155 /**
156  * Finds the first unused port within a specified range, checking
157  * sequentially from 'port' to 'max_port'.
158  * @param {number} port - The starting port number to check.
159  * @param {number} min_port - The minimum port number in the range.
160  * @param {number} max_port - The maximum port number in the range.
161  */
162 async findFirstUnusedPort(port, min_port, max_port) {
163   let currentPort = port;
164
165   while(true) {
166     const inUse = await this.jsl.env.tcpPortUsed.check(port);
167     if(!inUse) {
168       return currentPort;
169     }
170     currentPort++;
171     if(currentPort > max_port) {
172       currentPort = min_port;
173     }
174   }
175 }
176 /**
177  * Converts an IPv4 address to its decimal equivalent.
178  * @param {string} ip - The IPv4 address in dot-decimal notation.
179  * @param {number} [subnets=4] - The number of subnets in the IP address,
180  *   default is 4.
181  * @returns {number} The decimal equivalent of the IPv4 address.
182  */
183 ip2dec(ip, subnets = 4) {
184   const octets = ip.split('.').map(Number).reverse();
185   let value = 0;
186   for(let i = 0; i < subnets; i++) {
187     value += octets[i]*Math.pow(256, i);
188   }
189   return value;
190 }
191 /**
192  * Creates a TCP client for communication with a specified host and port.
193  * @param {string} host - The hostname or IP address to connect to.

```

```

195  * @param {number} port - The port number on the host to connect to.
196  * @param {function} onConnectCallback - A callback function that is called
197  * when the connection is successfully established.
198  * @returns {PRDC_JSLAB_TCP_CLIENT} An instance of the TCP client with event
199  * handlers set up.
200  */
201  tcp(host, port, onConnectCallback) {
202    return new PRDC_JSLAB_TCP_CLIENT(this.jsl, host, port, onConnectCallback);
203  }
204
205  /**
206   * Creates a TCP server to listen on a specified port.
207   * @param {string} host - The hostname or IP address.
208   * @param {number} port - The port number.
209   * @param {function} onConnectCallback - A callback function that is called
210   * when the connection is successfully established.
211   * @returns {PRDC_JSLAB_TCP_SERVER} An instance of the TCP server.
212  */
213  tcpServer(...args) {
214    return new PRDC_JSLAB_TCP_SERVER(this.jsl, ...args);
215  }
216
217  /**
218   * Creates a UDP client for sending data to a specified host and port.
219   * @param {string} host - The hostname or IP address to connect to.
220   * @param {number} port - The port number to connect to.
221   * @returns {PRDC_JSLAB_UDP} An instance of the UDP client.
222  */
223  udp(host, port) {
224    return new PRDC_JSLAB_UDP(this.jsl, host, port);
225  }
226
227  /**
228   * Creates a UDP server to listen on a specified port.
229   * @param {number} port - The port number to listen on.
230   * @returns {PRDC_JSLAB_UDP_SERVER} An instance of the UDP server.
231  */
232  udpServer(port) {
233    return new PRDC_JSLAB_UDP_SERVER(this.jsl, port);
234  }
235
236  /**
237   * Initializes and returns a new video call instance.
238   * @param {string} type - The call type ('server' or 'client').
239   * @param {string} video_source_type - The video source type ('webcam' or
240   * 'desktop').
241   * @param {string} video_id - The video device or source ID.
242   * @param {string} mic_id - The microphone device ID.
243   * @param {string} tcp_host - The TCP host address.
244   * @param {number} tcp_port - The TCP port number.
245   * @param {object} opts - Additional configuration options.
246   * @returns {PRDC_JSLAB_VIDEOCALL} The created video call object.
247  */
248  videoCall(type, video_source_type, video_id, mic_id, tcp_host, tcp_port,
249            opts) {

```

```

245     return new PRDC_JSLAB_VIDEOCALL(this.jsl, type, video_source_type,
246         video_id, mic_id, tcp_host, tcp_port, opts);
247   }
248 }
249
250 exports.PRDC_JSLAB_LIB_NETWORKING = PRDC_JSLAB_LIB_NETWORKING;

```

Listing 117 - networking.js

```

1  /**
2  * @file JSLAB library non blocking functions submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9 * Class for JSLAB non blocking functions submodule.
10 */
11 class PRDC_JSLAB_LIB_NON_BLOCKING {
12
13 /**
14 * Initializes a new instance of the non-blocking functions submodule.
15 * @param {Object} jsl Reference to the main JSLAB object.
16 */
17 constructor(jsl) {
18   var obj = this;
19   this.jsl = jsl;
20 }
21
22 /**
23 * Executes a given function in a non-blocking while loop.
24 * @param {Function} fn A function that returns a boolean value; when false ,
25 *   the loop exits.
26 */
27 nbwhile(fn) {
28   var obj = this;
29   function fnw() {
30     if (!obj.jsl.basic.checkStopLoop()) {
31       if (!fn()) {
32         obj.jsl.env.setImmediate(fnw);
33       }
34     } else {
35       obj.jsl.env.error('@nbwhile: '+language.string(125));
36     }
37   }
38   this.jsl.env.setImmediate(fnw);
39 }
40
41 /**
42 * Executes a given function once in a non-blocking manner.
43 * @param {Function} fn The function to be executed.
44 */
45 nbrun(fn) {
46   if (!this.jsl.basic.checkStopLoop()) {
47     this.jsl.env.setImmediate(fn);

```

```

47 } else {
48     this.jsl.env.error('@nbrun: '+language.string(125));
49 }
50 }

52 /**
53 * Schedules the next block of code to be executed in a non-blocking manner.
54 * @param {Function} fn The function to execute next.
55 */
56 nbnext(fn) {
57     var obj = this;
58     function fnw() {
59         if(!obj.jsl.basic.checkStopLoop()) {
60             fn();
61         } else {
62             obj.jsl.env.error('@nbnext: '+language.string(125));
63         }
64     }
65     this.jsl.env.setImmediate(fnw);
66 }

68 /**
69 * Waits for a specified number of milliseconds in a non-blocking manner.
70 * @param {Number} ms The number of milliseconds to wait.
71 * @returns {Promise<void>} A promise that resolves after the specified time
72 * has elapsed.
73 */
74 waitMSeconds(ms) {
75     if(!this.jsl.basic.checkStopLoop()) {
76         return new Promise(function(resolve, reject) { setTimeout(resolve, ms)
77             });
78     } else {
79         this.jsl.env.error('@waitMSeconds: '+language.string(125), true);
80     }
81     return false;
82 }

83 /**
84 * Waits for a specified number of seconds in a non-blocking manner.
85 * @param {Number} s The number of seconds to wait.
86 * @returns {Promise<void>} A promise that resolves after the specified time
87 * has elapsed.
88 */
89 waitSeconds(s) {
90     if(!this.jsl.basic.checkStopLoop()) {
91         return waitMSeconds(s*1000);
92     } else {
93         this.jsl.env.error('@waitSeconds: '+language.string(125), true);
94     }
95     return false;
96 }

97 /**
98 * Waits for a specified number of minutes in a non-blocking manner.
99 * @param {Number} min The number of minutes to wait.

```

```

99     * @returns {Promise<void>} A promise that resolves after the specified time
100    * has elapsed.
101   */
102  waitMinutes(min) {
103    if(!this.jsl.basic.checkStopLoop()) {
104      return waitMSeconds(min*60*1000);
105    } else {
106      this.jsl.env.error('@waitMinutes: '+language.string(125), true);
107    }
108    return false;
109  }
110 /**
111  * Clears the specified interval if it exists.
112  * @param {number|undefined} timeout - The interval ID to be cleared.
113  * @returns {boolean} Always returns false.
114  */
115 clearIntervalIf(timeout) {
116  if(timeout) {
117    clearInterval(timeout);
118  }
119  return false;
120}
121 /**
122  * Clears the specified timeout if it exists.
123  * @param {number|undefined} timeout - The timeout ID to be cleared.
124  * @returns {boolean} Always returns false.
125  */
126 clearTimeoutIf(timeout) {
127  if(timeout) {
128    clearTimeout(timeout);
129  }
130  return false;
131}
132 /**
133  * Initializes a new worker with the specified module path.
134  * @param {string} path - The path to the module to configure the worker.
135  * @returns {Worker} The initialized Worker instance.
136  */
137 initWorker(path) {
138  var worker = new Worker(app_path + "/js/sandbox/init-worker.js");
139  worker.postMessage({
140    type: 'configureWorker',
141    module_path: path
142  });
143  return worker;
144}
145}
146}
147}
148
149 exports.PRDC_JSLAB_LIB_NON_BLOCKING = PRDC_JSLAB_LIB_NON_BLOCKING;

```

Listing 118 - non-blocking.js

```

2  * @file JSLAB_OpenModelicaLink submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 const zmq = require('zeromq');
9 const fs = require('fs');
10 const path = require('path');
11 const os = require('os');
12 const { exec, execSync } = require('child_process');
13 const { XMLParser } = require('fast-xml-parser');
14
15 /**
16  * Class for JSLAB_OpenModelicaLink.
17  */
18 class PRDC_JSLAB_OPENMODELICA_LINK {
19
20 /**
21  * Initializes a new instance of the OpenModelicaLink.
22  * @param {Object} js1 Reference to the main JSLAB object.
23  */
24 constructor(js1) {
25   this.js1 = js1;
26 }
27
28 /**
29  * Starts the interaction with an external executable by initializing the
30  * necessary environment and parameters.
31  * Launches the executable with the appropriate command-line arguments for
32  * interaction via ZMQ.
33  * Waits for a port file to be created to establish the ZMQ communication.
34  * @param {string} exe - Path to the executable to be run, defaults to the
35  * OpenModelica compiler if not provided.
36 /**
37 async start(exe) {
38   var obj = this;
39   this.exe = exe;
40
41   this.pid = 0;
42   this.active = false;
43   this.requester = null;
44   this.portfile = '';
45
46   this.filename = '';
47   this.modelname = '';
48   this.xmlfile = '';
49   this.resultfile = '';
50   this.csvfile = '';
51   this.mat_temp_dir = '';
52   this.simulation_options = {};
53   this.quantities_list = [];
54   this.parameter_list = {};
55   this.continuous_list = {};
56   this.input_list = {};

```

```

54   this.output_list = {};
55   this.mapped_names = {};
56   this.override_variables = {};
57   this.sim_opt_override = {};
58   this.input_flag = false;
59   this.linear_options = {
60     startTime: '0.0',
61     stopTime: '1.0',
62     number_of_intervals: '500',
63     stepSize: '0.002',
64     tolerance: '1e-6'
65   };
66   this.linearfile = '';
67   this.linear_flag = false;
68   this.linear_modelname = '';
69   this.linear_inputs = '';
70   this.linear_outputs = '';
71   this.linear_states = '';
72   this.linear_quantity_list = [];
73
74   const random_string = Math.random().toString(36).substring(7);
75   let cmd, portfile;
76
77   if(process.platform === 'win32') {
78     exe = exe || path.join(process.env.OPENMODELICAHOME, 'bin', 'omc.exe');
79     cmd = 'START /b "" "${exe}" --interactive= zmq +z=jslab.${random_string}
80           }';
81     portfile = path.join(os.tmpdir(), 'openmodelica.port.jslab.${{
82       random_string}}');
83   } else {
84     exe = exe || 'omc';
85     cmd = '${exe} --interactive= zmq -z=jslab.${random_string} &';
86     portfile = path.join(os.tmpdir(), 'openmodelica.${process.env.USER}.port
87           .jslab.${random_string}');
88   }
89
90   this.portfile = portfile;
91
92   var [flag1, pids1] = isProgramRunning('omc.exe');
93   var omc_process = exec(cmd);
94   await waitMSeconds(200);
95   var [flag2, pids2] = isProgramRunning('omc.exe');
96   this.pid = pids2.filter(function(e) {
97     return !pids1.includes(e)
98   });
99
100  while(true) {
101    await waitMSeconds(10);
102    if(fs.existsSync(this.portfile)) {
103      const filedata = fs.readFileSync(this.portfile, 'utf-8');
104      this.requester = new zmq.Request();
105      this.requester.connect(filedata);
106      this.active = true;
107      break;
108    }
109  }

```

```

106     }
107   }
108
109  /**
110   * Sends an expression to be evaluated by the external executable through
111   * the ZMQ connection and waits for the result.
112   * Parses the response using a dedicated expression parser.
113   * @param {string} expr - The expression to be evaluated.
114   * @returns {Promise<any>} - A promise that resolves with the parsed result
115   * of the expression evaluation.
116   * @throws {Error} - Throws an error if there is no active connection.
117   */
118   async sendExpression(expr) {
119     if(this.active) {
120       await this.requester.send(expr);
121       const [result] = await this.requester.receive();
122       return this.parseExpression(result.toString());
123     } else {
124       this.jsl.env.error(`@sendExpression: ${language.string(201)})`);
125       return false;
126     }
127   }
128 /**
129  * Initializes and configures a Modelica system with the specified
130  * parameters and libraries.
131  * Loads necessary files and prepares the environment for simulation.
132  * @param {string} filename - The path to the Modelica file.
133  * @param {string} modelname - The name of the Modelica model.
134  * @param {string[]} [libraries=[]] - An array of library paths to load.
135  * @param {string} [command_line_options=''] - Additional command-line
136  * options for the simulation.
137  */
138   async ModelicaSystem(filename, modelname, libraries = [], command_line_options = '') {
139     if(!filename || !modelname) {
140       this.jsl.env.error(`@ModelicaSystem: ${language.string(203)})`);
141       return false;
142     }
143     if(command_line_options) {
144       const cmd_exp = await this.sendExpression(`setcommand_line_options("${{
145       command_line_options}}")`);
146       if(cmd_exp === 'false') {
147         this.jsl.env.error(`@ModelicaSystem: ${await this.sendExpression(`getErrorResponse()`)}`);
148         return false;
149       }
150     }
151     const filepath = path.normalize(filename).replace(/\//g, '/');
152     const load_file_msg = await this.sendExpression(`loadFile("${filepath}")`);
153     if(load_file_msg === 'false') {
154       this.jsl.env.error(`@ModelicaSystem: ${await this.sendExpression(`getErrorResponse()`)}`);
155     }

```

```

      getErrorString()"));
153   return false;
154 }
155
156 for(const lib of libraries) {
157   let libmsg;
158   if(fs.existsSync(lib)) {
159     libmsg = await this.sendExpression(`loadFile("${lib}")`);
160   } else {
161     libmsg = await this.sendExpression(`loadModel(${lib})`);
162   }
163   if(libmsg === 'false') {
164     this.jsl.env.error(`@ModelicaSystem: ${await this.sendExpression(
165       getErrorString())}`);
166     return false;
167   }
168
169   this.filename = filename;
170   this.modelname = modelname;
171   this.mat_temp_dir = fs.mkdtempSync(path.join(os.tmpdir(), `

OpenModelicaLink-`));
172   await this.sendExpression(`cd("${this.mat_temp_dir}")`);
173   await this.BuildModelicaModel();
174   return true;
175 }

176 /**
177 * Builds the Modelica model by sending the appropriate build command and
178 * parsing the resulting XML file.
179 */
180 async BuildModelicaModel() {
181   const build_model_result = await this.sendExpression(`buildModel(${this.

modelname})`);
182   if(!build_model_result) {
183     this.jsl.env.error(`@BuildModelicaModel: ${await this.sendExpression(
184       getErrorString())}`);
185     return false;
186   }
187
188   this.xmlfile = path.join(this.mat_temp_dir, build_model_result[1]);
189   this.xmlparse();
190   return true;
191 }

192 /**
193 * Retrieves the working directory used for temporary files and simulations.
194 * @returns {string} - The path to the working directory.
195 */
196 getWorkDirectory() {
197   return this.mat_temp_dir;
198 }

199 /**
200 * Parses the XML file generated by the Modelica compiler to extract

```

```

    simulation parameters and variables.

202  */
203  xmlparse() {
204    if(fs.existsSync(this.xmlfile)) {
205      const xml_data = fs.readFileSync(this.xmlfile, 'utf-8');
206      const parser = new XMLParser({ ignoreAttributes: false,
207        attributeNamePrefix: '' });
208      const xDoc = parser.parse(xml_data);

209      // default_experiment
210      const default_experiment = xDoc.fmiModelDescription.DefaultExperiment;
211      if(default_experiment) {
212        this.simulation_options.startTime = default_experiment.startTime || '0.0';
213        this.simulation_options.stopTime = default_experiment.stopTime || '1.0';
214        this.simulation_options.stepSize = default_experiment.stepSize || '0.002';
215        this.simulation_options.tolerance = default_experiment.tolerance || '1e-6';
216        this.simulation_options.solver = default_experiment.solver || '';
217      }
218
219      // scalar_variables
220      const scalar_variables = xDoc.fmiModelDescription.ModelVariables.
221        ScalarVariable || [];
222      const fields = ['name', 'isValueChangeable', 'description', 'variability',
223        'causality', 'alias', 'aliasVariable'];
224      for(let k = 0; k < scalar_variables.length; k++) {
225        const item = scalar_variables[k];
226        let scalar = {};
227        fields.forEach(field => {
228          scalar[field] = item[field] || '';
229        });
230
231        if(item.Real) {
232          scalar.value = item.Real.start || '';
233        }
234
235        this.processVariable(scalar);
236      } else {
237        this.jsl.env.error('@xmlparse: '+language.string(204));
238        return false;
239      }
240      return true;
241    }
242
243    /**
244     * Processes a scalar variable from the XML file and categorizes it based on
245     * its properties.
246     * @param {Object} scalar - The scalar variable to process.
247     */
248    processVariable(scalar) {
249      const name = scalar.name;

```

```

248     const value = scalar.value;
249
250     if(!this.linear_flag) {
251         if(scalar.variability === 'parameter') {
252             this.parameter_list[name] = value;
253         } else if(scalar.variability === 'continuous') {
254             this.continuous_list[name] = value;
255         } else if(scalar.causality === 'input') {
256             this.input_list[name] = value;
257         } else if(scalar.causality === 'output') {
258             this.output_list[name] = value;
259         }
260     }
261
262     if(this.linear_flag) {
263         if(scalar.alias === 'alias') {
264             if(name.startsWith('x')) {
265                 this.linear_states.push(name.slice(3, -1));
266             } else if(name.startsWith('u')) {
267                 this.linear_inputs.push(name.slice(3, -1));
268             } else if(name.startsWith('y')) {
269                 this.linear_outputs.push(name.slice(3, -1));
270             }
271         }
272         this.linear_quantity_list.push(scalar);
273     } else {
274         this.quantities_list.push(scalar);
275     }
276 }
277
278 /**
279 * Retrieves a list of quantities based on the provided arguments.
280 * @param {string[]} [args] - An array of quantity names to retrieve. If
281 *   omitted, returns all quantities.
282 * @returns {Object[]} - An array of quantity objects.
283 */
284 getQuantities(args) {
285     if(args && args.length > 0) {
286         const tmpresult = [];
287         for(let n = 0; n < args.length; n++) {
288             for(let q = 0; q < this.quantities_list.length; q++) {
289                 if(this.quantities_list[q].name === args[n]) {
290                     tmpresult.push(this.quantities_list[q]);
291                 }
292             }
293         }
294         return tmpresult;
295     } else {
296         return this.quantities_list;
297     }
298 }
299
300 /**
301 * Retrieves a list of linearized quantities based on the provided arguments
302 *
303 */

```

```

301   * @param {string[]} [args] - An array of linear quantity names to retrieve.  

302     If omitted, returns all linear quantities.  

303   * @returns {Object[]} - An array of linear quantity objects.  

304   */  

305   getLinearQuantities(args) {  

306     if(args && args.length > 0) {  

307       const tmpresult = [];  

308       for(let n = 0; n < args.length; n++) {  

309         for(let q = 0; q < this.linear_quantity_list.length; q++) {  

310           if(this.linear_quantity_list[q].name === args[n]) {  

311             tmpresult.push(this.linear_quantity_list[q]);  

312           }  

313         }  

314       }  

315     } else {  

316       return this.linear_quantity_list;  

317     }  

318   }  

319  

320  /**
321   * Retrieves simulation parameters based on the provided arguments.  

322   * @param {string|string[]} [args] - A single parameter name or an array of  

323     parameter names to retrieve. If omitted, returns all parameters.  

324   * @returns {Object|any} - An object containing the requested parameters or  

325     a single parameter value.  

326   */  

327   getParameters(args) {  

328     if(args && args.length > 0) {  

329       if(Array.isArray(args)) {  

330         const param = {};  

331         for(let n = 0; n < args.length; n++) {  

332           param[args[n]] = this.parameter_list[args[n]];  

333         }  

334       } else {  

335         return this.parameter_list[args];  

336       }  

337     } else {  

338       return this.parameter_list;  

339     }  

340   }  

341  /**
342   * Retrieves input variables based on the provided arguments.  

343   * @param {string|string[]} [args] - A single input name or an array of  

344     input names to retrieve. If omitted, returns all inputs.  

345   * @returns {Object|any} - An object containing the requested inputs or a  

346     single input value.  

347   */  

348   getInputs(args) {  

349     if(args && args.length > 0) {  

350       if(Array.isArray(args)) {  

351         const inputs = {};  

352         for(let n = 0; n < args.length; n++) {  


```

```

351         inputs[args[n]] = this.input_list[args[n]];
352     }
353     return inputs;
354 } else {
355     return this.input_list[args];
356 }
357 } else {
358     return this.input_list;
359 }
360 }
361 /**
362 * Retrieves output variables based on the provided arguments.
363 * @param {string|string[]} [args] - A single output name or an array of
364 *         output names to retrieve. If omitted, returns all outputs.
365 * @returns {Object|any} - An object containing the requested outputs or a
366 *         single output value.
367 */
368 getOutputs(args) {
369     if(args && args.length > 0) {
370         if(Array.isArray(args)) {
371             const outputs = {};
372             for(let n = 0; n < args.length; n++) {
373                 outputs[args[n]] = this.output_list[args[n]];
374             }
375             return outputs;
376         } else {
377             return this.output_list[args];
378         }
379     } else {
380         return this.output_list;
381     }
382 }
383 /**
384 * Retrieves continuous variables based on the provided arguments.
385 * @param {string|string[]} [args] - A single continuous variable name or an
386 *         array of names to retrieve. If omitted, returns all continuous
387 *         variables.
388 * @returns {Object|any} - An object containing the requested continuous
389 *         variables or a single value.
390 */
391 getContinuous(args) {
392     if(args && args.length > 0) {
393         if(Array.isArray(args)) {
394             const continuous = {};
395             for(let n = 0; n < args.length; n++) {
396                 continuous[args[n]] = this.continuous_list[args[n]];
397             }
398             return continuous;
399         } else {
400             return this.continuous_list[args];
401         }
402     } else {
403         return this.continuous_list;
404     }
405 }
```

```
401     }
402 }
403
404 /**
405 * Retrieves simulation options based on the provided arguments.
406 * @param {string|string[]} [args] - A single simulation option name or an
407 * array of names to retrieve. If omitted, returns all simulation options.
408 * @returns {Object|any} - An object containing the requested simulation
409 * options or a single option value.
410 */
411 getSimulationOptions(args) {
412     if(args && args.length > 0) {
413         if(Array.isArray(args)) {
414             const simoptions = {};
415             for(let n = 0; n < args.length; n++) {
416                 simoptions[args[n]] = this.simulation_options[args[n]];
417             }
418             return simoptions;
419         } else {
420             return this.simulation_options[args];
421         }
422     } else {
423         return this.simulation_options;
424     }
425 }
426 /**
427 * Retrieves linearization options based on the provided arguments.
428 * @param {string|string[]} [args] - A single linearization option name or
429 * an array of names to retrieve. If omitted, returns all linearization
430 * options.
431 * @returns {Object|any} - An object containing the requested linearization
432 * options or a single option value.
433 */
434 getLinearizationOptions(args) {
435     if(args && args.length > 0) {
436         if(Array.isArray(args)) {
437             const linoptions = {};
438             for(let n = 0; n < args.length; n++) {
439                 linoptions[args[n]] = this.linear_options[args[n]];
440             }
441             return linoptions;
442         } else {
443             return this.linear_options[args];
444         }
445     } else {
446         return this.linear_options;
447     }
448 }
449 /**
450 * Sets simulation parameters based on the provided arguments.
451 * @param {string|string[]} args - A single parameter assignment (e.g., "param=5")
452 * or an array of assignments.
453 */
454
```

```

450  setParameters(args) {
451    if(args && args.length > 0) {
452      if(!Array.isArray(args)) {
453        args = [args];
454      }
455      args.forEach(arg => {
456        const val = arg.replace(/\s+/g, " ");
457        const [key, value] = val.split("=");
458        if(key in this.parameter_list) {
459          this.parameter_list[key] = value;
460          this.override_variables[key] = value;
461        } else {
462          this.jsl.env.error('@setParameters: ' + key + language.string(209));
463        }
464      });
465    }
466  }
467
468 /**
469 * Sets simulation options based on the provided arguments.
470 * @param {string|string[]} args - A single simulation option assignment (e.g., "stepSize=0.01") or an array of assignments.
471 */
472 setSimulationOptions(args) {
473   if(args && args.length > 0) {
474     if(!Array.isArray(args)) {
475       args = [args];
476     }
477     args.forEach(arg => {
478       const val = arg.replace(/\s+/g, " ");
479       const [key, value] = val.split("=");
480       if(key in this.simulation_options) {
481         this.simulation_options[key] = value;
482         this.sim_opt_override[key] = value;
483       } else {
484         this.jsl.env.error('@setSimulationOptions: ' + key + language.string(210));
485       }
486     });
487   }
488 }
489
490 /**
491 * Sets linearization options based on the provided arguments.
492 * @param {string|string[]} args - A single linearization option assignment or an array of assignments.
493 */
494 setLinearizationOptions(args) {
495   if(args && args.length > 0) {
496     if(!Array.isArray(args)) {
497       args = [args];
498     }
499     args.forEach(arg => {
500       const val = arg.replace(/\s+/g, " ");
501       const [key, value] = val.split("=");

```

```

502     if(key in this.linear_options) {
503         this.linear_options[key] = value;
504     } else {
505         this.jsl.env.error('@setLinearizationOptions: ' + key + language.
506                             string(211));
507     });
508 }
509 }
510
511 /**
512 * Sets input variables based on the provided arguments.
513 * @param {string|string[]} args - A single input assignment (e.g., "input1
514             =10") or an array of assignments.
515 */
516 setInputs(args) {
517     if(args && args.length > 0) {
518         if(!Array.isArray(args)) {
519             args = [args];
520         }
521         args.forEach(arg => {
522             const val = arg.replace(/\s+/g, '');
523             const [key, value] = val.split("=");
524             if(key in this.input_list) {
525                 this.input_list[key] = value;
526                 this.input_flag = true;
527             } else {
528                 this.jsl.env.error('@setInputs: ' + key + language.string(212));
529             }
530         });
531     }
532 }
533 /**
534 * Creates a CSV file containing input data for the simulation.
535 */
536 createcsvData() {
537     this.csvfile = path.join(this.mat_temp_dir, `${this.modelname}.csv`);
538     const file_id = fs.openSync(this.csvfile, 'w');
539     const fields = Object.keys(this.input_list);
540     let header = `time,${fields.join(",")},end\n`;
541     fs.writeSync(file_id, header);
542
543     let time = [];
544     let tmpcsvdata = {};
545
546     fields.forEach(field => {
547         let var_data = this.input_list[field] || "0";
548         try {
549             var_data = eval(var_data.replace(/\[\|\]/\(|\)|/g, match => {
550                 return match === '[' || match === ']' ? ',' : '{';
551             }));
552         } catch {
553             var_data = [[0, 0]]; // Default to 0 if evaluation fails
554         }

```

```

555     tmpcsvdata[field] = var_data;
556
557     if(var_data.length > 1) {
558         var_data.forEach(entry => {
559             time.push(entry[0]);
560         });
561     }
562 });
563
564 if(time.length === 0) {
565     time = [parseFloat(this.simulation_options.startTime), parseFloat(this.
566         simulation_options.stopTime)];
567 }
568
569 const sorted_time = time.sort((a, b) => a - b);
570 let previous_value = {};
571
572 sorted_time.forEach(t => {
573     let line = `${t},`;
574     fields.forEach((field) => {
575         let data = tmpcsvdata[field];
576         let value = previous_value[field] || 0;
577
578         if(Array.isArray(data)) {
579             for(let j = 0; j < data.length; j++) {
580                 if(data[j][0] === t) {
581                     value = data[j][1];
582                     data.splice(j, 1);
583                     tmpcsvdata[field] = data;
584                     break;
585                 }
586                 previous_value[field] = value;
587             }
588             line += `${value},`;
589         );
590         line += "\0\n";
591         fs.writeFileSync(file_id, line);
592     );
593
594     fs.closeSync(file_id);
595 }
596
597 /**
598 * Runs the simulation with optional result file and simulation flags.
599 * @param {string} [resultfile=''] - The name of the result file to generate
600 *
601 * @param {string} [sim_flags=''] - Additional simulation flags.
602 */
603 async simulate(resultfile = '', sim_flags = '') {
604     let r = resultfile ? ` -r=${resultfile}` : '';
605     this.resultfile = resultfile ? path.join(this.mat_temp_dir, resultfile) :
606         path.join(this.mat_temp_dir, `${this.modelname}_res.mat`);
607
608     if(fs.existsSync(this.xmlfile)) {

```

```

607   let getexefile;
608   if(process.platform === 'win32') {
609     getexefile = path.join(this.mat_temp_dir, `${this.modelname}.exe`);
610   } else {
611     getexefile = path.join(this.mat_temp_dir, this.modelname);
612   }
613
614   if(fs.existsSync(getexefile)) {
615     let overridevar = '';
616     if(Object.keys(this.override_variables).length || Object.keys(this.
617       sim_opt_override).length) {
618       const allOverrides = { ...this.override_variables, ...this.
619         sim_opt_override };
620       const fields = Object.keys(allOverrides);
621       const tmpoverride1 = fields.map(field => `${field}=${allOverrides[
622         field]}`);
623       overridevar = ` -override=${tmpoverride1.join(' ', '')}`;
624     }
625
626     let csvinput = '';
627     if(this.input_flag) {
628       this.createcsvData();
629       csvinput = ` -csvInput=${this.csvfile}`;
630     }
631
632     const final_simulation_exe = `"${getexefile}"${overridevar}${csvinput}
633       ${r}${sim_flags}`;
634     execSync(final_simulation_exe, { cwd: this.mat_temp_dir });
635   } else {
636     this.jsl.env.error('@simulate: '+language.string(205));
637   }
638
639 /**
640 * Performs linearization of the model and retrieves the linear matrices.
641 * @returns {Array<Object>} - An array containing the A, B, C, and D
642 *   matrices.
643 */
644 async linearize() {
645   const linres = await this.sendExpression("setcommand_line_options(\"+
646     generateSymbolicLinearization\")");
647   if(linres && linres[0] === "false") {
648     this.jsl.env.error('@simulate: '+language.string(207)+ await this.
649       sendExpression("getErrorString()"));
650     return false;
651   }
652
653   const fields = Object.keys(this.override_variables);
654   const tmpoverride1 = fields.map(field => `${field}=${this.
655     override_variables[field]}`);
656   const tmpoverride2 = tmpoverride1.length ? ` -override=${tmpoverride1.join
657     (' ', '')}` : "";

```

```

653
654   const lin_fields = Object.keys(this.linear_options);
655   const tmpoverride1lin = lin_fields.map(field => `${field}=${this.
656     linear_options[field]}`);
657   const overridelinear = tmpoverride1lin.join(',');
658
659   let csvinput = '';
660   if(this.input_flag) {
661     this.createcsvData();
662     csvinput = `--csvInput=${this.csvfile.replace(/\//g, '/')}`;
663   }
664
665   const linexpr = `linearize(${this.modelname},${overridelinear},sim_flags="`+
666     `${csvinput} ${tmpoverride2}`)`;
667   const res = await this.sendExpression(linexpr);
668   this.resultfile = res.resultFile;
669   this.linear_modelname = `linear_${this.modelname}`;
670   this.linearfile = path.join(this.mat_temp_dir, `${this.linear_modelname}.
671     mo`).replace(/\//g, '/');
672   if(fs.existsSync(this.linearfile)) {
673     const loadmsg = await this.sendExpression(`loadFile("${this.linearfile}"`);
674     if(loadmsg && loadmsg[0] === "false") {
675       this.jsl.env.error('@linearize: '+ await this.sendExpression("`+
676         getErrorString()`));
677       return false;
678     }
679
680     const cNames = await this.sendExpression("getclassNames()");
681     const buildmodeexpr = `buildModel(${cNames[0]})`;
682     const buildModelmsg = await this.sendExpression(buildmodeexpr);
683     if(buildModelmsg && buildModelmsg.length > 0) {
684       this.linear_flag = true;
685       this.xmlfile = path.join(this.mat_temp_dir, buildModelmsg[1]);
686       this.linear_quantity_list = [];
687       await this.xmlparse();
688       return this.getLinearMatrix();
689     } else {
690       this.jsl.env.error('@linearize: '+ await this.sendExpression("`+
691         getErrorString()`));
692       return false;
693     }
694   }
695   return true;
696 }
697 */
698 * Retrieves the linear A, B, C, and D matrices.
699 * @returns {Array<Object>} - An array containing the A, B, C, and D
700   matrices.
701 */
702 getLinearMatrix() {
703   const matrix_A = {};
704   const matrix_B = {};
705   const matrix_C = {};

```

```

701   const matrix_D = {};
702
703   this.linear_quantity_list.forEach(item => {
704     const name = item.name;
705     const value = item.value;
706
707     if(item.variability === "parameter") {
708       if(name.startsWith('A')) {
709         matrix_A[name] = value;
710       } else if(name.startsWith('B')) {
711         matrix_B[name] = value;
712       } else if(name.startsWith('C')) {
713         matrix_C[name] = value;
714       } else if(name.startsWith('D')) {
715         matrix_D[name] = value;
716       }
717     }
718   });
719
720   return [matrix_A, matrix_B, matrix_C, matrix_D];
721 }
722
723 /**
724 * Converts linear matrix data into a two-dimensional array format.
725 * @param {Object} matrix_name - The linear matrix object to convert.
726 * @returns {number[][]|number} - The converted matrix as a 2D array or 0 if
727   empty.
728 */
729 getLinearMatrixValues(matrix_name) {
730   if(Object.keys(matrix_name).length > 0) {
731     const fields = Object.keys(matrix_name);
732     const last_field = fields[fields.length - 1];
733     const rows = parseInt(last_field.charAt(2), 10);
734     const columns = parseInt(last_field.charAt(4), 10);
735     const tmp_matrix = Array.from({ length: rows }, () => Array(columns).
736       fill(0));
737
738     fields.forEach(field => {
739       const r = parseInt(field.charAt(2), 10) - 1;
740       const c = parseInt(field.charAt(4), 10) - 1;
741       const val = parseFloat(matrix_name[field]);
742       tmp_matrix[r][c] = val;
743     });
744
745     return tmp_matrix;
746   } else {
747     return 0;
748   }
749 }
750 /**
751 * Retrieves the linear input variables.
752 * @returns {string|boolean} - The linear input variables or false if the
753   model is not linearized.
754 */

```

```

753     getlinear_inputs() {
754         if(this.linear_flag) {
755             return this.linear_inputs;
756         } else {
757             this.jsl.env.error("@getlinear_inputs: "+language.string(202));
758             return false;
759         }
760     }
761
762     /**
763      * Retrieves the linear output variables.
764      * @returns {string|boolean} - The linear output variables or false if the
765      * model is not linearized.
766      */
767     getlinear_outputs() {
768         if(this.linear_flag) {
769             return this.linear_outputs;
770         } else {
771             this.jsl.env.error("@getlinear_outputs: "+language.string(202));
772             return false;
773         }
774     }
775     /**
776      * Retrieves the linear state variables.
777      * @returns {string|boolean} - The linear state variables or false if the
778      * model is not linearized.
779      */
780     getlinear_states() {
781         if(this.linear_flag) {
782             return this.linear_states;
783         } else {
784             this.jsl.env.error("@getlinear_states: "+language.string(202));
785             return false;
786         }
787     }
788     /**
789      * Retrieves simulation solutions based on the provided arguments and result
790      * file.
791      * @param {string|string[]} [args] - A single variable name or an array of
792      * names to retrieve solutions for. If omitted, retrieves all variables.
793      * @param {string} [resultfile=this.resultfile] - The path to the result
794      * file.
795      * @returns {Promise<any>} - A promise that resolves with the simulation
796      * results or an error message.
797      */
798     async getSolutions(args, resultfile = this.resultfile) {
799         resultfile = resultfile.replace(/\//g, '/');
800         if(fs.existsSync(resultfile)) {
801             if(args && args.length > 0) {
802                 const variables = `{${
803                     args.join(' ')
804                 }}`;
805                 const simresult = await this.sendExpression(`readSimulationResult("${{
806                     resultfile
807                 }", ${variables}}`);
808                 await this.sendExpression("closeSimulationResultFile()");
809             }
810         }
811     }

```

```

801         return simresult ;
802     } else {
803         const variables = await this.sendExpression(`readSimulationResultVars(
804             "${resultfile}"`);
805         await this.sendExpression("closeSimulationResultFile()");
806         return variables;
807     } else {
808         this.jsl.env.error(`@getSolutions: ${language.string(208)} + ${resultfile}`);
809         return false;
810     }
811 }
812
813 /**
814 * Creates valid variable names by replacing invalid characters and
815 * categorizes them based on the structure name.
816 * @param {string} name - The original variable name.
817 * @param {any} value - The value of the variable.
818 * @param {string} structname - The structure name (e.g., 'continuous', 'parameter').
819 */
820 createValidNames(name, value, structname) {
821     const tmpname = name.replace(/[^a-zA-Z0-9]/g, '_'); // Replace invalid
822     characters with underscore
823     this.mapped_names[tmpname] = name;
824
825     if(structname === 'continuous') {
826         this.continuous_list[tmpname] = value;
827     } else if(structname === 'parameter') {
828         this.parameter_list[tmpname] = value;
829     } else if(structname === 'input') {
830         this.input_list[tmpname] = value;
831     } else if(structname === 'output') {
832         this.output_list[tmpname] = value;
833     }
834 }
835 /**
836 * Parses a given expression string into structured data based on predefined
837 * formats.
838 * Handles various formats including single and nested lists, records, and
839 * single elements.
840 * @param {string} args - The expression string to parse.
841 * @returns {Array|Object|string} - The parsed data which could be an array,
842 * an object, or a string.
843 */
844 parseExpression(args) {
845     // Use regular expressions to match strings and key parts of the
846     // expression
847     const final = args.match(/\((.*?)\)|[\{\}()]=|-?\d+(\.\d+)?([eE][+-]?\d+)?|[a-
848     zA-Z_][a-zA-Z0-9_.]*\g);
849
850     if(final.length > 1) {
851         if(final[0] === "{" && final[1] !== "}") {

```

```

846   // Handle single-level list
847   let buff = [];
848   for(let i = 0; i < final.length; i++) {
849     if (![ "{", "}", ")", "(", ","].includes(final[i])) {
850       const value = final[i].replace(/\//g, "");
851       buff.push(value);
852     }
853   }
854   return buff;
855
856 } else if (final[0] === "{" && final[1] === "(") {
857   // Handle nested lists
858   let buff = [];
859   let tmp = [];
860   for(let i = 1; i < final.length - 1; i++) {
861     if (final[i] === "{") {
862       tmp = [];
863     } else if (final[i] === ")") {
864       buff.push(tmp);
865       tmp = [];
866     } else {
867       tmp.push(final[i].replace(/\//g, ""));
868     }
869   }
870   return buff;
871
872 } else if (final[0] === "record") {
873   // Handle record structure
874   let result = {};
875   for(let i = 2; i < final.length - 2; i++) {
876     if (final[i] === "=") {
877       const key = final[i - 1];
878       const value = final[i + 1].replace(/\//g, "");
879       result[key] = value;
880     }
881   }
882   return result;
883
884 } else if (final[0] === "fail") {
885   // Handle failure case
886   return this.sendExpression("getErrorString()");
887 } else {
888   // Return as a simple string if no special cases match
889   return args.replace(/\//g, "");
890 }
891 } else if (final.length === 1) {
892   // Handle single element case
893   return final[0].replace(/\//g, "");
894 } else {
895   // Handle empty result
896   return args.replace(/\//g, "");
897 }
898 }
899
900 /**

```

```

901   * Closes the current session safely by cleaning up resources such as
902     * temporary files and network connections.
903   */
904   async close() {
905     if(this.portfile && fs.existsSync(this.portfile)) {
906       fs.unlinkSync(this.portfile);
907     }
908
909     if(this.active) {
910       await this.requester.close();
911       this.active = false;
912     }
913
914     killProcess(this.pid);
915   }
916 }
917
918 exports.PRDC_JSLAB_OPENMODELICA_LINK = PRDC_JSLAB_OPENMODELICA_LINK;

```

Listing 119 - om-link.js

```

1  /**
2   * @file JSLAB library optim Real Coded Mixed Integer Genetic Algorithm
3   * submodule
4   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
5   * PR-DC, Republic of Serbia
6   * info@pr-dc.com
7   * @version 0.0.1
8   */
9
10 /**
11  * Class for Real Coded Mixed Integer Genetic Algorithm - RCMIGA.
12 */
13 class PRDC_JSLAB_OPTIM_RCMIGA {
14
15 /**
16  * Creates an instance of PRDC_JSLAB_LIB_OPTIM_RCMIGA.
17  * @param {Object} problem - The optimization problem definition.
18  * @param {Object} opts - Configuration options for the algorithm.
19  */
20 constructor(problem, opts) {
21   this.flag = 'preinit';
22   this.problem = problem;
23   this.opts = opts;
24   this.state = {};
25   this.selection = {};
26   this.solution = {};
27   this.constrained = problem.constrained;
28   this.stoped = 0;
29
30   this.checkInputs();
31   if(this.bounded) {
32     this.opts.lbm = repCol(this.opts.lb, opts.PopulationSize);
33     this.opts.ubm = repCol(this.opts.ub, opts.PopulationSize);
34   }

```

```

34
35   this.opts.a = opts.a ?? 0;
36   this.opts.b_real = opts.b_real ?? 0.15;
37   this.opts.b_integer = opts.b_integer ?? 0.35;
38   this.opts.p_real = opts.p_real ?? 10;
39   this.opts.p_integer = opts.p_integer ?? 4;
40   this.opts.UseVectorized = opts.UseVectorized ?? false;
41   this.opts.UseParallel = opts.UseParallel ?? false;
42   this.opts.Display = opts.Display ?? 'iter';
43
44 // Inicijalizacija
45 this.flag = 'init';
46 this.outputFcn();
47 this.state.StartTime = tic;
48 this.state.StallTime = tic;
49 this.state.StallGenerations = 1;
50 this.state.RandSeed = 'rcmiga';
51 this.state.Generation = 0;
52
53   this.rand = seedRandom(this.state.RandSeed);
54 }
55
56 /**
57 * Executes the optimization process.
58 */
59 async run(parallel_context, parallel_setup_fun) {
60   if(this.opts.UseParallel) {
61     this.parallel_context = parallel_context;
62     this.parallel_setup_fun = parallel_setup_fun;
63     parallel.terminate();
64   }
65
66   this.initState();
67   this.initPopulation();
68   await this.updateState();
69
70   this.flag = 'iter';
71   this.outputFcn();
72   this.createPlotFcn();
73   this.plotFcn();
74
75 // Optimizacija
76 var nvars = this.problem.nvars;
77 var newPopulation = zeros(nvars * this.opts.PopulationSize);
78 var stop = 0;
79 while(!stop) { // Petlja za optimizaciju
80   // Merenje vremena
81   this.state.GenTime = tic;
82
83   // Elitizam
84   if(this.state.ReproductionCount.elite > 0) {
85     var [S, I] = sorti(this.state.FunVal);
86     this.selection.current_elite = getSub(I, this.selection.elite);
87     setSub(newPopulation, index(range(0, nvars - 1),
88       this.selection.elite, nvars), getSub(this.state.Population,

```

```

89         index(range(0, nvars - 1), this.selection.current_elite, nvars));
90     }
91
92     // Selekcija
93     this.state.parents = this.selectionFcn();
94
95     // Ukrstanje
96     setSub(newPopulation, index(range(0, nvars - 1),
97         this.selection.children, nvars), this.crossoverFcn());
98
99     // Mutacija
100    setSub(newPopulation, index(range(0, nvars - 1),
101        this.selection.mutants, nvars), this.mutationFcn());
102
103    // Provera granica
104    this.state.Population = [... newPopulation];
105    this.integerRestriction();
106    if(this.bounded) {
107        this.checkBounds();
108    }
109
110    // Odredjivanje vrednosti funkcije prilagodjenosti i ogranicenja
111    this.state.Generation = this.state.Generation + 1;
112    await this.updateState();
113
114    // Prikaz stanja
115    stop = this.stoppingCriteria(); // Kriterijumi zaustavljanja
116    this.outputFcn();
117    this.plotFcn();
118    await waitMSeconds(5);
119}
120
121 this.solution.generations = this.state.Generation;
122 var [x_min, I] = mini(this.state.FunVal);
123 this.solution.feasible = 1;
124 if(this.constrained && this.state.ConSumVal[I] > 0) {
125     this.solution.feasible = 0;
126 }
127 this.solution.x = getSub(this.state.Population,
128     index(range(0, nvars - 1), I, nvars));
129 this.solution.fval = this.problem.fitnessfcn(this.solution.x);
130 this.solution.StoppingCriteria = stop;
131
132 this.flag = 'done';
133 this.outputFcn();
134 }
135
136 /**
137 * Validates the input parameters and options.
138 */
139 checkInputs() {
140     if(typeof this.problem.nonlconfcn === 'function') {
141         this.constrained = 1;
142     }
143     if(typeof this.problem.IntCon === 'undefined') {

```

```

144     this.problem.IntCon = [];
145 }
146
147 this.bounded = true;
148 if(typeof this.opts.lb === 'undefined' || typeof this.opts.ub === 'undefined') {
149     this.bounded = false;
150     if(typeof this.opts.InitialUnboundedRange === 'undefined') {
151         this.opts.InitialUnboundedRange = createFilledArray(this.problem.nvars,
152                     [0, 1]);
153     }
154 } else if((this.problem.lb && this.problem.lb.length != this.problem.nvars) ||
155             (this.problem.ub && this.problem.ub.length != this.problem.nvars)) {
156     throw new Error('Problem bounds have invalid dimension!');
157 }
158
159 if(this.opts.UseVectorized) {
160     if(this.opts.UseParallel == true){
161         disp('Option UseParallel is ignored while option UseVectorized is true');
162     } else if(this.opts.UseParallel != false){
163         disp('Option UseParallel must be true or false.');
164     }
165 } else {
166     if(![true, false].includes(this.opts.UseParallel)) {
167         throw new Error('Option UseParallel must be true or false!');
168     }
169 }
170
171 /**
172 * Creates the initial population.
173 */
174 creationFcn() {
175     return this.creationMixedUniform();
176 }
177
178 /**
179 * Selects individuals from the population.
180 */
181 selectionFcn() {
182     return this.binaryTournamentSelection();
183 }
184
185 /**
186 * Performs crossover between selected parents.
187 */
188 crossoverFcn() {
189     return this.laplaceMixedCrossover();
190 }
191
192 /**
193 * Mutates individuals in the population.
194 */

```

```
195 mutationFcn() {
196     return this.powerMixedMutation();
197 }
198 /**
199 * Initializes the plotting function.
200 */
201 createPlotFcn() {
202 }
203 /**
204 * Updates the graphical plot with current optimization status.
205 */
206 plotFcn() {
207     this.displayPlot();
208 }
209 /**
210 * Renders the optimization plot.
211 */
212 displayPlot() {
213 }
214 /**
215 * Handles button events to stop the optimization process.
216 */
217 buttonCallback() {
218     this.stoped = 1;
219 }
220 /**
221 * Manages the output display based on configuration.
222 */
223 outputFcn() {
224     if(['iter', 'final'].includes(this.opts.Display)) {
225         this.displayOutput();
226     }
227 }
228 /**
229 * Displays the current state of optimization.
230 */
231 displayOutput() {
232     if(this.opts.Display === 'iter') {
233         switch(this.flag) {
234             case 'init':
235                 disp(' Optimization is initialized!');
236                 if(this.opts.UseVectorized) {
237                     disp(' Vectorized functions evaluation in use.');
238                 } else if(this.opts.UseParallel){
239                     disp(' Parallel functions evaluation in use.');
240                 }
241             break;
242         }
243     }
244 }
```

```

250      case 'iter':
251        if(this.state.Generation === 0 || this.state.Generation % 20 === 0)
252        {
253          if(this.constrained) {
254            dispMonospaced(`\n    Stall`);                                Best      Max
255            dispMonospaced(`    Generation\n    Generations`);           f(x)     Constraint
256          } else {
257            dispMonospaced(`\n    dispMonospaced(`    Generation
258                           f(x)     Best      Stall`);                      f(x)     Generations`);
259          }
260        }
261
262        let fval_s = num2str(this.state.Best[this.state.Generation], 5);
263        if(this.constrained) {
264          if(!this.state.Feasible[this.state.Generation]) {
265            fval_s += '*';
266            dispMonospaced(
267              `    ${num2str(this.state.Generation, 0).padStart(10)}  ${
268                fval_s.padStart(12)}  ${
269                  num2str(max(this.state.ConSumVal), 2).padStart(12)}  ${
270                    num2str(this.state.StallGenerations, 0).padStart(12)}`
271            );
272          } else {
273            dispMonospaced(
274              `    ${num2str(this.state.Generation, 0).padStart(10)}  ${
275                fval_s.padStart(12)}  ${
276                  num2str(this.state.StallGenerations, 0).padStart(12)}`
277            );
278          }
279          break;
280        case 'done':
281          var str = `\nOptimization is done`;
282          if(this.constrained) {
283            if(this.state.Feasible[this.state.Generation]) {
284              str += ', solution found';
285            } else {
286              str += ', no feasible solution found';
287            }
288            disp(` ${str}, stopping criteria = ${this.solution.StoppingCriteria
289                  }!\n`);
290            break;
291          }
292        } else if(this.opts.Display === 'final') {
293          if(this.flag === 'done') {
294            var str = `\nOptimization is done`;
295            if(this.constrained) {
296              if(this.state.Feasible[this.state.Generation]) {
297                str += ', solution found';
298              } else {
299                str += ', no feasible solution found';
300              }
301            }
302          }
303        }
304      }
305    }
306  }
307}

```

```

301      }
302      disp(` ${str} , stopping criteria = ${this.solution.StoppingCriteria}!\`n`);
303    }
304  }
305
306
307 /**
308 * Creates the initial population with mixed uniform distribution.
309 */
310 creationMixedUniform() {
311   var lb = this.opts.lb;
312   var ub = this.opts.ub;
313   if(!this.bounded) {
314     lb = this.opts.InitialUnboundedRange.map((v) => v[0]);
315     ub = this.opts.InitialUnboundedRange.map((v) => v[1]);
316   }
317   var N = this.problem.IntCon.length;
318   var population = arrayRand(lb, ub,
319     this.problem.nvars, this.opts.PopulationSize, this.rand);
320   if(N) {
321     var r = arrayRandi([0, 1], N, this.opts.PopulationSize, this.rand);
322     var I = index(this.problem.IntCon,
323       range(0, this.opts.PopulationSize - 1), this.problem.nvars);
324
325     setSub(population, I, plus(fix(getSub(population, I)), r));
326   }
327   return population;
328 }
329
330 /**
331 * Performs binary tournament selection of parents.
332 */
333 binaryTournamentSelection() {
334   var parents = zeros(this.state.nParents);
335   var r1 = arrayRandi([0, this.opts.PopulationSize - 1], 2,
336     this.state.nParents, this.rand);
337   var r2 = arrayRandi([0, this.opts.PopulationSize - 1], 2,
338     this.state.nParents, this.rand);
339   var neq = elementWise((a, b) => a > b, r1, r2);
340   var I1 = indexOfAll(neq, true); // closer r2
341   var I2 = indexOfAll(neq, false); // closer r1
342   setSub(parents, I1, getSub(r2, I1));
343   setSub(parents, I2, getSub(r1, I2));
344   return parents;
345 }
346
347 /**
348 * Executes Laplace mixed crossover to generate offspring.
349 */
350 laplaceMixedCrossover() {
351   var N = this.problem.IntCon.length;
352   var nvars = this.problem.nvars;
353   var iParents = this.state.parents;
354   var nChildren = this.state.ReproductionCount.children;

```



```

405   var ubm = repCol(this.opts.ub, nMutants);
406   var lbum = repCol(this.opts.lb, nMutants);
407   var t = elementWise((a, b, c) => (a - b)/(c - a), parents, this.opts.lb,
408     this.opts.ub);
409   mutants = elementWise((a, b, c) => a+b*(a-c), parents, s, lbum);
410
411   var I = indexOfAll(elementWise((a, b) => a < b, t, r), true);
412   setSub(mutants, I, elementWise((a, b, c) => a-b*(c-a),
413     getSub(parents, I), getSub(s, I), getSub(ubm, I)));
414 }
415
416 return mutants;
417
418 /**
419 * Applies Gaussian mutation to selected individuals.
420 */
421 gaussianMutation() {
422
423 }
424
425 /**
426 * Ensures integer constraints are met for specific variables.
427 */
428 integerRestriction() {
429   var N = this.problem.IntCon.length;
430   if(N) {
431     var I = index(this.problem.IntCon,
432       [...this.selection.children, ...this.selection.mutants], this.problem.
433         nvars);
434     var p = getSub(this.state.Population, I);
435
436     var r = arrayRandi([0, 1], N,
437       this.state.ReproductionCount.children +
438         this.state.ReproductionCount.mutants, this.rand);
439
440     var Is = indexOfAll(neg(isInteger(p)), true);
441     setSub(p, Is, plus(fix(getSub(p, Is)), getSub(r, Is)));
442     setSub(this.state.Population, I, p);
443   }
444
445 /**
446 * Checks and enforces variable bounds within the population.
447 */
448 checkBounds() {
449   this.state.Population = elementWise((a, b, c) => min([max([a, b]), c]),
450     this.state.Population, this.opts.lbm, this.opts.ubm);
451 }
452
453 /**
454 * Evaluates whether stopping criteria have been met.
455 * @returns {number} Code indicating the reason to stop or continue.
456 */
457 stoppingCriteria() {

```

```

458     var stop = 0;
459     var gen = this.state.Generation;
460
461     if(gen > 0) {
462         if(this.state.Best[gen] < this.state.Best[gen - 1] ||
463             !this.state.Feasible[gen] ||
464             this.state.Feasible[gen] > this.state.Feasible[gen - 1]) {
465             this.state.StallTime = tic;
466             this.state.StallGenerations = 1;
467         } else {
468             this.state.StallGenerations = this.state.StallGenerations + 1;
469         }
470     }
471
472     if(gen >= (this.opts.MaxGenerations - 1)) {
473         // 1 - Maksimalni broj generacija ostvaren
474         stop = 1;
475     } else if(toc(this.state.StartTime) >= this.opts.MaxTime) {
476         // 2 - Proteklo maksimalno vreme trajanja optimizacije
477         stop = 2;
478     } else if(this.state.Feasible[gen] &&
479             this.state.Best[gen] <= this.opts.FitnessLimit) {
480         // 3 - Funkcija prilagodjenosti dostigla ciljanu vrednost
481         stop = 3;
482     } else if(gen > (this.opts.MaxStallGenerations - 2) &&
483             this.state.Feasible[gen - (this.opts.MaxStallGenerations - 2)] &&
484             this.state.Feasible[gen] &&
485             ((this.state.Best[gen - (this.opts.MaxStallGenerations - 2)] -
486             this.state.Best[gen]) <= this.opts.FunctionTolerance)) {
487         // 4 - Promena vrednosti funkcija manja od tolerancije
488         stop = 4;
489     } else if(this.state.StallGenerations >= this.opts.MaxStallGenerations) {
490         // 5 - Bez promene vrednosti funkcije kroz odredjeni broj generacija
491         stop = 5;
492     } else if(toc(this.state.StallTime) >= this.opts.MaxStallTime) {
493         // 6 - Bez promene vrednosti funkcije odredjeno vreme
494         stop = 6;
495     } else if(this.stoped) {
496         // 7 - Korisnik zaustavlja proces
497         stop = 7;
498     }
499     return stop;
500 }
501
502 /**
503 * Initializes the population for the optimization process.
504 */
505 initPopulation() {
506     this.state.Population = this.creationFcn();
507     this.integerRestriction();
508     if(this.bounded) {
509         this.checkBounds();
510     }
511 }
512

```

```

513 /**
514 * Initializes the internal state of the algorithm.
515 */
516 initState() {
517 // Merenje vremena
518 this.state.GenTime = tic;
519
520 this.state.FunVal = zeros(this.opts.PopulationSize);
521
522 this.state.ConSumVal = zeros(this.opts.PopulationSize);
523 this.state.Best = zeros(this.opts.MaxGenerations);
524 this.state.Feasible = ones(this.opts.MaxGenerations);
525 this.state.Time = zeros(this.opts.MaxGenerations);
526 this.state.x = zeros(this.opts.MaxGenerations * this.problem.nvars);
527
528 this.state.ReproductionCount = {};
529 this.state.ReproductionCount.elite = this.opts.EliteCount;
530 this.state.ReproductionCount.children = fix(this.opts.CrossoverFraction *
531 (this.opts.PopulationSize - this.opts.EliteCount));
532
533 if(mod(this.state.ReproductionCount.children, 2) != 0) {
534   this.state.ReproductionCount.children = this.state.ReproductionCount.
535     children - 1;
536 }
537
538 this.state.ReproductionCount.mutants = this.opts.PopulationSize -
539   (this.state.ReproductionCount.elite + this.state.ReproductionCount.
540     children);
541
542 this.state.parentsCount = {};
543 this.state.parentsCount.crossover = this.state.ReproductionCount.children;
544 this.state.parentsCount.mutation = this.state.ReproductionCount.mutants;
545 this.state.nParents = this.state.parentsCount.crossover +
546   this.state.parentsCount.mutation;
547
548 this.state.parentsSelection = {};
549 this.state.parentsSelection.crossover = range(0,
550   this.state.parentsCount.crossover - 1);
551 this.state.parentsSelection.mutation = plus(
552   this.state.parentsCount.crossover,
553   range(0, this.state.parentsCount.mutation - 1));
554
555 this.selection.elite = range(0, this.state.ReproductionCount.elite - 1);
556 this.selection.children = plus(this.state.ReproductionCount.elite,
557   range(0, this.state.ReproductionCount.children - 1));
558 this.selection.mutants = plus((this.state.ReproductionCount.elite +
559   this.state.ReproductionCount.children),
560   range(0, this.state.ReproductionCount.mutants - 1));
561
562 if(this.constrained) {
563   this.state.ConSize = [
564     this.problem.nonlconfcn(zeros(this.problem.nvars)).length, this.opts.
565     PopulationSize];
566
567 this.state.ConVal = zeros(this.state.ConSize[0] * this.state.ConSize[1])

```

```

565 ;
566   this.state.ConNormVal = zeros(this.state.ConSize[0] * this.state.ConSize
567   [1]);
568 }
569 /**
570 * Updates the current state of the algorithm based on evaluations.
571 */
572 async updateState() {
573   await this.evalFitnessFcn();
574   if(this.constrained) {
575     await this.evalConstraintsFcn();
576     this.updatePenalty();
577   }
578   this.state.Time[this.state.Generation] = toc(this.state.GenTime);
579   var i;
580   [this.state.Best[this.state.Generation], i] = mini(this.state.FunVal);
581   setSub(this.state.x, index(range(0, this.problem.nvars - 1),
582         this.state.Generation, this.problem.nvars),
583         getSub(this.state.Population,
584         index(range(0, this.problem.nvars - 1), i, this.problem.nvars)));
585
586   if(this.constrained && this.state.ConSumVal[i] > 0) {
587     this.state.Feasible[this.state.Generation] = 0;
588   }
589 }
590 /**
591 * Updates penalty values based on constraint violations.
592 */
593 updatePenalty() {
594   if(any(this.state.ConSumVal.map((a) => a == 0))) {
595     if(this.state.Generation == 0) {
596       var [s, i] = sorti(this.state.FunVal);
597       this.selection.current_elite = getSub(i, this.selection.elite);
598     }
599     var fmin;
600     if(this.state.ReproductionCount.elite > 0) {
601       // Ne moze da bude bolje od najgore elitne jedinke
602       fmin = max(getSub(this.state.FunVal, this.selection.current_elite));
603     } else {
604       // Ne moze da bude bolje od najbolje jedinike
605       fmin = min(getSub(this.state.FunVal, this.selection.current_elite));
606     }
607     var I1 = indexOfAll(elementWise((a, b) => a > 0 && b <= fmin,
608                           this.state.ConSumVal, this.state.FunVal), true);
609     var I2 = indexOfAll(elementWise((a, b) => a > 0 && b > fmin,
610                           this.state.ConSumVal, this.state.FunVal), true);
611     setSub(this.state.FunVal, I1, plus(fmin,
612           getSub(this.state.ConSumVal, I1)));
613     setSub(this.state.FunVal, I2, plus(getSub(this.state.FunVal, I2),
614           getSub(this.state.ConSumVal, I2)));
615   } else {
616     this.state.FunVal = this.state.ConSumVal;

```

```

618     }
619   }
620
621   /**
622    * Evaluates the fitness function for the current population.
623    */
624   async evalFitnessFcn() {
625     var obj = this;
626     var nvars = this.problem.nvars;
627
628     var p;
629     var n;
630     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
631       p = getSub(this.state.Population, index(range(0, nvars - 1),
632         [...this.selection.children, ...this.selection.mutants], nvars));
633       n = this.state.ReproductionCount.children + this.state.ReproductionCount
634         .mutants;
635     } else {
636       p = this.state.Population;
637       n = this.opts.PopulationSize;
638     }
639
640     var val;
641     if(this.opts.UseVectorized) {
642       val = this.problem.fitnessfcn(p);
643     } else {
644       if(this.opts.UseParallel) {
645         val = await parallel.parfor(0, n-1, 1, parallel.getProcessorsNum(),
646           this.parallel_context, this.parallel_setup_fun, 'function(i) {
647             var fun = ${obj.problem.fitnessfcn.toString()};
648             return fun(getSub(${JSON.stringify(p)}, index(range(0, ${nvars} -
649               1), i, ${nvars})));
650           }');
651       } else {
652         val = zeros(n);
653         for(var i = 0; i < n; i++) {
654           val[i] = this.problem.fitnessfcn(getSub(p, index(range(0, nvars - 1)
655             , i, nvars)));
656         }
657       }
658     }
659     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
660       this.state.FunVal = [...getSub(this.state.FunVal, this.selection.
661         current_elite), ...val];
662     } else {
663       this.state.FunVal = val;
664     }
665   /**
666    * Evaluates constraint functions for the current population.
667    */
668   async evalConstraintsFcn() {
669     var obj = this;

```

```

669   var nvars = this.problem.nvars;
670
671   var p;
672   var n;
673   var m = this.state.ConSize[0];
674   if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
675     p = getSub(this.state.Population, index(range(0, nvars - 1),
676           [...this.selection.children, ...this.selection.mutants], nvars));
677     n = this.state.ReproductionCount.children + this.state.ReproductionCount
678       .mutants;
679   } else {
680     p = this.state.Population;
681     n = this.opts.PopulationSize;
682   }
683
684   var val;
685   if(this.opts.UseVectorized) {
686     val = obj.problem.nonlconfcn(p);
687   } else {
688     if(this.opts.UseParallel) {
689       val = await parallel.parfor(0, n-1, 1, parallel.getProcessorsNum(),
690         this.parallel_context, this.parallel_setup_fun, 'function(i) {
691           var fun = ${obj.problem.nonlconfcn.toString()};
692           return fun(getSub(${JSON.stringify(p)}, index(range(0, ${nvars} -
693             1), i, ${nvars})));
694         }');
695       val = val.flat();
696     } else {
697       m = this.state.ConSize[0];
698       val = zeros(m * n);
699       for(var i = 0; i < n; i++) {
700         setSub(val, index(range(0, m - 1), i, m),
701               this.problem.nonlconfcn(getSub(p,
702                 index(range(0, nvars - 1), i, nvars))));
703       }
704     }
705   }
706   if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
707     this.state.ConVal = concatCol(m, getSub(this.state.ConVal,
708       index(range(0, m - 1), this.selection.current_elite, m)), val);
709   } else {
710     this.state.ConVal = val;
711   }
712   this.normConstraintsFcn();
713   this.state.ConSumVal = sumCol(this.state.ConNormVal, m, this.opts.
714     PopulationSize);
715
716 /**
717  * Normalizes constraint values to maintain consistent scaling.
718  */
719 normConstraintsFcn() {
720   var ConNormVal = this.state.ConVal;
721   var sNaN = indexOfAll(isNaN(ConNormVal), true);
722   var sInf = indexOfAll(isInfinity(ConNormVal), true);

```

```

721     var sNeg = indexOfAll(isNegative(ConNormVal), true);
722
723     setSub(ConNormVal, sNaN, zeros(sNaN.length));
724     setSub(ConNormVal, sInf, zeros(sInf.length));
725     setSub(ConNormVal, sNeg, zeros(sNeg.length));
726
727     var m = this.state.ConSize[0];
728     var n = this.opts.PopulationSize;
729
730     if(this.state.Generation == 0 ||
731         (this.state.Generation > 0 &&
732          this.state.Feasible[this.state.Generation-1])) {
733
734         // Use first values of the first generation for normalization of
735         // constraints until there is a feasible solution
736         var l = elementWise((a) => Math.sqrt(a),
737             sumRow(elementWise((a) => Math.pow(a, 2), ConNormVal), m, n));
738         var I = indexOfAll(l, 0);
739         setSub(l, I, ones(I.length));
740         if(isEmpty(this.l)) {
741             this.l = l;
742         } else {
743             var s = indexOfAll(elementWise((a, b) => a > b, this.l, l), true);
744             setSub(this.l, s, getSub(l, s));
745         }
746
747         this.lm = repCol(this.l, n);
748     }
749
750     setSub(ConNormVal, sNaN, getSub(this.lm, sNaN));
751     setSub(ConNormVal, sInf, getSub(this.lm, sInf));
752     this.state.ConNormVal = divideEl(ConNormVal, this.lm);
753 }
754 }
755
756 exports.PRDC_JSLAB_OPTIM_RCMIGA = PRDC_JSLAB_OPTIM_RCMIGA;

```

Listing 120 - optim-rcmiga.js

```

1  /**
2   * @file JSLAB library optim submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  var { PRDC_JSLAB_OPTIM_RCMIGA } = require('./optim-rcmiga');
9
10 /**
11  * Class for JSLAB optim submodule.
12  */
13 class PRDC_JSLAB_LIB_OPTIM {
14
15 /**
16  * Initializes the optim submodule.
17  * @param {Object} js1 Reference to the main JSLAB object.

```

```

18  /*
19   * constructor(jsl) {
20     var obj = this;
21     this.jsl = jsl;
22   }
23
24 /**
25 * Minimizes an unconstrained function using a coordinate descent-like
26 * Powell algorithm.
27 * @param {function} fnc Function to be minimized. Accepts an array of size
28 * N and returns a scalar.
29 * @param {Array} x0 Initial guess for the parameters as an array of size N.
30 * @param {Object} [options] Optional parameters:
31 *   - eps: Convergence threshold (default: 1e-6)
32 *   - alpha: Initial step size scaling factor (default: 0.001)
33 *   - stepSize: Finite difference step size for gradient estimation (
34 *     default: 1e-6)
35 *   - maxIterations: Maximum number of iterations to prevent infinite loops
36 *     (default: 1000)
37 * @return {Object} An object with two fields:
38 *   - argument: The parameter array that minimizes the function.
39 *   - fncvalue: The function value at the minimized parameters.
40 */
41 optimPowell(fnc, x0, options = {}) {
42   const {
43     eps = 1e-6,
44     alpha = 0.001,
45     stepSize = 1e-6,
46     maxIterations = 1000
47   } = options;
48
49   let convergence = false;
50   let x = x0.slice(); // Create a copy of the initial guess
51   let currentAlpha = alpha; // Current step size scaling factor
52
53   let iteration = 0;
54   let dx;
55
56   while(!convergence && iteration < maxIterations) {
57     iteration++;
58     const indices = shuffleIndices(x);
59     convergence = true; // Assume convergence until a significant update is
60     // found
61
62     // Iterate over each variable in shuffled order
63     for(let i = 0; i < indices.length; i++) {
64       const idx = indices[i];
65
66       // Estimate the derivative (gradient) using finite differences
67       x[idx] += stepSize;
68       const fxi = fnc(x);
69       x[idx] -= stepSize;
70     }
71   }
72 }
```

```

68     dx = (fxi - fx) / stepSize;
69
70     // Check convergence based on the derivative
71     if(Math.abs(dx) > eps) {
72         convergence = false;
73     }
74
75     // Update the parameter by moving against the gradient
76     x[idx] -= currentAlpha * dx;
77
78     // Update the function value after the parameter change
79     fx = fnc(x);
80 }
81
82     // Adaptive step size adjustment
83     if(fx < pfx) {
84         currentAlpha *= 1.1; // Increase step size if improvement
85     } else {
86         currentAlpha *= 0.7; // Decrease step size if no improvement
87     }
88     pfx = fx;
89
90     // Optional: Log progress every 100 iterations
91     if(options.disp && iteration % 100 === 0) {
92         this.jsl.env.disp(`Iteration ${iteration}: f(x) = ${fx}`);
93     }
94 }
95
96     return { x, fx };
97 }
98
99 /**
100 * Performs optimization using the Nelder-Mead algorithm.
101 * @param {Function} f - The objective function to minimize. It should
102 * accept an array of numbers and return a scalar value.
103 * @param {number[]} x0 - An initial guess for the parameters as an array of
104 * numbers.
105 * @param {Object} [parameters] - Optional parameters to control the
106 * optimization process.
107 * @param {number} [parameters.maxIterations=x0.length * 200] - Maximum
108 * number of iterations to perform.
109 * @param {number} [parameters.nonZeroDelta=1.05] - Scaling factor for non-
110 * zero initial steps in the simplex.
111 * @param {number} [parameters.zeroDelta=0.001] - Initial step size for
112 * parameters that are initially zero.
113 * @param {number} [parameters.minErrorDelta=1e-6] - Minimum change in
114 * function value to continue iterations.
115 * @param {number} [parameters.minTolerance=1e-5] - Minimum change in
116 * parameters to continue iterations.
117 * @param {number} [parameters.rho=1] - Reflection coefficient.
118 * @param {number} [parameters.chi=2] - Expansion coefficient.
119 * @param {number} [parameters.psi=-0.5] - Contraction coefficient.
120 * @param {number} [parameters.sigma=0.5] - Reduction coefficient.
121 * @param {Array<Object>} [parameters.history] - Optional array to store the
122 * history of simplex states for analysis.

```

```

114  *
115  * @returns {{ fx: number, x: number[] }} An object containing:
116  *   - 'fx': The minimum function value found.
117  *   - 'x': The parameters corresponding to the minimum function value.
118  */
119 optimNelderMead (... args) {
120   return this.jsl.env.fmin.nelderMead (... args);
121 }
122
123 /**
124  * Performs optimization using the Conjugate Gradient method.
125  * @param {Function} f - The objective function to minimize. It should
126  *   accept an array of numbers and return a scalar value and its gradient.
127  * @param {number[]} initial - An initial guess for the parameters as an
128  *   array of numbers.
129  * @param {Object} [params] - Optional parameters to control the
130  *   optimization process.
131  * @param {number} [params.maxIterations=initial.length * 20] - Maximum
132  *   number of iterations to perform.
133  * @param {Array<Object>} [params.history] - Optional array to store the
134  *   history of optimization steps for analysis.
135  *
136  * @returns {{ fx: number, x: number[], fxprime: number[] }} An object
137  *   containing:
138  *   - 'fx': The minimum function value found.
139  *   - 'x': The parameters corresponding to the minimum function value.
140  *   - 'fxprime': The gradient of the function at the minimum.
141  */
142 optimConjugateGradient (... args) {
143   return this.jsl.env.fmin.conjugateGradient (... args);
144 }
145
146 /**
147  * Performs optimization using the Gradient Descent method.
148  * @param {Function} f - The objective function to minimize. It should
149  *   accept an array of numbers and return a scalar value and its gradient.
150  * @param {number[]} initial - An initial guess for the parameters as an
151  *   array of numbers.
152  * @param {Object} [params] - Optional parameters to control the
153  *   optimization process.
154  * @param {number} [params.maxIterations=initial.length * 100] - Maximum
155  *   number of iterations to perform.
156  * @param {number} [params.learnRate=0.001] - Learning rate or step size for
157  *   each iteration.
158  * @param {Array<Object>} [params.history] - Optional array to store the
159  *   history of optimization steps for analysis.
160  *
161  * @returns {{ fx: number, x: number[], fxprime: number[] }} An object
162  *   containing:
163  *   - 'fx': The minimum function value found.
164  *   - 'x': The parameters corresponding to the minimum function value.
165  *   - 'fxprime': The gradient of the function at the minimum.
166  */
167 optimGradientDescent (... args) {
168   return this.jsl.env.fmin.gradientDescent (... args);

```

```

156   }
157
158  /**
159   * Performs optimization using the Gradient Descent method with Wolfe Line
160   * Search.
161   * @param {Function} f - The objective function to minimize. It should
162   * accept an array of numbers and return a scalar value and its gradient.
163   * @param {number[]} initial - An initial guess for the parameters as an
164   * array of numbers.
165   * @param {Object} [params] - Optional parameters to control the
166   * optimization process.
167   * @param {number} [params.maxIterations=initial.length * 100] - Maximum
168   * number of iterations to perform.
169   * @param {number} [params.learnRate=1] - Initial learning rate or step size
170   * for the line search.
171   * @param {number} [params.c1=1e-3] - Parameter for the Armijo condition in
172   * Wolfe Line Search.
173   * @param {number} [params.c2=0.1] - Parameter for the curvature condition
174   * in Wolfe Line Search.
175   * @param {Array<Object>} [params.history] - Optional array to store the
176   * history of optimization steps for analysis, including line search
177   * details.
178   * @returns {{ fx: number, x: number[], fxprime: number[] }} An object
179   * containing:
180   * - 'fx': The minimum function value found.
181   * - 'x': The parameters corresponding to the minimum function value.
182   * - 'fxprime': The gradient of the function at the minimum.
183   */
184 optimGradientDescentLineSearch(... args) {
185   return this.jsl.env.fmin.gradientDescentLineSearch(... args);
186 }
187
188 /**
189  * Performs root finding using the Bisection method.
190  * @param {Function} f - The function for which to find a root. It should
191  * accept a number and return a number.
192  * @param {number} a - The start of the interval. Must satisfy f(a) and f(b)
193  * have opposite signs.
194  * @param {number} b - The end of the interval. Must satisfy f(a) and f(b)
195  * have opposite signs.
196  * @param {Object} [parameters] - Optional parameters to control the root-
197  * finding process.
198  * @param {number} [parameters.maxIterations=100] - Maximum number of
199  * iterations to perform.
200  * @param {number} [parameters.tolerance=1e-10] - Tolerance for convergence.
201  * The method stops when the interval width is below this value.
202  * @returns {number} The root found within the interval [a, b].
203  */
204 optimBisect(... args) {
205   return this.jsl.env.fmin.bisect(... args);
206 }
207
208 /**
209  * Performs search using the Nelder-Mead algorithm.
210  * @param {Function} f - The objective function to minimize. It should

```

```

    accept an array of numbers and return a scalar value.
194  * @param {number[]} x0 - An initial guess for the parameters as an array of
      numbers.
195  * @param {Object} [parameters] - Optional parameters to control the
      optimization process.
196  * @param {number} [parameters.maxIterations=x0.length * 200] - Maximum
      number of iterations to perform.
197  * @param {number} [parameters.nonZeroDelta=1.05] - Scaling factor for non-
      zero initial steps in the simplex.
198  * @param {number} [parameters.zeroDelta=0.001] - Initial step size for
      parameters that are initially zero.
199  * @param {number} [parameters.minErrorDelta=1e-6] - Minimum change in
      function value to continue iterations.
200  * @param {number} [parameters.minTolerance=1e-5] - Minimum change in
      parameters to continue iterations.
201  * @param {number} [parameters.rho=1] - Reflection coefficient.
202  * @param {number} [parameters.chi=2] - Expansion coefficient.
203  * @param {number} [parameters.psi=-0.5] - Contraction coefficient.
204  * @param {number} [parameters.sigma=0.5] - Reduction coefficient.
205  * @param {Array<Object>} [parameters.history] - Optional array to store the
      history of simplex states for analysis.
206  *
207  * @returns {{ fx: number, x: number[] }} An object containing:
208  *   - 'fx': The minimum function value found.
209  *   - 'x': The parameters corresponding to the minimum function value.
210  */
211 fminsearch(... args) {
212   return this.jsl.env.fmin.nelderMead(... args);
213 }
214 /**
215  * Finds the minimum of a univariate function within a specified interval
216  * using a bracketing method.
217  * @param {function} func - The function to minimize. Should accept a single
218  *   number and return a number.
219  * @param {number} a - The lower bound of the interval.
220  * @param {number} b - The upper bound of the interval.
221  * @param {number} [tol=1e-5] - The tolerance for convergence (optional).
222  * @returns {{ fx: number, x: number[] }} An object containing:
223  *   - 'fx': The minimum function value found.
224  *   - 'x': The x-value where the function attains its minimum within [a, b]
225  *].
226 /**
227 fminbnd(func, a, b, tol = 1e-5) {
228   const eps = 1e-10; // A small value to prevent division by zero or to
229   // avoid precision issues.
230   const golden_ratio = (3 - Math.sqrt(5)) / 2;
231   let x = a + golden_ratio * (b - a);
232   let w = x;
233   let v = w;
234   let fx = func(x);
235   let fw = fx;
236   let fv = fw;

```

```

236     let d = 0;
237     let e = 0;
238
239     while(Math.abs(b - a) > tol) {
240         const m = 0.5 * (a + b);
241         const tol1 = tol * Math.abs(x) + eps;
242         const tol2 = 2 * tol1;
243
244         // Check for convergence
245         if(Math.abs(x - m) <= tol2 - 0.5 * (b - a)) {
246             break;
247         }
248
249         let u;
250         let useGolden = true;
251
252         // Try parabolic interpolation
253         if(Math.abs(e) > tol1) {
254             const r = (x - w) * (fx - fv);
255             const q = (x - v) * (fx - fw);
256             const p = (x - v) * q - (x - w) * r;
257             const q2 = 2 * (q - r);
258             const q2abs = Math.abs(q2);
259             if(q2abs > eps) {
260                 u = x - p / q2;
261                 if(a + tol1 <= u && u <= b - tol1 && Math.abs(u - x) < 0.5 * Math.
262                     abs(e)) {
263                     useGolden = false;
264                 }
265             }
266
267         // If parabolic interpolation is not used , fall back to golden section
268         if(useGolden) {
269             if(x < m) {
270                 u = x + golden_ratio * (b - x);
271             } else {
272                 u = x - golden_ratio * (x - a);
273             }
274             e = d;
275         } else {
276             e = d;
277         }
278
279         const fu = func(u);
280
281         // Update a, b, v, w, x
282         if(fu <= fx) {
283             if(u < x) b = x;
284             else a = x;
285             v = w; fv = fw;
286             w = x; fw = fx;
287             x = u; fx = fu;
288         } else {
289             if(u < x) a = u;

```

```

290     else b = u;
291     if(fu <= fw || w == x) {
292         v = w; fv = fw;
293         w = u; fw = fu;
294     } else if(fu <= fv || v == x || v == w) {
295         v = u; fv = fu;
296     }
297 }
298 }
299
300     return { x, fx };
301 }
302
303 /**
304 * Creates an instance of PRDC_JSLAB_LIB_OPTIM_RCMIGA.
305 * @param {Object} problem - The optimization problem definition.
306 * @param {Object} opts - Configuration options for the algorithm.
307 */
308 rcmiga(... args) {
309     return new PRDC_JSLAB_OPTIM_RCMIGA(... args);
310 }
311
312 }
313
314 exports.PRDC_JSLAB_LIB_OPTIM = PRDC_JSLAB_LIB_OPTIM;

```

Listing 121 - optim.js

```

1 /**
2 * @file JSLAB library parallel submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB parallel submodule.
10 */
11 class PRDC_JSLAB_PARALLEL {
12
13 /**
14 * Constructs parallel submodule object with access to JSLAB's parallel
15 * functions.
16 * @constructor
17 * @param {Object} jsl - Reference to the main JSLAB object.
18 */
19 constructor(jsl) {
20     this.jsl = jsl;
21     this.worker_pool = [];
22     this.task_queue = [];
23     this.is_initialized = false;
24 }
25
26 /**
27 * Retrieves the number of logical processors available.
28 * @returns {number} Number of processors.
29 */

```

```

28  /*
29   * getProcessorsNum() {
30     return this.jsl.env.processors_number || 4;
31   }
32
33 /**
34 * Generates the worker's internal script.
35 * @param {Object} [context={}] - Optional context to pass to the
36 *                               work_function.
37 * @param {Function|String} work_function_str - The work function to execute
38 *                                              .
39 * @param {Function|String} [setup_function_str] - Optional setup function
40 *                                               to execute on init.
41 */
42 workerFunction(context = {}, setup_function_str = "") {
43   return `
44     self.addEventListener("message", async function(e) {
45       if(e.data.type === 'execute') {
46         try {
47           const { work_fun_str, args } = e.data;
48
49           // Reconstruct the work function
50           const work_function = new Function(`return ${work_fun_str}`);
51
52           // Execute the work function with provided arguments
53           const result = await work_function(...args);
54
55           // Send back the result
56           self.postMessage({ type: 'result', result });
57         } catch(err) {
58           self.postMessage({ type: 'error', error: err });
59         }
60       }
61     });
62
63     // Assign context variables
64     Object.assign(self, ${JSON.stringify(context)});
65
66     // Reconstruct and execute the setup function if provided
67     (async () => {
68       const __setup = ${setup_function_str || 'null'};
69       if (typeof __setup === 'function') {
70         await __setup.call(self);
71       }
72       self.postMessage({ type: 'ready' });
73     })();
74   `;
75 }
76 /**
77 * Initializes the worker pool with the specified number of workers.
78 * @param {number} num_workers - Number of workers to initialize.
79 * @param {Object} [context={}] - Optional context to pass to the
80 *                               work_function.

```

```

79   * @param {Function|String} [setup_function_str] - Optional setup function
80   * to execute on init.
81   */
82 init(num_workers, context = {}, setup_function_str = "") {
83   if(this.is_initialized) return;
84
85   if(!num_workers || num_workers <= 0) {
86     num_workers = this.getProcessorsNum();
87   }
88
89   const worker_script =
90     `${this.jsl.getWorkerInit()}${this.workerFunction(context, setup_function_str)}`
91   ;
92
93   if(config.DEBUG_PARALLEL_WORKER_SETUP_FUN) {
94     this.jsl._console.log(worker_script);
95   }
96
97   const blob = new Blob([worker_script], { type: 'application/javascript' })
98   ;
99   const blobURL = URL.createObjectURL(blob);
100
101  for(let i = 0; i < num_workers; i++) {
102    const worker = new Worker(blobURL);
103    worker.busy = false;
104    worker.ready = false;
105    this.worker_pool.push(worker);
106  }
107
108  this.is_initialized = true;
109}
110
111 /**
112  * Assigns tasks from the queue to available workers.
113 */
114 assignTasksToWorkers() {
115   for(const worker of this.worker_pool) {
116     if(!worker.busy && this.task_queue.length > 0) {
117       const task = this.task_queue.shift();
118       worker.busy = true;
119
120       if(config.DEBUG_PARALLEL_WORKER_WORK_FUN) {
121         this.jsl._console.log(task);
122       }
123
124       function executeTask() {
125         worker.postMessage({
126           type: 'execute',
127           work_fun_str: task.work_function_str,
128           args: task.args,
129         });
130     }
131   }
132
133   worker.onmessage = (e) => {

```

```

132     if(e.data.type === 'ready') {
133         worker.ready = true;
134         executeTask();
135     } else if(e.data.type === 'result') {
136         task.resolve(e.data.result);
137     } else if(e.data.type === 'error') {
138         task.reject(new Error(e.data.error));
139     }
140     worker.busy = false;
141     this.assignTasksToWorkers();
142 };
143
144     worker.onerror = (e) => {
145         task.reject(new Error(e.message));
146         worker.busy = false;
147         this.assignTasksToWorkers();
148     };
149
150     if(worker.ready) {
151         executeTask();
152     }
153   }
154 }
155
156 /**
157 * Enqueues a task to be executed by the worker pool.
158 * @param {Object} context - Context variables to assign in the worker.
159 * @param {Function|String} [setup_function] - Optional setup function to
160   execute on init.
161 * @param {Array} args - Arguments to pass to the work_function.
162 * @param {Function|String} work_function - The work function to execute.
163 * @param {boolean} reset_workers - Whether to reset all workers or not.
164 * @returns {Promise} - Resolves with the result of the work_function.
165 */
166 run(context = {}, setup_function = null, args = [] , work_function,
167   reset_workers = false) {
168   var setup_function_str = setup_function;
169   if(typeof setup_function_str !== 'string') {
170       setup_function_str = this.jsl.eval.rewriteCode(this.jsl.eval.
171           getFunctionBody(setup_function)).code;
172   }
173   var work_function_str = work_function;
174   if(typeof work_function_str !== 'string') {
175       work_function_str = work_function.toString();
176   }
177   if(reset_workers) {
178       this.terminate();
179   }
180   if(!this.is_initialized) {
181       this.init(0, context, setup_function_str);
182   }
183   return new Promise((resolve, reject) => {
184       this.task_queue.push({
185           work_function_str,

```

```

184         args ,
185         resolve ,
186         reject ,
187     });
188
189     this.assignTasksToWorkers();
190 });
191 }
192
193 /**
194 * Executes a parallel for loop by dividing the iteration range among
195 * workers.
196 *
197 * @param {number} start - The initial value of the loop counter.
198 * @param {number} end - The terminating value of the loop counter.
199 * @param {number} [step=1] - The amount by which to increment the loop
200 *   counter each iteration.
201 * @param {number} [num_workers=this.getProcessorsNum()] - The number of
202 *   workers to use.
203 * @param {Object} [context={}] - Optional context to pass to the
204 *   work_function.
205 * @param {Function} [setup_function=null] - Optional setup function to
206 *   execute before work_function.
207 * @param {Function} work_function - The function to execute on each
208 *   iteration.
209 * @param {boolean} reset_workers - Whether to reset all workers or not.
210 * @returns {Promise<Array>} - A promise that resolves to an array of
211 *   results.
212 */
213 async parfor(start, end, step = 1, num_workers, context,
214   setup_function, work_function, reset_workers = false) {
215   var setup_function_str = this.jsl.eval.rewriteCode(this.jsl.eval.
216     getFunctionBody(setup_function)).code;
217   if(reset_workers) {
218     this.terminate();
219   }
220   if(!this.is_initialized) {
221     this.init(num_workers, context, setup_function_str);
222   }
223   if(step === 0) {
224     this.jsl.env.error('@parfor: '+language.string(197));
225   }
226   const isAscending = (end - start) * step > 0;
227   if(!isAscending) {
228     this.jsl.env.error('@parfor: '+language.string(198));
229   }
230   const total_items = Math.ceil(Math.abs((end - start) / step)) + 1;
231   const chunk_size = Math.ceil(total_items / num_workers);
232   const tasks = [];
233
234   const task_function = async function(chunk_start, chunk_end, step,
235

```

```

  work_fun_str) {
231  const work_function = new Function('return ' + work_fun_str)();
232  const results = [];
233  for(let i = chunk_start; i <= chunk_end; i += step) {
234    const result = await work_function(i);
235    results.push(result);
236  }
237  return results;
238};
239 var task_function_str = task_function.toString();
240
241 for(let worker_index = 0; worker_index < num_workers; worker_index++) {
242   const chunk_start_index = worker_index * chunk_size;
243   const chunk_end_index = Math.min(chunk_start_index + chunk_size,
244     total_items);
245
246   if(chunk_start_index >= chunk_end_index) {
247     break;
248   }
249
250   const chunk_start = start + chunk_start_index * step;
251   let chunk_end = start + (chunk_end_index * step) - step;
252
253   if(chunk_end > end) {
254     chunk_end = end;
255   }
256
257   var work_function_str = work_function;
258   if(typeof work_function === 'function') {
259     work_function_str = work_function.toString();
260   }
261
262   const task = this.run(
263     context,
264     setup_function_str,
265     [chunk_start, chunk_end, step, work_function_str],
266     task_function_str
267   );
268
269   tasks.push(task);
270 }
271
272 const nested_results = await Promise.all(tasks);
273 return nested_results.flat();
274
275 /**
276  * Terminates all workers and resets the worker pool.
277  */
278 terminate() {
279   for(const worker of this.worker_pool) {
280     worker.terminate();
281   }
282   this.worker_pool = [];
283   this.task_queue = [];

```

```

284     this.is_initialized = false;
285   }
286 }
287
288 exports.PRDC_JSLAB_PARALLEL = PRDC_JSLAB_PARALLEL;

```

Listing 122 - parallel.js

```

1  /**
2  * @file JSLAB library path submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9 * Class for JSLAB path submodule.
10 */
11 class PRDC_JSLAB_LIB_PATH {
12
13 /**
14 * Initializes a new instance of the path submodule, providing access to
15 * path manipulation utilities.
16 * @param {Object} js1 Reference to the main JSLAB object.
17 */
18 constructor(js1) {
19   var obj = this;
20   this.js1 = js1;
21 }
22 /**
23 * Extracts the directory of file.
24 * @param {String} path The filesystem path from which to extract the
25 * directory.
26 * @returns {String} The directory from the given path.
27 */
28 getDir(path) {
29   return this.js1.env.pathDirName(path);
30 }
31 /**
32 * Extracts the name of the directory from a given filesystem path.
33 * @param {String} path The filesystem path from which to extract the
34 * directory name.
35 * @returns {String} The name of the directory from the given path.
36 */
37 getDirName(path) {
38   return path.replaceAll('\\', '/').match(/([^\//]* )\/*$/)[1];
39 }
40 /**
41 * Retrieves the platform-specific path separator character.
42 * @returns {String} The path separator character used by the system.
43 */
44 pathSep() {
45   return this.js1.env.pathSep();

```

```
46     }
47
48     /**
49      * Determines if the current path is absolute.
50      * @returns {boolean} True if the current path is absolute, false otherwise.
51      */
52     isAbsolutePath() {
53       return this.jsl.env.pathIsAbsolute();
54     }
55
56     /**
57      * Joins all given path segments together using the platform-specific
58      * separator as a delimiter.
59      * @param {...string} paths The path segments to join.
60      * @returns {string} The combined path.
61      */
62     pathJoin(...args) {
63       return this.jsl.env.pathJoin(...args);
64     }
65
66     /**
67      * Retrieves the file name from the provided file path.
68      * @param {string} path - The complete file path.
69      * @returns {string} The file name extracted from the path.
70      */
71     pathFileName(path) {
72       return this.jsl.env.pathFileName(path);
73     }
74
75     /**
76      * Returns the last portion of a path, similar to the Unix ‘basename’
77      * command.
78      * @param {string} path - The file path to process.
79      * @returns {string} The last segment of the path.
80      */
81     pathBaseName(path) {
82       return this.jsl.env.pathBaseName(path);
83     }
84
85     /**
86      * Retrieves the file extension from the provided file path.
87      * @param {string} path - The complete file path.
88      * @returns {string} The file extension extracted from the path.
89      */
90     pathFileExt(path) {
91       return this.jsl.env.pathExtName(path);
92     }
93
94     /**
95      * Retrieves the file extension from the provided file path.
96      * @param {string} path - The complete file path.
97      * @returns {string} The file extension extracted from the path.
98      */
99     pathExtName(path) {
100       return this.jsl.env.pathExtName(path);
```

```
99     }
100
101    /**
102     * Resolves a sequence of path segments into an absolute path using the
103     * environment's path resolver.
104     * @param {string} path - The path or sequence of paths to resolve.
105     * @returns {string} - The resolved absolute path.
106     */
107    pathResolve(path) {
108      return this.jsl.env.pathResolve(path);
109    }
110
111    /**
112     * Computes the relative path from one path to another.
113     * @param {string} from - The starting path.
114     * @param {string} to - The target path.
115     * @returns {string} - The relative path from the 'from' path to the 'to'
116     * path.
117     */
118    pathRelative(from, to) {
119      return this.jsl.env.pathRelative(from, to);
120    }
121
122    /**
123     * Normalizes a given path, resolving '..' and '.' segments using the
124     * environment's path normalizer.
125     * @param {string} path - The path to normalize.
126     * @returns {string} - The normalized path.
127     */
128    pathNormalize(path) {
129      return this.jsl.env.pathNormalize(path);
130    }
131
132    /**
133     * Compares two file paths after resolving them to their absolute forms to
134     * check if they refer to the same location.
135     * @param {string} path1 - The first file path to compare.
136     * @param {string} path2 - The second file path to compare.
137     * @returns {boolean} Returns true if both paths resolve to the same
138     * absolute path, otherwise false.
139     */
140    comparePaths(path1, path2) {
141      return this.jsl.env.pathResolve(path1) === this.jsl.env.pathResolve(path2)
142      ;
143    }
144
145    /**
146     * Generates a unique filesystem path by appending a number to the input
147     * path if the original path exists.
148     * @param {String} path The base path for which a unique version is required
149     *
150     * @returns {String} A unique filesystem path based on the input path.
151     */
152    getUniquePath(path) {
153      var i = 0;
```

```

146     var unique_path = path;
147     while(fs.existsSync(unique_path)) {
148         i = i+1;
149         unique_path = path+i;
150     }
151     return unique_path;
152 }
153
154 /**
155 * Generates a unique filename by appending a number to the original path if
156 * it already exists.
157 * @param {string} path - The original file path.
158 * @param {string} ext - The original file extension.
159 * @returns {string} A unique folder path.
160 */
161 getUniqueFilename(filename, ext) {
162     var i = 0;
163     var unique_filename = filename+'.'+ext;
164     while(fs.existsSync(unique_filename)) {
165         i = i+1;
166         unique_filename = filename+i+'.'+ext;
167     }
168     return unique_filename;
169 }
170
171 exports.PRDC_JSLAB_LIB_PATH = PRDC_JSLAB_LIB_PATH;

```

Listing 123 - path.js

```

1 /**
2 * @file JSLAB library presentation submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB presentation submodule.
10 */
11 class PRDC_JSLAB_LIB_PRESENTATION {
12
13 /**
14 * Constructs a presentation submodule object with access to JSLAB's
15 * functions.
16 * @constructor
17 * @param {Object} jsl - Reference to the main JSLAB object.
18 */
19 constructor(jsl) {
20     var obj = this;
21     this.jsl = jsl;
22 }
23
24 /**
25 * Opens an existing presentation in a new window and returns its context.
26 * @param {String} file_path - Absolute or relative path to the presentation

```

```

    directory.
26 * @param {String} type - Type of presentation.
27 * @returns {Promise<Window>} Resolves to the window context of the opened
28   presentation.
29 */
30 async openPresentation(file_path, type) {
31   file_path = this._getPath('openPresentation', file_path);
32   if(this._checkPresentation('editPresentation', file_path)) {
33     var obj = this;
34     var name = this.jsl.env.pathBaseName(file_path);
35     if(type === 'standalone') {
36       var url = this.jsl.env.pathJoin(file_path, 'index.html')
37       var wid = this.jsl.windows.openWindow(url);
38     } else {
39       var url = await this._startPresentation(this.jsl.env.pathJoin(
40         file_path, name + '.exe'));
41       var wid = this.jsl.windows.openWindow('url.html');
42     }
43     await this.jsl.windows.open_windows[wid].ready;
44     var context = this.jsl.windows.open_windows[wid].context;
45     var fullscreen = false;
46     if(type === 'standalone') {
47       while(typeof context.presentation === 'undefined') {
48         await this.jsl.non_blocking.waitMSeconds(1);
49     }
50     context.document.addEventListener('keydown', (event) => {
51       if(event.key === 'F11') {
52         fullscreen = !fullscreen;
53         obj.jsl.windows.open_windows[wid].setFullscreen(fullscreen);
54       }
55     });
56   } else {
57     context.document.getElementById('webview').src = url;
58     context.webview.addEventListener('ipc-message', (e) => {
59       if(e.args[0].key !== undefined) {
60         if(e.args[0].key === 'F11') {
61           fullscreen = !fullscreen;
62           obj.jsl.windows.open_windows[wid].setFullscreen(fullscreen);
63         }
64       }
65     });
66   }
67   this.jsl.windows.open_windows[wid].setTitle(file_path + ' - Presentation
68   - JSLAB | PR-DC');
69   return context;
70 }
71 /**
72 * Opens the presentation editor for the specified project and returns its
73   context.
74 * @async
75 * @param {String} file_path - Absolute or relative path to the presentation
76   directory.
77 * @param {String} type - Type of presentation.

```

```

75   * @returns {Promise<Window>} Resolves to the window context of the editor
76   * @param {String} file_path - Path to the presentation file.
77   */
78   async editPresentation(file_path, type) {
79     file_path = this._getPath('editPresentation', file_path);
80     if(this._checkPresentation('editPresentation', file_path)) {
81       var name = this.jsl.env.pathBaseName(file_path);
82       if(type === 'standalone') {
83         var url = this.jsl.env.pathJoin(file_path, 'index.html')
84       } else {
85         var url = await this._startPresentation(this.jsl.env.pathJoin(
86           file_path, name + '.exe'));
87       }
88       var wid = this.jsl.windows.openWindow('presentation-editor.html');
89       await this.jsl.windows.open_windows[wid].ready;
90       var context = this.jsl.windows.open_windows[wid].context;
91       while(typeof context.presentation_editor === 'undefined') {
92         await this.jsl.non_blocking.waitMSeconds(1);
93       }
94       context.presentation_editor.setPath(file_path, url);
95       this.jsl.windows.open_windows[wid].setTitle(file_path + ' - Presentation
96       editor - JSLAB | PR-DC');
97       return context;
98     }
99   }
100 /**
101 * Creates a new presentation project on disk and optionally opens it in the
102 * editor.
103 * @param {String} file_path - Target directory where the presentation
104 * project will be created.
105 * @param {Object} [opts_in] Extra options
106 * @param {Boolean} [open_editor=true] - If true, automatically opens the
107 * new project in the editor.
108 */
109 createPresentation(file_path, opts_in = {}, open_editor = true) {
110   file_path = this._getPath('createPresentation', file_path);
111   var name = this.jsl.env.pathBaseName(file_path);
112   var presentation_config = {
113     "jslab_version": this.jsl.context.version,
114     "slide_width": 1920,
115     "slide_height": 1080,
116     ...opts_in
117   }
118   this.jsl.env.makeDirectory(file_path);
119   this.jsl.env.makeDirectory(this.jsl.env.pathJoin(file_path, 'res/'));
120   this.jsl.env.makeDirectory(this.jsl.env.pathJoin(file_path, 'res/internal/'));
121
122   var js = this.jsl.env.readFileSync(this.jsl.env.pathJoin(app_path, 'js/
123     windows/presentation.js')).toString();
124   js = js.replace('%presentation_config%', JSON.stringify(
125     presentation_config, false, 2));
126   this.jsl.file_system.writeFile(this.jsl.env.pathJoin(file_path, 'res/
127     internal/presentation.js'), js);

```

```

120
121     this.jsl.file_system.copyFile(this.jsl.env.pathJoin(app_path, 'lib/
122         portable_server/portable_server.exe'),
123     this.jsl.env.pathJoin(file_path, name + '.exe'));
124     this.jsl.file_system.copyFile(this.jsl.env.pathJoin(app_path, 'css/
125         presentation.css'),
126     this.jsl.env.pathJoin(file_path, 'res/internal/presentation.css'));
127
128     this.jsl.file_system.writeFile(this.jsl.env.pathJoin(file_path, 'main.css'
129         ), '');
130     this.jsl.file_system.writeFile(this.jsl.env.pathJoin(file_path, 'main.js'
131         ), '');
132     this.jsl.file_system.writeFile(this.jsl.env.pathJoin(file_path, 'res/
133         internal/config.json'), JSON.stringify(presentation_config, false, 2));
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160

```

```

161     this.jsl.file_system.copyFile(this.jsl.env.pathJoin(app_path, 'js/
  windows/mathjax-config.js'), this.jsl.env.pathJoin(file_path,
  res/mathjax-config.js));
162     this.jsl.file_system.copyFolder(this.jsl.env.pathJoin(app_path, 'lib/
  /tex-mml-ctml-3.2.0'), this.jsl.env.pathJoin(file_path, 'res/
  tex-mml-ctml-3.2.0));
163     presentation_scripts += '
164     <script type="text/javascript" src="./res/mathjax-config.js"></script>
165     <script type="text/javascript" src="./res/tex-mml-ctml-3.2.0/tex-mml-ctml
  -3.2.0.js"></script>
166   ';
167   } else if(module == 'scene-3d-json') {
168     this.jsl.file_system.copyFolder(this.jsl.env.pathJoin(app_path, 'lib/
  /three.js-r162'), this.jsl.env.pathJoin(file_path, 'res/three.js-
  r162));
169     presentation_scripts += '
170     <script type="importmap">
171       {
172         "imports": {
173           "three": "./res/three.js-r162/build/three.module.js",
174           "three/addons/": "./res/three.js-r162/examples/jsm/"
175         }
176       }
177     </script>
178     <script type="module">
179       import * as THREE from 'three';
180       window.THREE = THREE;
181     </script>
182   ';
183   }
184   }
185 }
186 var html = this.jsl.env.readFileSync(this.jsl.env.pathJoin(app_path, 'html/
  /presentation.html')).toString();
187 html = html.replace('%presentation_scripts%', presentation_scripts);
188 html = html.replace('%presentation_stylesheets%', presentation_stylesheets
  );
189 this.jsl.file_system.writeFile(this.jsl.env.pathJoin(file_path, 'index.
  html'), html);
190
191 if(open_editor) {
192   this.editPresentation(file_path);
193 }
194
195 /**
196 * Packages an existing presentation directory into a ZIP archive beside it.
197 * @param {String} file_path - Path to the presentation directory to be
198   archived.
199 */
200 packPresentation(file_path) {
201   file_path = this._getPath('packPresentation', file_path);
202   if(this._checkPresentation('packPresentation', file_path)) {
203     var dest = this.jsl.env.pathResolve(this.jsl.env.pathJoin(file_path, '..',
  this.jsl.env.pathBaseName(file_path) + '.zip'));

```

```

204     this.jsl.env.execSync(`"${this.jsl.env.bin7zip}" a -tzip "${dest}" "${
205       this.jsl.env.pathJoin(file_path, '*')}"`);
206   this.jsl.env disp('@packPresentation: ' + language.string(241) + dest);
207 }
208
209 /**
210 * Converts an existing presentation to standalone presentation.
211 * @param {String} file_path - Path to the presentation directory to be
212   archived.
213 */
214 makeStandalonePresentation(file_path) {
215   file_path = this._getPath('makeStandalonePresentation', file_path);
216   if(this._checkPresentation('makeStandalonePresentation', file_path)) {
217     var html_file = this.jsl.env.pathJoin(file_path, 'index.html');
218     var html = this.jsl.env.readFileSync(html_file).toString();
219     var reImagePdf = /<img-pdf\b[^>]*?\bsrc\s*=\s*["]([^\"]+)[^"]*\">/gi;
220     var rePlotJson = /<plot-json\b[^>]*?\bsrc\s*=\s*["]([^\"]+)[^"]*\">/gi;
221     var reScene3dJson = /<scene-3d-json\b[^>]*?\bsrc\s*=\s*["]([^\"]+)[^"]*
222       ][^>]*>/gi;
223     var assets = new Set();
224     for(var re of [reImagePdf, rePlotJson, reScene3dJson]) {
225       let m;
226       while((m = re.exec(html)) !== null) {
227         var rel = m[1].trim();
228         if(!rel) continue;
229         assets.add(rel);
230       }
231     }
232     for(var file of [...assets]) {
233       this._fileToBuffer(file_path, file);
234     }
235   }
236 }
237
238 /**
239 * Converts an presentation to PDF format.
240 * @param {String} file_path - Path to the presentation directory.
241 * @param {Boolean} run_make_standalone - Whether to run
242   makeStandalonePresentation method or not.
243 */
244 async presentationToPdf(file_path, run_make_standalone = true) {
245   file_path = this._getPath('presentationToPdf', file_path);
246   if(this._checkPresentation('presentationToPdf', file_path)) {
247     if(run_make_standalone) {
248       this.makeStandalonePresentation(file_path);
249     }
250     var win = await this.openPresentation(file_path, 'standalone');
251     var p = win.presentation;
252     p._interpolateAllSlides();
253     win.setSize(p.config.slide_width, p.config.slide_height);

```

```

254     await waitMSeconds(200);
255     while(typeof win.presentation == 'undefined') {
256         await this.jsl.non_blocking.waitMSeconds(1);
257     }
258     win.setOpacity(0);
259     for(var i = 0; i < win.presentation.total_slides; i++) {
260         win.presentation.setSlide(i);
261         await this._waitForSlide(win, win.presentation.slides[i]);
262         await waitMSeconds(1);
263     }
264     win.presentation.setSlide(0);
265     var pdf = await this.jsl.windows.printWindowToPdf(win.wid, {
266         margins: { top: 0, right: 0, bottom: 0, left: 0 },
267         printBackground: true,
268         landscape: false,
269         pageSize: {
270             width: p.config.slide_width / 96,
271             height: p.config.slide_height / 96
272         }
273     });
274     var name = this.jsl.env.pathBaseName(file_path);
275     var dest = this.jsl.env.pathJoin(file_path, name + '.pdf');
276     this.jsl.file_system.writeFileSync(dest, pdf);
277     win.close();
278     this.jsl.env.disp('@presentationToPdf: ' + language.string(244) + dest);
279   }
280 }
281
282 /**
283 * Waits for slide elements to be loaded
284 * @param {Window} win - Window context with presentation.
285 * @param {HTMLElement} slide - HTML element of slide.
286 */
287 async _waitForSlide(win, slide) {
288   var img_pdfs = Array.from(slide.querySelectorAll('img-pdf'));
289   var plot_jsons = Array.from(slide.querySelectorAll('plot-json'));
290   var scene_3d_jsons = Array.from(slide.querySelectorAll('scene-3d-json'));
291   for(var e of [...img_pdfs, ...plot_jsons]) {
292     while(!e._finished_loading) {
293       await waitMSeconds(1);
294     }
295   }
296   var videos = Array.from(slide.querySelectorAll('video'));
297   for(var v of videos) {
298     await this._waitForVideo(v);
299   }
300   await this._replaceCanvases(win, slide);
301 }
302
303 /**
304 * Waits for video elements to be loaded
305 * @param {HTMLElement} slide - HTML element of video.
306 */
307 async _waitForVideo(video) {

```

```

309   if(video.preload === 'none') {
310     video.preload = 'auto';
311     video.load();
312   }
313
314   await new Promise(resolve => {
315     if(video.readyState >= HTMLMediaElement.HAVE_CURRENT_DATA) {
316       resolve();
317     } else {
318       video.addEventListener('loadeddata', resolve, { once: true });
319     }
320   });
321
322   video.controls = false;
323   video.muted = true;
324   try { await video.play(); } catch {};
325   video.pause();
326
327   if('requestVideoFrameCallback' in video) {
328     await new Promise(res => video.requestVideoFrameCallback(() => res()));
329   } else {
330     if(video.currentTime === 0) {
331       video.currentTime = 0.05;
332       video.currentTime = 0;
333     }
334   }
335 }
336
337 /**
338 * Replaces all canvases with static images.
339 * @param {Window} win - Window context with presentation.
340 * @param {HTMLElement} slide - HTML element of slide.
341 */
342 async _replaceCanvases(win, slide) {
343   var plot_divs = Array.from(slide.querySelectorAll('.js-plotly-plot'))
344   .filter(div => div.querySelector('canvas'));
345   for(var plot_div of plot_divs) {
346     var data_url;
347     try {
348       data_url = await win.Plotly.toImage(plot_div, {
349         format : 'png',
350         width : plot_div.clientWidth * win.devicePixelRatio,
351         height : plot_div.clientHeight * win.devicePixelRatio
352       });
353     } catch(err) {}
354
355   var img = new win.Image();
356   img.src = data_url;
357   img.style.width = (plot_div.style.width || plot_div.width + 'px');
358   img.style.height = (plot_div.style.height || plot_div.height + 'px');
359   img.style.maxWidth = '100%';
360   img.style.maxHeight = '100%';
361   plot_div.parentNode.replaceChild(img, plot_div);
362   await waitMSeconds(1);
363 }

```

```
364     }
365
366     /**
367      * Resolves and returns a presentation directory path, prompting the user if
368      * necessary.
369      * @param {String} method - Name of the calling method for error reporting.
370      * @param {String} [file_path] - Candidate path supplied by the caller.
371      * @returns {((String|false))} Resolved presentation directory path or false
372      * if cancelled.
373      */
374     _getPath(method, file_path) {
375       if(!file_path) {
376         var options = {
377           title: language.currentString(239),
378           buttonLabel: language.currentString(231),
379           properties: [ 'openDirectory' ],
380         };
381         file_path = this.jsl.env.showOpenDialogSync(options);
382         if(file_path === undefined) {
383           this.jsl.env.error('@' + method + ': '+language.string(119)+'.');
384           return false;
385         } else {
386           file_path = file_path[0];
387         }
388       }
389       return file_path;
390     }
391
392     /**
393      * Checks whether the supplied directory contains a valid presentation
394      * structure.
395      * @param {String} method - Name of the calling method for error reporting.
396      * @param {String} file_path - Path to the presentation directory to
397      * validate.
398      * @returns {Boolean} True if the directory contains a 'index.html' file,
399      * otherwise false.
400      */
401     _checkPresentation(method, file_path) {
402       if(!this.jsl.file_system.existFile(this.jsl.env.pathJoin(file_path, 'index
403         .html'))){
404         this.jsl.env.error('@' + method + ': '+language.string(240));
405         return false;
406       }
407       return true;
408     }
409
410     /**
411      * Starts the portable HTTP server that serves the presentation and
412      * resolves once the server prints the listening URL.
413      * @param {String} exe_file - Absolute path to the portable server
414      * executable.
415      * @returns {Promise<String>} Resolves to the presentation URL.
416      */
417     _startPresentation(exe_file) {
418       var obj = this;
```

```

412     return new Promise((resolve, reject) => {
413       const child = obj.jsl.env.spawn(exe_file, [ '--prog' ], {
414         stdio: [ 'ignore', 'pipe', 'inherit' ],
415         windowsHide: true
416       });
417
418       let buffer = '';
419       child.stdout.setEncoding('utf8');
420       child.stdout.on('data', chunk => {
421         buffer += chunk;
422         const nl = buffer.indexOf('\n');
423         if(nl !== -1) {
424           let line = buffer.slice(0, nl).replace(/\r$/,'').trim();
425           child.stdout.removeAllListeners('data');
426           let url = line.replace(/^s*url:/i,'');
427           resolve(url);
428         }
429       });
430
431       child.once('error', reject);
432       child.once('exit', code => {
433         reject(`Server exited early with code ${code}`);
434       });
435     });
436   }
437
438 /**
439 * Convert file to JavaScript base64 buffer.
440 * @param {String} file_path - Path to presentation.
441 * @param {String} rel - Path to file.
442 */
443 _fileToBuffer(file_path, rel) {
444   var abs = this.jsl.env.pathResolve(this.jsl.env.pathJoin(file_path, rel));
445   var bin = this.jsl.env.readFileSync(abs);
446   var b64 = bin.toString('base64');
447   var name = encodeURIComponent(rel.replace(/\//g, "/"));
448   this.jsl.file_system.writeFile(abs + '.buf.js',
449     `registerFile("${name}", "${b64}")`);
450 }
451 }
452
453 exports.PRDC_JSLAB_LIB_PRESENTATION = PRDC_JSLAB_LIB_PRESENTATION;

```

Listing 124 - presentation.js

```

1 /**
2  * @file JSLAB library render submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB render submodule.
10 */
11 class PRDC_JSLAB_LIB_RENDER {

```

```
12
13  /**
14   * Initializes a new instance of the render submodule.
15   * @param {Object} js1 Reference to the main JSLAB object.
16   */
17 constructor(js1) {
18     var obj = this;
19     this.js1 = js1;
20 }
21
22 /**
23  * Debounces a function, ensuring it's only invoked once at the beginning of
24  * consecutive calls during the wait period.
25  * @param {Function} func - The function to debounce.
26  * @param {number} wait - The period to wait before allowing another call,
27  *   in milliseconds.
28  * @returns {Function} The debounced function.
29  */
30 debounceIn(func, wait) {
31   var timeout;
32   let last_args;
33
34   return function(...args) {
35     var context = this;
36     last_args = args;
37     var later = function() {
38       timeout = null;
39     };
40     if (!timeout) {
41       timeout = setTimeout(later, wait);
42       func.apply(context, last_args);
43     }
44   };
45 }
46 /**
47  * Debounces a function, calling it at the first and last of consecutive
48  * calls during the wait period.
49  * @param {Function} func - The function to debounce.
50  * @param {number} wait - The period to wait before allowing another call,
51  *   in milliseconds.
52  * @returns {Function} The debounced function.
53  */
54 debounceInOut(func, wait) {
55   var timeout;
56   var hit = false;
57   let last_args;
58
59   return function(...args) {
60     var context = this;
61     last_args = args;
62     var later = function() {
63       if (hit) {
64         hit = false;
65         func.apply(context, last_args);
66       }
67     };
68   };
69 }
```

```

63         setTimeout(later, wait);
64     } else {
65         timeout = null;
66     }
67 };
68 if (!timeout) {
69     timeout = setTimeout(later, wait);
70     func.apply(context, last_args);
71 } else {
72     hit = true;
73 }
74 };
75 }

77 /**
78 * Debounces a function, ensuring it's only invoked once at the end of
79 * consecutive calls during the wait period.
80 * @param {Function} func - The function to debounce.
81 * @param {number} wait - The period to wait before allowing another call,
82 *           in milliseconds.
83 * @returns {Function} The debounced function.
84 */
85 debounceOut(func, wait) {
86     var timeout;
87     let last_args;
88
89     return function(...args) {
90         var context = this;
91         last_args = args;
92         var later = function() {
93             timeout = null;
94             func.apply(context, last_args);
95         };
96         if (!timeout) {
97             timeout = setTimeout(later, wait);
98         }
99     };
100 }
101 exports.PRDC_JSLAB_LIB_RENDER = PRDC_JSLAB_LIB_RENDER;

```

Listing 125 - render.js

```

1 /**
2  * @file JSLAB library serial device submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB serial device submodule.
10 */
11 class PRDC_JSLAB_LIB_SERIAL_DEVICE {
12

```

```

13  /**
14   * Constructs a serial device submodule object with access to JSLAB's device
15   * functions.
16   * @constructor
17   * @param {Object} js1 - Reference to the main JSLAB object.
18   */
19  constructor(js1) {
20    var obj = this;
21    this.js1 = js1;
22  }
23  /**
24   * Retrieves all available serial ports.
25   * @returns {Promise<Array>} Resolves with an array of serial port info.
26   */
27  async listSerialPorts() {
28    return await this.js1.env.SerialPort.list();
29  }
30  /**
31   * Checks if there is a USB device connected with the specified Vendor ID
32   * and Product ID.
33   * @param {string} VID - Vendor ID of the USB device.
34   * @param {string} PID - Product ID of the USB device.
35   * @returns {boolean} True if the device is found, false otherwise.
36   */
37  async checkDeviceUSB(VID, PID){
38    var devices = await this.listSerialPorts();
39    if(Array.isArray(devices)) {
40      if(this.js1.debug) {
41        this.js1.env.disp(`@checkDeviceUSB: ${JSON.stringify(devices)}`);
42      }
43      return devices.some(device => device.vendorId === VID && device.
44                      productId === PID);
45    }
46    return false;
47  }
48  /**
49   * Checks for a connected USB device by STM and an optional Product ID.
50   * @param {string} [PID='5740'] - Product ID of the USB device, default is
51   * for Virtual COM Port.
52   * @returns {boolean} True if the device is found, false otherwise.
53   */
54  async checkDeviceSTM(PID = '5740') {
55    return await this.checkDeviceUSB('0483', PID);
56  }
57  /**
58   * Checks if there is a USB device connected using a CH340 chip.
59   * @returns {boolean} True if the device is found, false otherwise.
60   */
61  async checkDeviceCH340() {
62    return await this.checkDeviceUSB('1A86', '7523');
63  }

```

```

64
65  /**
66   * Opens a serial port.
67   * @param {string} port - Port path.
68   * @param {number} [baudrate=9600] - Baud rate.
69   * @param {object} [opts={}] - Additional options.
70   * @returns {SerialPort} The opened SerialPort instance.
71   */
72 connectSerialPorts(port_path, baudrate = 9600, opts_in = {}) {
73   var opts = {
74     dataBits: 8,
75     parity: 'none',
76     stopBits: 1,
77     flowControl: false,
78     ...opts_in
79   }
80   var port = new this.jsl.env.SerialPort({
81     path: port_path,
82     baudRate: baudrate,
83     ...opts
84 });
85   port.on('open', function() {
86     try {
87       port.set({
88         dtr: true,
89         rts: false
90       });
91     } catch(err) {}
92   });
93
94   this.jsl.addForCleanup(this, function() {
95     if(port && port.isOpen) {
96       port.close();
97     }
98   });
99
100  return port;
101}
102
103 /**
104  * Opens a window to choose a serial port.
105  * @returns {Promise<string|false>}
106  */
107 async chooseSerialPort() {
108   var context = await this.jsl.windows.openWindowBlank();
109   context.setTitle('Choose serial port');
110   var ports = await this.listSerialPorts();
111   var ports_list = '';
112   if(ports.length > 0) {
113     for(var port of ports) {
114       ports_list += '<li class="ui" port=' + port.path + '>' + port.friendlyName
115       + '</li>';
116     }
117   } else {
118     ports_list += '<li class="ui"><str sid="230"></str></li>';
119   }
120   return ports_list;
121 }
122
123 
```

```

118 }
119 context.document.body.innerHTML = '<ul class="ui">$ports_list</ul>';
120 var win = this.jsl.windows.open_windows[context.wid];
121 win._updateLanguage();
122 win.addUI();
123 context.setSize(300, 100);
124 await this.jsl.non_blocking.waitMSeconds(30);
125 var win_height = context.document.body.offsetHeight;
126 context.setSize(300, (win_height > 500 ? 500 : win_height));
127
128
129 if(ports.length == 0) {
130   return false;
131 } else {
132   var returned = false;
133   var p = new Promise((resolve, reject) => {
134     context.document.querySelectorAll('li').forEach(function(li) {
135       li.addEventListener('click', function(e) {
136         if(!returned) {
137           returned = true;
138           resolve(this.getAttribute('port'));
139           context.close();
140         }
141       });
142     });
143     win.onClosed = function() {
144       if(!returned) {
145         returned = true;
146         resolve(false);
147       }
148     });
149   });
150   return await p;
151 }
152
153
154 /**
155 * Opens a window to choose serial options.
156 * @returns {Promise<string|false>}
157 */
158
159 async chooseSerialOptions() {
160   var context = await this.jsl.windows.openWindowBlank();
161   context.setTitle('Choose serial options');
162   var ports = await this.listSerialPorts();
163   var ports_list = '';
164   if(ports.length > 0) {
165     for(var port of ports) {
166       ports_list += '<option value=' + port.path + '>' + port.friendlyName + '</';
167       option>';
168     }
169   } else {
170     ports_list += "<option value=''" + str='230' + '></option>"';
171   }
172   context.document.body.innerHTML =

```

```

172   <label class="ui"><str sid="232"></str></label>
173   <select class="ui" id="port">${
174     ports_list
175   }</select>
176   <label class="ui"><str sid="233"></str></label>
177   <input class="ui" type="text" id="baudrate" str="233"></input>
178   <button class="ui blue" id="choose"><str sid="231"></str></button>;
179   var win = this.jsl.windows.open_windows[context.wid];
180   win._updateLanguage();
181   win.addUI();
182   context.setSize(400, 100);
183   await this.jsl.non_blocking.waitMSeconds(30);
184   var win_height = context.document.body.offsetHeight;
185   context.setSize(400, (win_height > 500 ? 500 : win_height));
186
187   if(ports.length == 0) {
188     warn('@chooseSerialOptions: '+language.currentString(230));
189     context.close();
190     return [false, false];
191   } else {
192     var returned = false;
193     var p = new Promise((resolve, reject) => {
194       context.document.getElementById('choose').addEventListener('click', function(e) {
195         if(!returned) {
196           returned = true;
197           resolve([context.document.getElementById('port').value,
198                   Number(context.document.getElementById('baudrate').value)]);
199           context.close();
200         }
201       });
202       win.onClosed = function() {
203         if(!returned) {
204           returned = true;
205           resolve([false, false]);
206         }
207       });
208     return await p;
209   }
210 }
211
212 /**
213 * Opens a serial terminal.
214 * @param {string} port_path - The identifier or path of the serial port to
215 * connect to.
216 * @param {number} [baudrate] - The communication speed in bits per second.
217 * @param {Object} [opts={}] - An optional configuration object for
218 * additional settings.
219 * @returns {Promise<Object>} A promise that resolves with the terminal
220 * context.
221 */
222 async openSerialTerminal(port_path, baudrate = 9600, opts = {}) {
223   var obj = this;
224
225   if(!baudrate) {
226     baudrate = 9600;

```

```

224 }
225
226 this.port_path = port_path;
227 this.baudrate = baudrate;
228 this.opts = opts;
229
230 var wid = this.jsl.windows.openWindow('serial_terminal.html');
231 var win = this.jsl.windows.open_windows[wid];
232 await win.ready;
233 win.addUI();
234 win.addScript("../js/windows/terminal.js");
235 var context = win.context;
236 context.setTitle(port_path + ' (' + baudrate + ') | Serial Terminal');
237 context.setSize(500, 500);
238 context.document.getElementById('terminal-title').innerText =
239   port_path + ' (' + baudrate + ')';
240 win._updateLanguage();
241
242 // Wait for terminal
243 while (!context.terminal) {
244   await waitMSeconds(1);
245 }
246
247 context.terminal.setOptions(opts);
248
249 win.onClosed = function() {
250   if(obj.port && obj.port.isOpen) {
251     obj.port.close();
252   }
253   delete obj.port;
254 }
255
256 this.port = this.connectSerialPorts(port_path, baudrate);
257
258 this.port.on('data', (data) => {
259   var data_in = data;
260   if(obj.opts && typeof obj.opts.decodeData == 'function') {
261     data_in = obj.opts.decodeData(data);
262   } else {
263     data_in = data.toString('utf8');
264   }
265   if(data_in !== false) {
266     if(Array.isArray(data_in) && data_in.length) {
267       for(var message of data_in) {
268         context.terminal.addMessage(' ', message);
269       }
270     } else {
271       context.terminal.addMessage(' ', data_in);
272     }
273   }
274 });
275
276 function sendMessage() {
277   var data_out_raw = context.terminal.message_input.value;
278   var data_out;

```

```

279     if (obj.opts && typeof obj.opts.encodeData === 'function') {
280         data_out = obj.opts.encodeData(data_out_raw);
281     } else {
282         data_out = data_out_raw;
283     }
284     if (data_out !== false) {
285         obj.port.write(data_out);
286         context.terminal.message_input.value = '';
287         context.terminal.autoResizeInput();
288     }
289 }
290
291 context.terminal.message_input.addEventListener('keydown', function(e) {
292     if (e.key === 'Enter' && !e.shiftKey) {
293         e.preventDefault();
294         sendMessage();
295     }
296 });
297 context.document.getElementById('send-button')
298     .addEventListener('click', function(e) {
299         sendMessage();
300     });
301
302 context.document.getElementById('timestamp').addEventListener('click', () => {
303     setTimeout(function() {
304         win._updateLanguage();
305     }, 30);
306 });
307 context.document.getElementById('autoscroll').addEventListener('click', () => {
308     setTimeout(function() {
309         win._updateLanguage();
310     }, 30);
311 });
312
313 function saveLog() {
314     let options = {
315         title: language.currentString(58),
316         defaultPath: 'terminal_' + obj.jsl.time.getDateTimeStr() + '.log',
317         buttonLabel: language.currentString(58),
318         filters : [
319             {name: 'Log', extensions: ['log', 'txt']},
320             {name: 'All Files', extensions: ['*']}
321         ]
322     };
323     var log_path = obj.jsl.env.showSaveDialogSync(options);
324     if(log_path) {
325         var data = context.terminal.getLog();
326         obj.jsl.env.writeFileSync(log_path, data);
327     }
328 }
329 context.document.getElementById('save-log').addEventListener('click', () => {
330     saveLog();

```

```

331     });
332
333     return context;
334 }
335
336 /**
337 * Prompts the user to choose serial options and opens a serial terminal if
338 * a valid port is selected.
339 * @param {Object} [opts={}] - An optional configuration object for
340 * additional settings.
341 * @returns {Promise<Object|undefined>} A promise that resolves with the
342 * terminal context if a serial port is chosen.
343 */
344 async chooseSerialTerminal(opts) {
345   var [port, baudrate] = await this.chooseSerialOptions();
346   if(port) {
347     return await this.openSerialTerminal(port, baudrate, opts);
348   }
349 }
350
351 exports.PRDC_JSLAB_LIB_SERIAL_DEVICE = PRDC_JSLAB_LIB_SERIAL_DEVICE;

```

Listing 126 - serial-device.js

```

1 /**
2 * @file JSLAB library symbolic submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB symbolic submodule.
10 */
11 class PRDC_JSLAB_SYMBOLIC_MATH {
12
13 /**
14 * Constructs a symbolic submodule object with access to JSLAB's symbolic
15 * functions.
16 * @constructor
17 * @param {Object} jsl - Reference to the main JSLAB object.
18 */
19 constructor(jsl) {
20   var obj = this;
21   this.jsl = jsl;
22
23   this.loaded = false;
24
25   this._var_counter = 0;
26   this._symbols = [];
27 }
28
29 /**
30 * Loads the symbolic math libraries (SymPy and NumPy) using Pyodide.
31 * Initializes the Python environment for symbolic computations.
32 */

```

```

31   * @returns {Promise<void>} A promise that resolves when the libraries are
32   * loaded.
33   */
34   async load() {
35     if(!this.loaded) {
36       this.pyodide = await loadPyodide({
37         indexURL: app_path+'lib/sympy-0.26.2/',
38       });
39       await this.pyodide.loadPackage(['sympy', 'numpy'], { messageCallback: () => {}});
40       this.loaded = true;
41       this.pyodide.runPython('globals().clear()');
42       this.pyodide.runPython(`
43         from sympy import *
44         import numpy as np
45       `);
46     }
47   }
48 /**
49  * Generates the next unique variable name for symbolic expressions.
50  * @returns {string} The next unique variable name (e.g., 'jslabVar1').
51  */
52 _nextVar() {
53   this._var_counter += 1;
54   return 'jslabVar'+this._var_counter;
55 }
56 /**
57  * Creates a new symbolic variable with an optional name and value.
58  * @param {string} [name] - The name of the symbolic variable. If undefined,
59  *   a unique name is generated.
60  * @param {*} [value] - The initial value of the symbolic variable.
61  * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The newly created symbolic
62  *   variable.
63  */
64 _newSymbol(name, value) {
65   if(typeof name === 'undefined') {
66     name = this._nextVar();
67   }
68   var symbol = new PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL(name, value);
69   this._symbols.push(symbol);
70   return symbol;
71 }
72 /**
73  * Retrieves the name of a symbolic variable.
74  * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL| string} symbol - The symbolic
75  *   variable or its name.
76  * @returns {string} The name of the symbolic variable.
77  */
78 getSymbolName(symbol) {
79   if(typeof symbol === 'object' && symbol.constructor.name === 'PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL') {
80     return symbol.name;

```

```

80     } else {
81         return symbol;
82     }
83 }
84
85 /**
86 * Checks if the symbolic libraries are loaded. Throws an error if not.
87 * @throws {Error} If the symbolic libraries are not loaded.
88 */
89 checkLoaded() {
90     if(!this.loaded) {
91         this.jsl.env.error('@sym: '+language.string(175));
92     }
93 }
94
95 /**
96 * Evaluates a Python code string within the symbolic math environment.
97 * @param {string} code - The Python code to evaluate.
98 * @returns {*} The result of the evaluated code.
99 * @throws {Error} If there is an error during code evaluation.
100 */
101 eval(code) {
102     this.checkLoaded();
103     if(config.DEBUG_SYM_PYTHON_EVAL_CODE) {
104         this.jsl._console.log('@sym: eval: ' + code);
105     }
106     try {
107         return this.pyodide.runPython(code);
108     } catch(err) {
109         this.jsl.env.error('@sym: ' + err);
110     }
111     return false;
112 }
113
114 /**
115 * Creates a single symbolic variable.
116 * @param {string} name - The name of the symbolic variable.
117 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The created symbolic variable.
118 */
119 sym(name) {
120     this.checkLoaded();
121
122     this.eval(` ${name} = Symbol('${name}') `);
123     return this._newSymbol(name, name);
124 }
125
126 /**
127 * Creates multiple symbolic variables.
128 * @param {string[]} names - An array of names for the symbolic variables.
129 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL[]} An array of created symbolic
130 * variables.
131 */
132 syms(names) {
133     var obj = this;
134     this.checkLoaded();

```

```

134
135   this.eval(` ${names.join(' ', ')} = symbols(` ${names.join(' ', ')} `)` );
136
137   var symbols = [];
138   names.forEach(function(name) {
139     symbols.push(obj._newSymbol(name, name));
140   });
141   return symbols;
142 }
143
144 /**
145 * Creates a symbolic matrix from a nested array expression.
146 * @param {Array<Array<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number>>} expr
147   - The nested array representing the matrix.
148 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The symbolic matrix.
149 */
150 mat(expr) {
151   this.checkLoaded();
152
153   expr = JSON.stringify(expr.map(row => row.map(el => this.getSymbolName(el))));
154
155   var symbol = this._newSymbol();
156   symbol.setValue(this.eval(`
157     ${symbol.name} = Matrix(${expr})
158     ${symbol.name}
159   `));
160   return symbol;
161 }
162 /**
163 * Multiplies multiple symbolic expressions.
164 * @param {...PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} args - The
165   symbolic expressions to multiply.
166 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The resulting symbolic
167   expression after multiplication.
168 */
169 mul(...args) {
170   var obj = this;
171   this.checkLoaded();
172
173   var expr = args
174     .map(function(item) {
175       return obj.getSymbolName(item);
176     }).join(' * ');
177
178   var symbol = this._newSymbol();
179   symbol.setValue(this.eval(`
180     ${symbol.name} = ${expr}
181     ${symbol.name}
182   `));
183   return symbol;
184 }
185 /**

```

```

185  * Divides multiple symbolic expressions.
186  * @param {...PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} args - The
187  *   symbolic expressions to divide.
188  * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The resulting symbolic
189  *   expression after division.
190  */
191 div(...args) {
192   var obj = this;
193   this.checkLoaded();
194
195   var expr = args
196     .map(function(item) {
197       return obj.getSymbolName(item);
198     }).join(' / ');
199
200   var symbol = this._newSymbol();
201   symbol.setValue(this.eval(`
202     ${symbol.name} = ${expr}
203     ${symbol.name}
204   `));
205   return symbol;
206 }
207
208 /**
209  * Adds multiple symbolic expressions.
210  * @param {...PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} args - The
211  *   symbolic expressions to add.
212  * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The resulting symbolic
213  *   expression after addition.
214  */
215 plus(...args) {
216   var obj = this;
217   this.checkLoaded();
218
219   var expr = args
220     .map(function(item) {
221       return obj.getSymbolName(item);
222     }).join(' + ');
223
224   var symbol = this._newSymbol();
225   symbol.setValue(this.eval(`
226     ${symbol.name} = ${expr}
227     ${symbol.name}
228   `));
229   return symbol;
230 }
231
232 /**
233  * Subtracts multiple symbolic expressions.
234  * @param {...PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} args - The
235  *   symbolic expressions to subtract.
236  * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The resulting symbolic
237  *   expression after subtraction.
238  */
239 minus(...args) {

```

```

234     var obj = this;
235     this.checkLoaded();
236
237     var expr = args
238       .map(function(item) {
239         return obj.getSymbolName(item);
240       }).join(' - ');
241
242     var symbol = this._newSymbol();
243     symbol.setValue(this.eval(`
244       ${symbol.name} = ${expr}
245       ${symbol.name}
246     `));
247     return symbol;
248   }
249
250 /**
251 * Raises a symbolic expression to a power.
252 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} expr - The base
253 * expression.
254 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number} n - The exponent.
255 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The resulting symbolic
256 * expression after exponentiation.
257 */
258 pow(expr, n) {
259   this.checkLoaded();
260
261   expr = this.getSymbolName(expr);
262   n = this.getSymbolName(n);
263
264   var symbol = this._newSymbol();
265   symbol.setValue(this.eval(`
266     ${symbol.name} = ${expr}**${n}
267     ${symbol.name}
268   `));
269   return symbol;
270 }
271
272 /**
273 * Transposes a symbolic matrix expression.
274 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The matrix
275 * expression to transpose.
276 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The transposed matrix
277 * expression.
278 */
279 transp(expr) {
280   this.checkLoaded();
281
282   expr = this.getSymbolName(expr);
283
284   var symbol = this._newSymbol();
285   symbol.setValue(this.eval(`
286     ${symbol.name} = ${expr}.T
287     ${symbol.name}
288   `));

```

```

285     return symbol;
286 }
287
288 /**
289 * Computes the inverse of a symbolic matrix expression.
290 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The matrix
291 *   expression to invert.
292 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The inverse of the matrix
293 *   expression.
294 */
295 inv(expr) {
296   this.checkLoaded();
297
298   var symbol = this._newSymbol();
299   symbol.setValue(this.eval(`
300     ${symbol.name} = ${expr}.inv()
301     ${symbol.name}
302   `));
303   return symbol;
304 }
305
306 /**
307 * Computes the determinant of a symbolic matrix expression.
308 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The matrix
309 *   expression whose determinant is to be computed.
310 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The determinant of the matrix
311 *   expression.
312 */
313 det(expr) {
314   this.checkLoaded();
315
316   var symbol = this._newSymbol();
317   symbol.setValue(this.eval(`
318     ${symbol.name} = ${expr}.det()
319     ${symbol.name}
320   `));
321   return symbol;
322 }
323
324 /**
325 * Differentiates a symbolic expression with respect to a variable.
326 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The expression to
327 *   differentiate.
328 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} x - The variable with
329 *   respect to which differentiation is performed.
330 * @param {number} [n=1] - The order of differentiation.
331 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The differentiated expression.
332 */
333 diff(expr, x, n = 1) {
334   this.checkLoaded();

```

```

334     expr = this.getSymbolName(expr);
335
336     var symbol = this._newSymbol();
337     symbol.setValue(this.eval(
338         '${symbol.name} = diff(${expr}, ${x}, ${n})
339         ${symbol.name}
340     ));
341     return symbol;
342 }
343
344 /**
345 * Integrates a symbolic expression with respect to a variable.
346 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The expression to
347 * integrate.
348 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} x - The variable with
349 * respect to which integration is performed.
350 * @param {Array<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string>|undefined} lims -
351 * The limits of integration as [lower, upper]. If undefined, indefinite
352 * integration is performed.
353 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The integrated expression.
354 */
355 intg(expr, x, lims) {
356     this.checkLoaded();
357
358     expr = this.getSymbolName(expr);
359     x = this.getSymbolName(x);
360
361     var symbol = this._newSymbol();
362     if(lims) {
363         lims[0] = this.getSymbolName(lims[0]);
364         lims[1] = this.getSymbolName(lims[1]);
365         symbol.setValue(this.eval(
366             '${symbol.name} = integrate(${expr}, (${x}, ${lims[0]}, ${lims[1]}))
367             ${symbol.name}
368         ));
369     } else {
370         symbol.setValue(this.eval(
371             '${symbol.name} = integrate(${expr}, ${x})
372             ${symbol.name}
373         ));
374     }
375     return symbol;
376 }
377
378 /**
379 * Substitutes a value into a symbolic expression.
380 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The expression in
381 * which to substitute.
382 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} x - The variable to
383 * substitute.
384 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number|Array} val - The
385 * value or array of values to substitute.
386 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The expression after
387 * substitution.
388 */

```

```

381   subs(expr, x, val) {
382     this.checkLoaded();
383
384     expr = this.getSymbolName(expr);
385     x = this.getSymbolName(x);
386
387     var symbol = this._newSymbol();
388     if(Array.isArray(val)) {
389       val = JSON.stringify(val);
390       symbol.setValue(this.eval(`
391         ${symbol.name} = ${expr}.subs(${x}, ${val}).evalf() for val in ${val}
392         ${symbol.name}
393       `));
394     } else {
395       val = this.getSymbolName(val);
396       if(isNumber(val)) {
397         symbol.setValue(this.eval(`
398           ${symbol.name} = ${expr}.subs(${x}, ${val}).evalf()
399             ${symbol.name}
400           `));
401       } else {
402         symbol.setValue(this.eval(`
403           ${symbol.name} = ${expr}.subs(${x}, ${val})
404             ${symbol.name}
405           `));
406       }
407     }
408     return symbol;
409   }
410
411 /**
412 * Simplifies a symbolic expression.
413 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The expression to
414 *   simplify.
415 * @returns {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} The simplified expression.
416 */
417 simplify(expr) {
418   this.checkLoaded();
419
420   expr = this.getSymbolName(expr);
421
422   var symbol = this._newSymbol();
423   symbol.setValue(this.eval(`
424     ${symbol.name} = simplify(${expr})
425     ${symbol.name}
426   `));
427   return symbol;
428 }
429 /**
430 * Displays the LaTeX representation of a symbolic expression.
431 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string} expr - The expression to
432 *   display in LaTeX format.
433 */
434 showLatex(expr) {

```

```

434     expr = this.getSymbolName(expr);
435     this.jsl.env.dispLatex(this.eval(`\${{expr}}`));
436 }
437 /**
438 * Displays the LaTeX string of a symbolic expression.
439 * @param {PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL} expr - The expression to
440   convert to a LaTeX string.
441 */
442 dispLatex(expr) {
443     expr = this.getSymbolName(expr);
444     this.jsl.env.disp(this.eval(`\${{expr}}`));
445 }
446 /**
447 * Clears all symbolic variables and resets the symbolic math environment.
448 */
449 clear() {
450     if(this.loaded) {
451         this.pyodide.runPython(`globals().clear()`);
452         this.pyodide.runPython(`
453             from sympy import *
454             import numpy as np
455         `);
456         this._var_counter = 0;
457         this._symbols = [];
458     }
459 }
460 }
461 }
462
463 exports.PRDC_JSLAB_SYMBOLIC_MATH = PRDC_JSLAB_SYMBOLIC_MATH;
464
465 /**
466 * Class for JSLAB symbolic math symbol.
467 */
468 class PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL {
469
470 /**
471 * Constructs a symbolic math symbol with a name and initial value.
472 * @constructor
473 * @param {string} name - The name of the symbolic variable.
474 * @param {*} value - The initial value of the symbolic variable.
475 */
476 constructor(name, value) {
477     this.name = name;
478     this.value = value;
479 }
480
481 /**
482 * Sets the value of the symbolic variable.
483 * @param {*} value - The new value to assign to the symbolic variable.
484 */
485 setValue(value) {
486     this.value = value;
487     this.parsed_value = parseString(value.toString());

```

```
488     }
489
490     /**
491      * Returns a string representation of the vector.
492      * @returns {string} The string representation in the format 'Vector(x:, y:, z:)'.
493      */
494     toString() {
495       return `Symbolic(${JSON.stringify(this.toJSON(), null, 2)})`;
496     }
497
498     /**
499      * Converts the symbolic value to a numeric value.
500      * @returns {*} The numeric representation of the symbolic value.
501      */
502     toNumeric() {
503       return this.parsed_value;
504     }
505
506     /**
507      * Converts the symbolic value to a JSON string.
508      * @returns {string} The JSON string representation of the symbolic value.
509      */
510     toJSON() {
511       if(typeof this.parsed_value === undefined) {
512         return this.name;
513       }
514       return this.parsed_value;
515     }
516
517     /**
518      * Converts the object to a safe JSON representation.
519      * @returns {Object} The safe JSON representation of the object.
520      */
521     toSafeJSON() {
522       return this.toJSON();
523     }
524
525     /**
526      * Converts the object to a pretty string representation.
527      * @returns {string} The pretty string representation of the object.
528      */
529     toPrettyString() {
530       return this.toString();
531     }
532   }
533
534   /**
535    * Parses a string representation of Python objects into JavaScript objects.
536    * Specifically handles conversion of SymPy Matrix objects to nested arrays.
537    * @param {string} str - The string to parse.
538    * @returns {Array<Array<*>>|Array<Array<Array<*>>>} The parsed JavaScript
539    * representation of the input string.
540    */
541   function parseString(str) {
```

```

541 // Remove any leading or trailing square brackets if the input is a list
542 const trimmed_str = str.trim().replace(/^\[\|\]$/g, '');
543
544 // Find all occurrences of "Matrix([ [... ]])" using regex
545 const matrix_regex = /Matrix\(\[\|\[.*?\]\|\]\)/g;
546 const matrices = [];
547 let match;
548
549 // Iterate over all matches of the regex
550 while((match = matrix_regex.exec(trimmed_str)) !== null) {
551   // Each match's first capture group contains the matrix content
552   const matrix_content = match[1];
553
554   // Split rows by detecting "],[ " pattern
555   const rows = matrix_content.split(/\],\s*/);
556
557   // For each row, split by commas while ignoring any spaces
558   const parsed_rows = rows.map(row => row.split(',')).map(entry => {
559     const trimmed_entry = entry.trim();
560     // Convert numeric strings to numbers
561     return isNaN(trimmed_entry) ? trimmed_entry : Number(trimmed_entry);
562   });
563
564   // Push the parsed rows (matrix) into the matrices array
565   matrices.push(parsed_rows);
566 }
567
568 // If only one Matrix(...) was found, return the array directly instead of
569 // wrapping in another array
570 return matrices.length === 1 ? matrices[0] : matrices;
}

```

Listing 127 - sym-math.js

```

1 /**
2  * @file JSLAB library system submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6 */
7
8 /**
9  * Class for JSLAB system submodule.
10 */
11 class PRDC_JSLAB_LIB_SYSTEM {
12
13 /**
14  * Initializes a new instance of the system submodule.
15  * @param {Object} js1 Reference to the main JSLAB object.
16  */
17 constructor(js1) {
18   var obj = this;
19   this.js1 = js1;
20 }
21
22 /**

```

```
23 * Executes a system command and returns the output .
24 * @param {...*} args Command arguments .
25 * @returns {string|boolean} The output of the command as a string , or false
26 * if an error occurred .
27 */
28 system (... args) {
29     try {
30         return this . jsl . env . execSync (... args) . toString () ;
31     } catch (err) {
32         this . jsl . env . error ('@system: '+err);
33         return false ;
34     }
35 }
36 /**
37 * Executes a system command .
38 * @param {...*} args Command arguments .
39 */
40 exec (... args) {
41     try {
42         return this . jsl . env . exec (... args) ;
43     } catch (err) {
44         this . jsl . env . error ('@exec: '+err);
45         return false ;
46     }
47 }
48 /**
49 * Executes a system command .
50 * @param {...*} args Command arguments .
51 */
52 spawn (... args) {
53     try {
54         return this . jsl . env . spawn (... args) ;
55     } catch (err) {
56         this . jsl . env . error ('@spawn: '+err);
57         return false ;
58     }
59 }
60 }
61 /**
62 * Retrieves a list of all running tasks on the system .
63 * @returns {Array} An array containing the tasklist output or [ false , [] ]
64 * if an error occurred .
65 */
66 getTaskList () {
67     try {
68         return this . jsl . env . execSync ('tasklist') . toString () ;
69     } catch (err) {
70         this . jsl . env . error ('@getTaskList: '+err);
71         return false ;
72     }
73 }
74 /**
75 */
```

```

76  * Checks if a specific program is running and retrieves its process IDs.
77  * @param {string} program_name - The name of the program to check.
78  * @returns {Array} An array where the first element is a boolean indicating
79  *                 if the program is running,
80  *                 and the second element is an array of process IDs if the
81  *                 program is running.
82  */
83  isProgramRunning(program_name) {
84    try {
85      // Execute the tasklist command and get the output
86      const output = this.jsl.env.execSync('tasklist').toString();
87
88      // Convert the output to lowercase for case-insensitive comparison
89      const output_lower = output.toLowerCase();
90      const program_nameLower = program_name.toLowerCase();
91
92      // Split the output into lines and filter the ones containing the
93      // program name
94      const lines = output_lower.split('\n');
95      const matching_lines = lines.filter(function(line) {
96        return line.includes(program_nameLower);
97      });
98
99      // Extract PIDs from the matching lines
100     const pids = matching_lines.map(line => {
101       // Split the line by spaces and filter out empty elements
102       const columns = line.trim().split(/\s+/);
103       return parseInt(columns[1], 10); // PID is the second column
104     }).filter(function(pid) {
105       return !isNaN(pid);
106     });
107
108     const is_running = pids.length > 0;
109     return [is_running, pids];
110   } catch(err) {
111     this.jsl.env.error('@isProgramRunning: '+err);
112     return [false, []];
113   }
114
115 /**
116  * Attempts to kill a process by its process ID.
117  * @param {number} pid - The process ID of the process to kill.
118  * @returns {string|boolean} The output of the kill command as a string, or
119  *                         false if an error occurred.
120  */
121 killProcess(pid) {
122   try {
123     // Execute the taskkill command and get the output
124     return this.jsl.env.execSync('taskkill /pid '+pid+' /T /F').toString();
125   } catch(err) {
126     this.jsl.env.error('@killProcess: '+err);
127     return false;
128   }
129 }
```

```

127  }
128
129 exports.PRDC_JSLAB_LIB_SYSTEM = PRDC_JSLAB_LIB_SYSTEM;

```

Listing 128 - system.js

```

1  /**
2   * @file JSLAB library time submodule
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB time submodule.
10  */
11 class PRDC_JSLAB_LIB_TIME {
12
13 /**
14  * Initializes the time submodule, setting up properties for time tracking
15  * and exposing time measurement utilities.
16  * @param {Object} jsl Reference to the main JSLAB object.
17  */
18 constructor(jsl) {
19   var obj = this;
20   this.jsl = jsl;
21
22 /**
23  * Current timezone string.
24  * @type {string}
25  */
26 this.timezone = 'Europe/Belgrade';
27
28 /**
29  * Last tic timestamp.
30  * @type {number}
31  */
32 this.last_tic = 0;
33
34 /**
35  * Starts a timer for measuring elapsed time. To be used with 'toc' to
36  * measure time intervals.
37  * @name tic
38  * @kind function
39  * @memberof PRDC_JSLAB_LIB_TIME
40  */
41 Object.defineProperty(this.jsl.context, 'tic', {configurable: true, get:
42   this._tic});
43
44 /**
45  * Records the current wall-clock time to measure elapsed time.
46  * @returns {number} The current wall-clock time in milliseconds.
47  */
48 _tic() {
49   this.jsl.context.last_tic = performance.now() / 1000; // [s]

```

```
48     return this.jsl.context.last_tic;
49 }
50
51 /**
52 * Calculates the wall-clock time elapsed since a specified start time or
53 * the last call to 'tic()', in seconds.
54 * @param {number} [tic] - The start time in milliseconds from which to
55 * calculate elapsed time. If omitted, uses the last time recorded by 'tic()
56 * ()'.
57 * @returns {number} The elapsed time in seconds.
58 */
59 toc(tic) {
60     var dt = performance.now() / 1000;
61     if(arguments.length < 1) {
62         return dt - this.jsl.context.last_tic;
63     } else {
64         return dt - tic;
65     }
66
67 /**
68 * Calculates the wall-clock time elapsed since a specified start time or
69 * the last call to 'tic()', in seconds.
70 * @param {number} [tic] - The start time in milliseconds from which to
71 * calculate elapsed time. If omitted, uses the last time recorded by 'tic()
72 * ()'.
73 * @returns {number} The elapsed time in milliseconds.
74 */
75 tocms(tic) {
76     if(!tic) {
77         tic = global.last_tic;
78     }
79     return (performance.now() - tic * 1000);
80 }
81
82 /**
83 * Gets the current Unix timestamp adjusted for a specified timezone.
84 * @returns {number} The current Unix timestamp as an integer.
85 */
86 getTimestamp() {
87     return luxon.DateTime.now().setZone(this.timezone).toMillis();
88 }
89
90 /**
91 * Gets the current time as a string adjusted for a specified timezone.
92 * @returns {string} The current time in 'HH:mm:ss' format.
93 */
94 getTime() {
95     return luxon.DateTime.now().setZone(this.timezone).toFormat('HH:mm:ss');
96 }
97
98 /**
99 * Gets the current time with milliseconds as a string adjusted for a
100 * specified timezone.
101 * @returns {string} The current time in 'HH:mm:ss.SSS' format.
```

```

96  /*
97   * Gets the current full time as a string adjusted for a specified timezone.
98   * @returns {string} The current date and time in 'HH:mm:ss.SSS' format.
99   */
100  getFullTime() {
101    return luxon.DateTime.now().setZone(this.timezone).toFormat('HH:mm:ss.SSS');
102  }
103
104 /**
105  * Gets the current date as a string adjusted for a specified timezone.
106  * @returns {string} The current date in 'dd.MM.yyyy.' format.
107  */
108  getDate() {
109    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.');
110  }
111
112 /**
113  * Gets the current date and time as a string adjusted for a specified
114  * timezone.
115  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss'
116  * format.
117  */
118  getTime() {
119    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.
120      HH:mm:ss');
121  }
122
123 /**
124  * Gets the current date and time with milliseconds as a string adjusted for
125  * a specified timezone.
126  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss.SSS'
127  * format.
128  */
129  getDateTimeFull() {
130    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.
131      HH:mm:ss.SSS');
132  }
133
134 /**
135  * Gets the current date and time as a string suitable for filenames,
136  * adjusted for a specified timezone.
137  * @returns {string} The current date and time in 'ddMMyyyy_HHmmss' format
138  * for use in filenames.
139  */
140  getDateTimeStr() {
141    return luxon.DateTime.now().setZone(this.timezone).toFormat(
142      'ddMMyyyy_HHmmss');
143  }
144
145 /**
146  * Sets the timezone to be used for time calculations and formatting. This
147  * method allows the application to adjust displayed times according to a
148  * specific timezone.
149  * @param {String} tz The timezone identifier (e.g., "America/New_York", "Europe/Paris")
150  * to be set for all time-related operations.
151  */
152
153 /**
154  * Gets the current date and time as a string adjusted for a specified timezone.
155  * @returns {string} The current date and time in 'dd.MM.yyyy.' format.
156  */
157  getTime() {
158    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.');
159  }
160
161 /**
162  * Gets the current date and time as a string adjusted for a specified
163  * timezone.
164  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss'
165  * format.
166  */
167  getDateTimeFull() {
168    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.
169      HH:mm:ss');
170  }
171
172 /**
173  * Gets the current date and time with milliseconds as a string adjusted for
174  * a specified timezone.
175  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss.SSS'
176  * format.
177  */
178  getDateTimeStr() {
179    return luxon.DateTime.now().setZone(this.timezone).toFormat(
180      'ddMMyyyy_HHmmss');
181  }
182
183 /**
184  * Sets the timezone to be used for time calculations and formatting. This
185  * method allows the application to adjust displayed times according to a
186  * specific timezone.
187  * @param {String} tz The timezone identifier (e.g., "America/New_York", "Europe/Paris")
188  * to be set for all time-related operations.
189  */
190
191 /**
192  * Gets the current date and time as a string suitable for filenames,
193  * adjusted for a specified timezone.
194  * @returns {string} The current date and time in 'ddMMyyyy_HHmmss' format
195  * for use in filenames.
196  */
197  getTime() {
198    return luxon.DateTime.now().setZone(this.timezone).toFormat('ddMMyyyy_HHmmss');
199  }
200
201 /**
202  * Gets the current date and time as a string adjusted for a specified timezone.
203  * @returns {string} The current date and time in 'dd.MM.yyyy.' format.
204  */
205  getDate() {
206    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.');
207  }
208
209 /**
210  * Gets the current date and time as a string adjusted for a specified
211  * timezone.
212  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss'
213  * format.
214  */
215  getDateTimeFull() {
216    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.
217      HH:mm:ss');
218  }
219
220 /**
221  * Gets the current date and time with milliseconds as a string adjusted for
222  * a specified timezone.
223  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss.SSS'
224  * format.
225  */
226  getDateTimeStr() {
227    return luxon.DateTime.now().setZone(this.timezone).toFormat(
228      'ddMMyyyy_HHmmss');
229  }
230
231 /**
232  * Sets the timezone to be used for time calculations and formatting. This
233  * method allows the application to adjust displayed times according to a
234  * specific timezone.
235  * @param {String} tz The timezone identifier (e.g., "America/New_York", "Europe/Paris")
236  * to be set for all time-related operations.
237  */
238
239 /**
240  * Gets the current date and time as a string suitable for filenames,
241  * adjusted for a specified timezone.
242  * @returns {string} The current date and time in 'ddMMyyyy_HHmmss' format
243  * for use in filenames.
244  */
245  getTime() {
246    return luxon.DateTime.now().setZone(this.timezone).toFormat('ddMMyyyy_HHmmss');
247  }
248
249 /**
250  * Gets the current date and time as a string adjusted for a specified timezone.
251  * @returns {string} The current date and time in 'dd.MM.yyyy.' format.
252  */
253  getDate() {
254    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.');
255  }
256
257 /**
258  * Gets the current date and time as a string adjusted for a specified
259  * timezone.
260  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss'
261  * format.
262  */
263  getDateTimeFull() {
264    return luxon.DateTime.now().setZone(this.timezone).toFormat('dd.MM.yyyy.
265      HH:mm:ss');
266  }
267
268 /**
269  * Gets the current date and time with milliseconds as a string adjusted for
270  * a specified timezone.
271  * @returns {string} The current date and time in 'dd.MM.yyyy. HH:mm:ss.SSS'
272  * format.
273  */
274  getDateTimeStr() {
275    return luxon.DateTime.now().setZone(this.timezone).toFormat(
276      'ddMMyyyy_HHmmss');
277  }
278
279 /**
280  * Sets the timezone to be used for time calculations and formatting. This
281  * method allows the application to adjust displayed times according to a
282  * specific timezone.
283  * @param {String} tz The timezone identifier (e.g., "America/New_York", "Europe/Paris")
284  * to be set for all time-related operations.
285  */
286
287 /**
288  * Gets the current date and time as a string suitable for filenames,
289  * adjusted for a specified timezone.
290  * @returns {string} The current date and time in 'ddMMyyyy_HHmmss' format
291  * for use in filenames.
292  */
293  getTime() {
294    return luxon.DateTime.now().setZone(this.timezone).toFormat('ddMMyyyy_HHmmss');
295  }
296
297 /**
298  * Gets the current date and time as a string adjusted for a specified timezone.
299  * @returns {string} The current date and time in 'dd.MM.yyyy.' format.
300  */
301  getDate() {
301  }

```

```

137     setTimezone(tz) {
138         this.timezone = tz;
139     }
140 }
141
142 exports.PRDC_JSLAB_LIB_TIME = PRDC_JSLAB_LIB_TIME;

```

Listing 129 - time.js

```

1  /**
2  * @file JSLAB library vector math submodule
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 /**
9 * Class for JSLAB vector math submodule.
10 */
11 class PRDC_JSLAB_VECTOR_MATH {
12
13 /**
14 * Constructs vector math submodule object with access to JSLAB's vector
15 * functions.
16 * @constructor
17 * @param {Object} js1 - Reference to the main JSLAB object.
18 */
19 constructor(js1) {
20     this.js1 = js1;
21 }
22
23 /**
24 * Creates a new vector with specified x, y, z components.
25 * @param {number} x - The x-component of the vector.
26 * @param {number} y - The y-component of the vector.
27 * @param {number} z - The z-component of the vector.
28 * @returns {PRDC_JSLAB_VECTOR} A new vector instance.
29 */
30 new(x, y, z) {
31     return new PRDC_JSLAB_VECTOR(this.js1, x, y, z);
32 }
33
34 /**
35 * Creates a vector from polar coordinates.
36 * @param {number} length - The length of the vector.
37 * @param {number} radian - The angle in radians.
38 * @returns {PRDC_JSLAB_VECTOR} The resulting vector.
39 */
40 polar(length, radian) {
41     const x = length * Math.cos(radian);
42     const y = length * Math.sin(radian);
43     const z = 0;
44     return new PRDC_JSLAB_VECTOR(this.js1, x, y, z);
45 }
46 */

```

```

47  * Creates a vector from spherical coordinates.
48  * @param {number} length - The length (magnitude) of the vector.
49  * @param {number} azimuth - The azimuth angle in radians (angle from the X-
50   axis in the XY plane).
51  * @param {number} elevation - The elevation angle in radians (angle from
52   the XY plane towards Z).
53  * @returns {PRDC_JSLAB_VECTOR} The resulting vector.
54  */
55  spherical(length, azimuth, elevation) {
56    const x = length * Math.cos(elevation) * Math.cos(azimuth);
57    const y = length * Math.cos(elevation) * Math.sin(azimuth);
58    const z = length * Math.sin(elevation);
59    return new PRDC_JSLAB_VECTOR(this.jsl, x, y, z);
60  }
61
62
63 /**
64  * Class for JSLAB vector.
65  */
66 class PRDC_JSLAB_VECTOR {
67
68  #jsl;
69
70  /**
71   * Creates a new vector instance.
72   * @constructor
73   * @param {Object} jsl - Reference to the main JSLAB object.
74   * @param {number|Object} x - The x-component or an object with x, y, z
75   * properties.
76   * @param {number} [y=0] - The y-component of the vector.
77   * @param {number} [z=0] - The z-component of the vector.
78   */
79  constructor(jsl, x, y, z) {
80    this.#jsl = jsl;
81    this._set(x, y, z);
82  }
83
84  /**
85   * Sets the components of the vector.
86   * @param {number|Object} x - The x-component or an object with x, y, z
87   * properties.
88   * @param {number} [y=0] - The y-component of the vector.
89   * @param {number} [z=0] - The z-component of the vector.
90   * @returns {PRDC_JSLAB_VECTOR} The updated vector instance.
91   */
92  _set(x, y, z) {
93    if(Array.isArray(x)) {
94      z = x[2];
95      y = x[1];
96      x = x[0];
97    } else if(isObject(x)) {
98      z = x.z;
99      y = x.y;
100    }
101  }

```

```
98         x = x.x;
99     }
100
101    this.x = x || 0;
102    this.y = y || 0;
103    this.z = z || 0;
104}
105
106 /**
107 * Calculates the length (magnitude) of the vector.
108 * @returns {number} The length of the vector.
109 */
110 length() {
111     return Math.sqrt(this.x * this.x + this.y * this.y + this.z * this.z);
112 }
113
114 /**
115 * Calculates the length (magnitude) of the vector.
116 * @returns {number} The length of the vector.
117 */
118 norm() {
119     return this.length();
120 }
121
122 /**
123 * Adds two vectors and returns the result.
124 * @param {PRDC_JSLAB_VECTOR} v - The second vector.
125 * @returns {PRDC_JSLAB_VECTOR} The resulting vector after addition.
126 */
127 add(v) {
128     return this.#jsl.vec.new(this.x + v.x, this.y + v.y, this.z + v.z);
129 }
130
131 /**
132 * Adds two vectors and returns the result.
133 * @param {PRDC_JSLAB_VECTOR} v - The second vector.
134 * @returns {PRDC_JSLAB_VECTOR} The resulting vector after addition.
135 */
136 plus(v) {
137     return this.add(v);
138 }
139
140 /**
141 * Subtracts the second vector from the first and returns the result.
142 * @param {PRDC_JSLAB_VECTOR} v - The vector to subtract.
143 * @returns {PRDC_JSLAB_VECTOR} The resulting vector after subtraction.
144 */
145 subtract(v) {
146     return this.#jsl.vec.new(this.x - v.x, this.y - v.y, this.z - v.z);
147 }
148
149 /**
150 * Subtracts the second vector from the first and returns the result.
151 * @param {PRDC_JSLAB_VECTOR} v - The vector to subtract.
152 * @returns {PRDC_JSLAB_VECTOR} The resulting vector after subtraction.
```

```

153     */
154     minus(v) {
155         return this.subtract(v);
156     }
157
158     /**
159      * Scales a vector by the given factors.
160      * @param {number|Object} scale_x - The scale factor for the x-component or
161      * an object with x, y, z properties.
162      * @param {number} [scale_y] - The scale factor for the y-component.
163      * @param {number} [scale_z] - The scale factor for the z-component.
164      * @returns {PRDC_JSLAB_VECTOR} The scaled vector.
165      */
166     scale(scale_x, scale_y, scale_z) {
167         if(isObject(scale_x)) {
168             scale_x = scale_x.x;
169             scale_y = scale_x.y;
170             scale_z = scale_x.z;
171         } else if(!isNumber(scale_y)) {
172             scale_y = scale_x;
173             scale_z = scale_x;
174         }
175         return this.#jsl.vec.new(this.x * scale_x, this.y * scale_y, this.z *
176             scale_z);
177     }
178
179     /**
180      * Scales a vector by the given factor.
181      * @param {number} s - The scale factor.
182      * @returns {PRDC_JSLAB_VECTOR} The scaled vector.
183      */
184     multiply(s) {
185         return this.scale(s);
186     }
187
188     /**
189      * Scales a vector by dividing each element by given factor.
190      * @param {number} s - The scale factor.
191      * @returns {PRDC_JSLAB_VECTOR} The scaled vector.
192      */
193     divide(s) {
194         return this.scale(1 / s);
195     }
196
197     /**
198      * Checks if two vectors are equal.
199      * @param {PRDC_JSLAB_VECTOR} v - The second vector.
200      * @returns {boolean} True if vectors are equal, false otherwise.
201      */
202     equals(v) {
203         return this.x == v.x && this.y == v.y && this.z == v.z;
204     }
205
206     /**
207      * Calculates the angle of a vector.

```

```
206     * @param {PRDC_JSLAB_VECTOR} v - The vector.
207     * @returns {number} The angle in degrees.
208     */
209    angleTo(v) {
210      let cos_theta = this.dot(v) / (this.norm() * v.norm());
211      return Math.acos(cos_theta) * (180 / Math.PI);
212    }
213
214    /**
215     * Projects vector to given vector.
216     * @param {PRDC_JSLAB_VECTOR} v - The vector.
217     * @returns {PRDC_JSLAB_VECTOR} The projected vector.
218     */
219    projectTo(v) {
220      return v.multiply(this.dot(v) / Math.pow(v.norm(), 2));
221    }
222
223    /**
224     * Calculates the angles of a vector.
225     * @param {PRDC_JSLAB_VECTOR} v - The vector.
226     * @returns {Array} The angles azimuth and elevation in degrees.
227     */
228    angles() {
229      const length_xy = Math.sqrt(this.x * this.x + this.y * this.y);
230      const azimuth = Math.atan2(this.y, this.x) * (180 / Math.PI);
231      const elevation = Math.atan2(this.z, length_xy) * (180 / Math.PI);
232      return [azimuth, elevation];
233    }
234
235    /**
236     * Calculates the distance between two vectors.
237     * @param {PRDC_JSLAB_VECTOR} v - The second vector.
238     * @returns {number} The distance between the two vectors.
239     */
240    distance(v) {
241      var a = this.x - v.x;
242      var b = this.y - v.y;
243      var c = this.z - v.z;
244      return Math.sqrt(a * a + b * b + c * c);
245    }
246
247    /**
248     * Calculates the dot product of two vectors.
249     * @param {PRDC_JSLAB_VECTOR} v - The second vector.
250     * @returns {number} The dot product.
251     */
252    dot(v) {
253      return this.x * v.x + this.y * v.y + this.z * v.z;
254    }
255
256    /**
257     * Calculates the cross product of two vectors.
258     * @param {PRDC_JSLAB_VECTOR} v - The second vector.
259     * @returns {PRDC_JSLAB_VECTOR} The cross product vector.
260     */
```

```

261  cross(v) {
262      var cross_x = this.y * v.z - this.z * v.y;
263      var cross_y = this.z * v.x - this.x * v.z;
264      var cross_z = this.x * v.y - this.y * v.x;
265      return this.#jsl.vec.new(cross_x, cross_y, cross_z);
266  }
267
268  /**
269   * Interpolates between two vectors by a factor.
270   * @param {PRDC_JSLAB_VECTOR} v - The ending vector.
271   * @param {number} f - The interpolation factor.
272   * @returns {PRDC_JSLAB_VECTOR} The interpolated vector.
273   */
274  interpolate(v, f) {
275      var dx = v.x - this.x;
276      var dy = v.y - this.y;
277      var dz = v.z - this.z;
278      return this.#jsl.vec.new(this.x + dx * f, this.y + dy * f, this.z + dz * f
279          );
280  }
281  /**
282   * Offsets the vector by the given amounts.
283   * @param {number|Object} x - The amount to offset in the x-direction or an
284   * object with x, y, z properties.
285   * @param {number} [y=0] - The amount to offset in the y-direction.
286   * @param {number} [z=0] - The amount to offset in the z-direction.
287   * @returns {PRDC_JSLAB_VECTOR} The updated vector instance.
288   */
289  offset(x, y, z) {
290      if(isObject(x)) {
291          x = x.x;
292          z = x.z;
293          y = x.y;
294      }
295      x = this.x + x || 0;
296      y = this.y + y || 0;
297      z = this.z + z || 0;
298
299      return this.#jsl.vec.new(x, y, z);
300  }
301
302  /**
303   * Normalizes the vector to have a length of 1.
304   * @returns {PRDC_JSLAB_VECTOR} The normalized vector.
305   */
306  normalize() {
307      var length = this.length();
308
309      if(length > 0) {
310          var x = this.x / length;
311          var y = this.y / length;
312          var z = this.z / length;
313          return this.#jsl.vec.new(x, y, z);

```

```

314     }
315     return this.clone();
316 }
317
318 /**
319 * Negates the vector components.
320 * @returns {PRDC_JSLAB_VECTOR} The negated vector.
321 */
322 negate() {
323     var x = this.x * -1;
324     var y = this.y * -1;
325     var z = this.z * -1;
326
327     return this.#jsl.vec.new(x, y, z);
328 }
329
330 /**
331 * Creates a clone of this vector.
332 * @returns {PRDC_JSLAB_VECTOR} A new vector instance with the same
333 * components.
334 */
335 clone() {
336     return this.#jsl.vec.new(this.x, this.y, this.z);
337 }
338
339 /**
340 * Converts the vector to an array.
341 * @returns {number[]} An array containing the x, y, z components of the
342 * vector.
343 */
344 toArray() {
345     return [this.x, this.y, this.z];
346 }
347
348 /**
349 * Converts the vector to a matrix.
350 * @returns {Object} A matrix representation of the vector.
351 */
352 toMatrix() {
353     return this.#jsl.mat.new([this.x, this.y, this.z], 3, 1);
354 }
355
356 /**
357 * Converts the vector to a column matrix.
358 * @returns {Object} A column matrix representation of the vector.
359 */
360 toColMatrix() {
361     return this.#jsl.mat.new([this.x, this.y, this.z], 1, 3);
362 }
363
364 /**
365 * Converts the vector to a row matrix.
366 * @returns {Object} A row matrix representation of the vector.
367 */
368 toRowMatrix() {

```

```

367     return this.toMatrix();
368 }
369
370 /**
371 * Returns a string representation of the vector.
372 * @returns {string} The string representation in the format 'Vector(x:, y:, z:)'.
373 */
374 toString() {
375     return 'Vector([' + this.x + ', ' + this.y + ', ' + this.z + '])';
376 }
377
378 /**
379 * Returns a string representation of the vector.
380 * @returns {string} The string representation in the format 'Vector(x:, y:, z:)'.
381 */
382 toJSON() {
383     return { x: this.x, y: this.y, z: this.z };
384 }
385
386 /**
387 * Returns a string representation of the vector.
388 * @returns {string} The string representation in the format 'Vector(x:, y:, z:)'.
389 */
390 toSafeJSON() {
391     return this.toJSON();
392 }
393
394 /**
395 * Converts the object to a pretty string representation.
396 * @returns {string} The pretty string representation of the object.
397 */
398 toPrettyString() {
399     return this.toString();
400 }
401 }
```

Listing 130 - vector-math.js

```

1 /**
2 * @file JSLAB library windows submodule
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB windows submodule.
10 */
11 class PRDC_JSLAB_LIB_WINDOWS {
12
13 /**
14 * Initializes a new instance of the windows submodule.
15 * @param {Object} js1 Reference to the main JSLAB object.

```

```
16 */  
17 constructor(jsl) {  
18     var obj = this;  
19     this.jsl = jsl;  
20     this._wid = 0;  
21  
22     /**  
23      * Current active window ID.  
24      * @type {Number}  
25      */  
26     this.active_window;  
27  
28     /**  
29      * Array of open windows.  
30      * @type {Array}  
31      */  
32     this.open_windows = {};  
33 }  
34  
35 /**  
36  * Opens a new window with the specified file.  
37  * @param {string} file - The path to the HTML file to open in the new  
38  * window.  
39  * @returns {number} The identifier (wid) of the newly opened window.  
40  */  
41 openWindow(file) {  
42     if(!this.jsl.env.pathIsAbsolute(file)) {  
43         file = app_path + '/html/' + file;  
44     }  
45  
46     if(!this.jsl.env.checkFile(file)) {  
47         this.jsl.env.error('@openWindow: '+language.string(199));  
48     }  
49  
50     this._wid += 1;  
51     var wid = this._wid;  
52     this.open_windows[wid] = new PRDC_JSLAB_WINDOW(this.jsl, wid);  
53     this.open_windows[wid].open(file);  
54     this._setActiveWindow(wid);  
55     this.jsl.no_ans = true;  
56     this.jsl.ignore_output = true;  
57     return wid;  
58 }  
59  
60 /**  
61  * Opens the developer tools for a specified window by ID if it exists.  
62  * @param {string} wid - The window ID.  
63  * @returns {boolean} True if the developer tools were opened; otherwise,  
64  * false.  
65  */  
66 openWindowDevTools(wid) {  
67     if(this.open_windows.hasOwnProperty(wid)) {  
68         return this.open_windows[wid].openDevTools();  
69     } else {  
70         return false;  
71     }  
72 }
```

```
69      }
70  }
71
72 /**
73 * Returns media source id for a specified window in the renderer process .
74 * @param {string} wid - The window ID .
75 * @returns {String|boolean} Media source id if there is window; otherwise ,
76     false .
77 */
78 getWindowMediaSourceId(wid) {
79     if(this.open_windows.hasOwnProperty(wid)) {
80         return this.open_windows[wid].getMediaSourceId();
81     } else {
82         return false;
83     }
84
85 /**
86 * Starts video recording of the window .
87 * @param {string} wid - The window ID .
88 * @param {Object} - Optional settings .
89 * @returns {Object|boolean} - Returns recorder object if there is window ;
90     otherwise , false .
91 */
92 startWindowVideoRecording(wid, opts) {
93     if(this.open_windows.hasOwnProperty(wid)) {
94         return this.open_windows[wid].startVideoRecording(opts);
95     } else {
96         return false;
97     }
98
99 /**
100 * Closes the specified window .
101 * @param {number} wid - Identifier for the window to close .
102 */
103 closeWindows(wid) {
104     this.jsl.env.closeWindow(wid);
105     this.jsl.no_ans = true;
106     this.jsl.ignore_output = true;
107 }
108
109 /**
110 * Closes the specified window .
111 * @param {number} wid - Identifier for the window to close .
112 */
113 closeWindow(wid) {
114     if(this.open_windows.hasOwnProperty(wid)) {
115         return this.open_windows[wid].close();
116     } else {
117         return false;
118     }
119 }
120
121 /**
```

```
122 * Retrieves the window object with the specified ID.  
123 * @param {number} wid - The ID of the window.  
124 * @returns {Object|boolean} - The window object if found, otherwise false.  
125 */  
126 getWindow(wid) {  
127     if(this.open_windows.hasOwnProperty(wid)) {  
128         return this.open_windows[wid];  
129     } else {  
130         return false;  
131     }  
132 }  
133  
134 /**
135 * Retrieves current active window object.  
136 * @returns {Object|boolean} - The window object if found, otherwise false.  
137 */  
138 getCurrentWindow() {  
139     if(this.open_windows.hasOwnProperty(this.active_window)) {  
140         return this.open_windows[this.active_window];  
141     } else {  
142         return false;  
143     }  
144 }  
145  
146 /**
147 * Retrieves current active window object.  
148 * @returns {Object|boolean} - The window object if found, otherwise false.  
149 */  
150 gcw() {  
151     return this.getCurrentWindow();  
152 }  
153  
154 /**
155 * Shows the specified window.  
156 * @param {number} wid - The ID of the window to show.  
157 * @returns {boolean|undefined} - Returns false if the window ID is invalid,  
158 * otherwise the result of the show() method.  
159 */  
160 showWindow(wid) {  
161     if(this.open_windows.hasOwnProperty(wid)) {  
162         return this.open_windows[wid].show();  
163     } else {  
164         return false;  
165     }  
166 }  
167 /**
168 * Hides the specified window.  
169 * @param {number} wid - The ID of the window to hide.  
170 * @returns {boolean|undefined} - Returns false if the window ID is invalid,  
171 * otherwise the result of the hide() method.  
172 */  
173 hideWindow(wid) {  
174     if(this.open_windows.hasOwnProperty(wid)) {  
175         return this.open_windows[wid].hide();  
176     }  
177 }
```

```
175     } else {
176         return false;
177     }
178 }
179
180 /**
181 * Brings the specified window to the foreground.
182 * @param {number} wid - The ID of the window to focus.
183 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
184 */
185 focusWindow(wid) {
186     if(this.open_windows.hasOwnProperty(wid)) {
187         return this.open_windows[wid].focus();
188     } else {
189         return false;
190     }
191 }
192
193 /**
194 * Minimizes the specified window.
195 * @param {number} wid - The ID of the window to minimize.
196 * @returns {boolean|undefined} - Returns false if the window ID is invalid,
197 * otherwise the result of the minimize() method.
198 */
199 minimizeWindow(wid) {
200     if(this.open_windows.hasOwnProperty(wid)) {
201         return this.open_windows[wid].minimize();
202     } else {
203         return false;
204     }
205
206 /**
207 * Centers the specified window on the screen.
208 * @param {number} wid - The ID of the window to center.
209 * @returns {boolean|undefined} - Returns false if the window ID is invalid,
210 * otherwise the result of the center() method.
211 */
212 centerWindow(wid) {
213     if(this.open_windows.hasOwnProperty(wid)) {
214         return this.open_windows[wid].center();
215     } else {
216         return false;
217     }
218
219 /**
220 * Moves the specified window to the top of the window stack.
221 * @param {number} wid - The ID of the window to move to the top.
222 * @returns {boolean|undefined} - Returns false if the window ID is invalid,
223 * otherwise the result of the moveTop() method.
224 */
225 moveTopWindow(wid) {
226     if(this.open_windows.hasOwnProperty(wid)) {
227         return this.open_windows[wid].moveTop();
```

```
227     } else {
228         return false;
229     }
230 }
231
232 /**
233 * Sets the size of the specified window.
234 * @param {number} wid - The ID of the window.
235 * @param {number} width - The new width of the window.
236 * @param {number} height - The new height of the window.
237 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
238 */
239 setWindowSize(wid, width, height) {
240     if(this.open_windows.hasOwnProperty(wid)) {
241         return this.open_windows[wid].setSize(width, height);
242     } else {
243         return false;
244     }
245 }
246
247 /**
248 * Sets the position of the specified window.
249 * @param {number} wid - The ID of the window.
250 * @param {number} left - The new left position of the window.
251 * @param {number} top - The new top position of the window.
252 * @returns {boolean|undefined} - Returns false if the window ID is invalid.
253 */
254 setWindowPos(wid, left, top) {
255     if(this.open_windows.hasOwnProperty(wid)) {
256         return this.open_windows[wid].setPos(left, top);
257     } else {
258         return false;
259     }
260 }
261
262 /**
263 * Sets the resizable state of the specified window.
264 * @param {number} wid - The ID of the window.
265 * @param {boolean} state - Whether the window should be resizable.
266 * @returns {boolean|undefined} - Returns false if the window ID is invalid,
267 *          otherwise the result of the setResizable() method.
268 */
269 setWindowResizable(wid, state) {
270     if(this.open_windows.hasOwnProperty(wid)) {
271         return this.open_windows[wid].setResizable(state);
272     } else {
273         return false;
274     }
275 }
276 /**
277 * Sets the movable state of the specified window.
278 * @param {number} wid - The ID of the window.
279 * @param {boolean} state - Whether the window should be movable.
280 * @returns {boolean|undefined} - Returns false if the window ID is invalid,
```

```
otherwise the result of the setMovable() method.  
281 */  
282 setWindowMovable(wid, state) {  
283     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
284         return this.open_windows[wid].setMovable(state);  
285     } else {  
286         return false;  
287     }  
288 }  
289  
290 /**
291 * Sets the aspect ratio of the specified window.  
292 * @param {number} wid - The ID of the window.  
293 * @param {number} aspect_ratio - The desired aspect ratio of the window.  
294 * @returns {boolean|undefined} - Returns false if the window ID is invalid,  
295         otherwise the result of the setAspectRatio() method.  
296 */  
297 setWindowAspectRatio(wid, aspect_ratio) {  
298     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
299         return this.open_windows[wid].setAspectRatio(aspect_ratio);  
300     } else {  
301         return false;  
302     }  
303 }  
304 /**
305 * Sets the opacity of the specified window.  
306 * @param {number} wid - The ID of the window.  
307 * @param {number} opacity - The desired opacity level (0 to 1).  
308 * @returns {boolean|undefined} - Returns false if the window ID is invalid,  
309         otherwise the result of the setOpacity() method.  
310 */  
311 setWindowOpacity(wid, opacity) {  
312     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
313         return this.open_windows[wid].setOpacity(opacity);  
314     } else {  
315         return false;  
316     }  
317 }  
318 /**
319 * Sets the fullscreen state of the specified window.  
320 * @param {number} wid - The ID of the window.  
321 * @param {number} state - The fullscreen state  
322 * @returns {boolean|undefined} - Returns false if the window ID is invalid,  
323         otherwise the result of the setFullscreen() method.  
324 */  
325 setWindowFullscreen(wid, state) {  
326     if(this.jsl.windows.open_windows.hasOwnProperty(wid)) {  
327         return this.open_windows[wid].setFullscreen(state);  
328     } else {  
329         return false;  
330     }  
331 }
```

```
332  /**
333   * Sets the position of the specified window.
334   * @param {number} wid - The ID of the window.
335   * @param {number} left - The new left position of the window.
336   * @param {number} top - The new top position of the window.
337   * @returns {boolean|undefined} - Returns false if the window ID is invalid.
338   */
339 setWindowTitle(wid, title) {
340     if(this.open_windows.hasOwnProperty(wid)) {
341       return this.open_windows[wid].setTitle(title);
342     } else {
343       return false;
344     }
345   }
346
347 /**
348   * Retrieves the size of the specified window.
349   * @param {number} wid - The ID of the window.
350   * @returns {Array|boolean} - An array [width, height] if the window exists,
351   * otherwise false.
352   */
353 getWindowSize(wid) {
354   if(this.open_windows.hasOwnProperty(wid)) {
355     return this.open_windows[wid].getSize();
356   } else {
357     return false;
358   }
359
360 /**
361   * Retrieves the position of the specified window.
362   * @param {number} wid - The ID of the window.
363   * @returns {Array|boolean} - An array [left, top] if the window exists,
364   * otherwise false.
365   */
366 getWindowPos(wid) {
367   if(this.open_windows.hasOwnProperty(wid)) {
368     return this.open_windows[wid].getPos();
369   } else {
370     return false;
371   }
372
373 /**
374   * Prints window to PDF format.
375   * @param {string} wid - The window ID.
376   * @param {Object} options - Options for printing
377   * @returns {Buffer} Generated PDF data.
378   */
379 printWindowToPdf(wid, options) {
380   if(this.open_windows.hasOwnProperty(wid)) {
381     return this.open_windows[wid].printToPdf(options);
382   } else {
383     return false;
384   }
```

```

385   }
386
387  /**
388   * Opens window with documentation
389   */
390  async openDocumentation(query) {
391   var wid = this.openWindow('../docs/documentation.html');
392   var win = this.open_windows[wid];
393   await win.ready;
394   if(query) {
395     win.context.location.href = '../docs/documentation.html#' + encodeURI(
396       query);
397     win.context.location.reload(true);
398   }
399   win.setTitle('JSLAB / DOCUMENTATION');
400 }
401 /**
402  * Opens window with documentation
403 */
404 async openDoc(query) {
405   await this.openDocumentation(query);
406 }
407 /**
408  * Opens a new 3D window and imports specified modules.
409  * @param {Array<Object>} [imports=[]] - An array of import objects
410  * specifying modules to import.
411  * @returns {Promise<Object>} A promise that resolves to the window object
412  * once imports are ready.
413 */
414 async openWindow3D(imports = []) {
415   var obj = this;
416   var wid = this.openWindow('three.html');
417   await this.open_windows[wid].ready;
418   var context = this.open_windows[wid].context;
419   var script = context.document.createElement('script');
420   script.type = 'module';
421
422   const import_statements = [];
423   const window_assignments = [];
424   var imports_array = [{import: '*', as: 'THREE', from: 'three'}];
425   imports_array.push(...imports);
426
427   imports_array.forEach((item) => {
428     const { import: imported, as, from } = item;
429
430     if(imported === '*') {
431       if(!as) {
432         this.jsl.env.error('@openWindow3D: '+language.string(200)+from+'.');
433       }
434       // Namespace import
435       import_statements.push(`import * as ${as} from '${from}';`);
436       window_assignments.push(`window.${as} = ${as};`);
437     } else if(Array.isArray(imported)) {

```

```

437   // Named imports with multiple specifiers
438   const specifiers = imported.join(', ');
439   import_statements.push(`import { ${specifiers} } from '${from}';`);
440   imported.forEach((spec) => {
441     window_assignments.push(`window.${spec} = ${spec};`);
442   });
443 } else {
444   // Single named import
445   import_statements.push(`import { ${imported} } from '${from}';`);
446   window_assignments.push(`window.${imported} = ${imported};`);
447 }
448 );
449
450 script.textContent = `${import_statements.join('\n')}\n\n${window_assignments.join('\n')}`.replace(/\n/g, '\n\n');
451
452 context.imports_ready = false;
453 context.document.body.appendChild(script);
454 while (!context.imports_ready) {
455   await this.jsl.non_blocking.waitMSeconds(1);
456 }
457 context.sceneToJSON = function(file_path, scene = context.scene) {
458   if (!scene) return;
459   if (!file_path) {
460     let options = {
461       title: language.currentString(144),
462       defaultPath: 'window-3d.json',
463       buttonLabel: language.currentString(144),
464       filters : [
465         {name: 'json', extensions: ['json']},
466       ]
467     };
468     file_path = obj.jsl.env.showSaveDialogSync(options);
469     if (!file_path) return;
470   }
471   var scene_json = scene.toJSON();
472   obj.jsl.env.writeFileSync(file_path, JSON.stringify(scene_json));
473 }
474
475 context.sceneFromJSON = function(file_path) {
476   if (!file_path) {
477     var options = {
478       title: language.currentString(247),
479       buttonLabel: language.currentString(231)
480     };
481     file_path = obj.jsl.env.showOpenDialogSync(options);
482     if (file_path === undefined) {
483       obj.jsl.env.error(`@openWindow3D.fromJSON: ${language.string(132)}`);
484       return false;
485     } else {
486       file_path = file_path[0];
487     }
488   }
489   if (!obj.jsl.file_system.existFile(pwd + file_path)) {

```

```

490     obj.jsl.env.error('@openWindow3D.fromJSON: '+language.string(248));
491     return false;
492   }
493   var data = JSON.parse(obj.jsl.env.readFileSync(pwd + file_path).toString());
494
495   const loader = new context.THREE.ObjectLoader();
496   return loader.parse(data)
497 }
498 return context;
499 }
500 /**
501 * Opens a Plotly.js window and initializes the plot container.
502 * @returns {Promise<Window>} The window object where Plotly.js is loaded
503 * and the plot container is available.
504 */
505 async openPlotlyjs() {
506   var wid = this.openWindow('plotlyjs.html');
507   await this.open_windows[wid].ready;
508   var context = this.open_windows[wid].context;
509   context.imports_ready = false;
510   while(!context.imports_ready) {
511     if(typeof context.Plotly != 'undefined') {
512       context.imports_ready = true;
513     }
514     await this.jsl.non_blocking.waitMSeconds(1);
515   }
516   context.plot_cont = context.document.getElementById('plot-cont');
517   context.plot_cont.style = 'position: absolute; top:0; left:0; right:0; bottom
518   :0;';
519   return context;
520 }
521 /**
522 * Opens a window with canvas and D3 and initializes the canvas element.
523 * @returns {Promise<Window>} The window object where D3 is loaded and the
524 * canvas element is available.
525 */
526 async openCanvas() {
527   var wid = this.openWindow('d3.html');
528   await this.open_windows[wid].ready;
529   var context = this.open_windows[wid].context;
530   context.imports_ready = false;
531   while(!context.imports_ready) {
532     if(typeof context.d3 != 'undefined') {
533       context.imports_ready = true;
534     }
535     await this.jsl.non_blocking.waitMSeconds(1);
536   }
537   context.svg = context.document.getElementById('d3-svg');
538   context.canvas = context.document.getElementById('d3-canvas');
539   context.svg.style = 'position: absolute; top:0; left:0; right:0; bottom:0;';
540   context.canvas.style = 'position: absolute; top:0; left:0; right:0; bottom:0;';
541 }

```

```

540     return context;
541   }
542
543   /**
544    * Opens a new blank window.
545    * @returns {Promise<Object>} A promise that resolves to the window object
546    * once it is ready.
547    */
548   async openWindowBlank() {
549     var wid = this.openWindow('blank.html');
550     await this.open_windows[wid].ready;
551     var context = this.open_windows[wid].context;
552     context.document.custom_style = context.document.getElementById('custom-
553       style');
554     return context;
555   }
556
557   /**
558    * Renders a Mermaid diagram in a new window.
559    * @param {string} graph_definition - The Mermaid graph definition.
560    * @returns {Promise<Object>} A promise that resolves to the window context
561    * once the graph is rendered.
562    */
563   async showMermaidGraph(graph_definition) {
564     var wid = this.openWindow('mermaid_graph.html');
565     await this.open_windows[wid].ready;
566     var context = this.open_windows[wid].context;
567     context.document.custom_style = context.document.getElementById('custom-
568       style');
569     var graph = context.document.getElementById('graph');
570     while(!graph.svg_viewer) {
571       await this.jsl.non_blocking.waitMSeconds(1);
572     }
573     context.showGraph = async function(graph_definition) {
574       try {
575         var res = await context.mermaid.parse(graph_definition);
576         if(res) {
577           var { svg } = await context.mermaid.render('id1', graph_definition);
578           graph.innerHTML = svg;
579           graph.svg_viewer.attach();
580           graph.style.display = 'block';
581         }
582       } catch(err) {
583         error('@showGraph: '+err);
584       }
585     }
586     await context.showGraph(graph_definition);
587     return context;
588   }
589
590   /**
591    * Sets the specified window as the active window.
592    * @param {number} wid - The identifier of the window to set as active.
593    */
594   _setActiveWindow(wid) {

```

```

591     if (this.open_windows.hasOwnProperty(wid)) {
592         this.active_window = wid;
593     } else {
594         this.active_window = -1;
595     }
596 }
597
598 /**
599 * Updates the language of the text elements within all open windows.
600 */
601 _updateLanguage() {
602     Object.values(this.open_windows).forEach(function(win) {
603         win._updateLanguage(false);
604     });
605 }
606
607 /**
608 * Closes a window identified by the given ID and updates the active window
609 * if necessary.
610 * @param {number} wid - The identifier of the window to close.
611 */
612 _closedWindow(wid) {
613     if (this.open_windows.hasOwnProperty(wid)) {
614         var new_wid = -1;
615         if (this.active_window == wid) {
616             var wids = Object.keys(this.open_windows);
617             var N = wids.length;
618             if (N > 1) {
619                 if (wids[N-1] != wid) {
620                     new_wid = wids[N-1];
621                 } else {
622                     new_wid = wids[N-2];
623                 }
624                 this._ setActiveWindow(new_wid);
625             }
626             this.open_windows[wid].onClosed();
627             delete this.open_windows[wid];
628         }
629     }
630 }
631
632 exports.PRDC_JSLAB_LIB_WINDOWS = PRDC_JSLAB_LIB_WINDOWS;
633
634 /**
635 * Class for JSLAB window.
636 */
637 class PRDC_JSLAB_WINDOW {
638
639     #jsl;
640
641 /**
642 * Initializes a new instance of the JSLAB window.
643 * @param {Object} jsl - Reference to the main JSLAB object.
644 * @param {number} wid - Identifier for the window.

```



```
645  */
646 constructor(jsl, wid) {
647     var obj = this;
648
649     this.#jsl = jsl;
650     this.wid = wid;
651
652     this.context;
653     this.dom;
654
655     this.opened = false;
656     this.ready = new Promise((resolve) => {
657         obj._readyResolve = resolve;
658     });
659
660     this.onClosed = function() {};
661 }
662
663 /**
664 * Opens the window with the specified file or url.
665 * @param {string} file - The path to the HTML file or url to open in the
666 * window.
667 * @returns {Promise<void>} A promise that resolves when the window is
668 * opened and ready.
669 */
670 async open(file) {
671     if(!this.opened) {
672         var obj = this;
673         this.opened = true;
674         var [context, ready] = this.#jsl.env.openWindow(this.wid, file);
675         this.context = context;
676         this.context.getWindow = function() {
677             return obj;
678         }
679         await ready;
680         this.dom = this.context.document;
681         this._onReady();
682     }
683
684 /**
685 * Shows the window.
686 * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the window
687 * was shown successfully, or 'false' if the window ID is invalid.
688 */
689 async show() {
690     await this.#jsl.promiseOrStoped(this.ready);
691     return this.#jsl.env.showWindow(this.wid);
692 }
693
694 /**
695 * Hides the window.
696 * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the window
697 * was hidden successfully, or 'false' if the window ID is invalid.
698 */
699
```

```

696     async hide() {
697         await this.#jsl.promiseOrStoped(this.ready);
698         return this.#jsl.env.hideWindow(this.wid);
699     }
700
701     /**
702      * Brings focus to the window.
703      * @returns {Promise} - Resolves when the window size is focused.
704      */
705     async focus() {
706         await this.#jsl.promiseOrStoped(this.ready);
707         return this.#jsl.env.focusWindow(this.wid);
708     }
709
710     /**
711      * Minimizes the window.
712      * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the window
713      * was minimized successfully, or 'false' if the window ID is invalid.
714      */
715     async minimize() {
716         await this.#jsl.promiseOrStoped(this.ready);
717         return this.#jsl.env.minimizeWindow(this.wid);
718     }
719
720     /**
721      * Centers the window on the screen.
722      * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the window
723      * was centered successfully, or 'false' if the window ID is invalid.
724      */
725     async center() {
726         await this.#jsl.promiseOrStoped(this.ready);
727         return this.#jsl.env.centerWindow(this.wid);
728     }
729
730     /**
731      * Moves the window to the top of the window stack.
732      * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the window
733      * was moved to the top successfully, or 'false' if the window ID is
734      * invalid.
735      */
736     async moveTop() {
737         await this.#jsl.promiseOrStoped(this.ready);
738         return this.#jsl.env.moveTopWindow(this.wid);
739     }
740
741     /**
742      * Sets the size of the current window.
743      * @param {number} width - The desired width of the window.
744      * @param {number} height - The desired height of the window.
745      * @returns {Promise} - Resolves when the window size is set.
746      */
747     async setSize(width, height) {
748         await this.#jsl.promiseOrStoped(this.ready);
749         return this.#jsl.env.setWindowSize(this.wid, width, height);
750     }

```

```
747
748  /**
749   * Sets the position of the current window.
750   * @param {number} left - The desired left position of the window.
751   * @param {number} top - The desired top position of the window.
752   * @returns {Promise} - Resolves when the window position is set.
753   */
754   async setPos(left, top) {
755     await this.#jsl.promiseOrStoped(this.ready);
756     return this.#jsl.env.setWindowPos(this.wid, left, top);
757   }
758
759  /**
760   * Sets the resizable state of the window.
761   * @param {boolean} state - Whether the window should be resizable ('true')
762   * or not ('false').
763   * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the
764   * resizable state was set successfully, or 'false' if the window ID is
765   * invalid.
766   */
767   async setResizable(state) {
768     await this.#jsl.promiseOrStoped(this.ready);
769     return this.#jsl.env.setWindowResizable(this.wid, state);
770   }
771
772  /**
773   * Sets the movable state of the window.
774   * @param {boolean} state - Whether the window should be movable ('true')
775   * or not ('false').
776   * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the movable
777   * state was set successfully, or 'false' if the window ID is invalid.
778   */
779   async setMovable(state) {
780     await this.#jsl.promiseOrStoped(this.ready);
781     return this.#jsl.env.setWindowMovable(this.wid, state);
782   }
783
784  /**
785   * Sets the aspect ratio of the window.
786   * @param {number} aspect_ratio - The desired aspect ratio (width divided by
787   * height) for the window.
788   * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the aspect
789   * ratio was set successfully, or 'false' if the window ID is invalid.
790   */
791   async setAspectRatio(aspect_ratio) {
792     await this.#jsl.promiseOrStoped(this.ready);
793     return this.#jsl.env.setWindowAspectRatio(this.wid, aspect_ratio);
794   }
795
796  /**
797   * Sets the opacity of the window.
798   * @param {number} opacity - The desired opacity level of the window (
799   * ranging from '0' for fully transparent to '1' for fully opaque).
800   * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the opacity
801   * was set successfully, or 'false' if the window ID is invalid.
```

```

793  */
794  async setOpacity(opacity) {
795   await this.#jsl.promiseOrStoped(this.ready);
796   return this.#jsl.env.setWindowOpacity(this.wid, opacity);
797 }
798
799 /**
800 * Sets the fullscreen state of the window.
801 * @param {number} state - State of fullscreen
802 * @returns {Promise<boolean|undefined>} - Resolves to 'true' if the
803   fullscreen was set successfully, or 'false' if the window ID is invalid
804   .
805 /**
806  async setFullscreen(state) {
807   await this.#jsl.promiseOrStoped(this.ready);
808   return this.#jsl.env.setWindowFullscreen(this.wid, state);
809 }
810
811 /**
812 * Sets the title of the current window.
813 * @param {string} title - The new title for the window.
814 * @returns {Promise<*>} A promise that resolves when the title is set.
815 */
816  async setTitle(title) {
817   await this.#jsl.promiseOrStoped(this.ready);
818   return this.#jsl.env.setTitle(this.wid, title);
819 }
820
821 /**
822 * Prints the current window to PDF.
823 * @param {Object} options - Options for printing.
824 * @returns {Buffer} Generated PDF data.
825 */
826  async printToPdf(options) {
827   await this.#jsl.promiseOrStoped(this.ready);
828   return this.#jsl.env.printWindowToPdf(this.wid, options);
829 }
830
831 /**
832 * Retrieves the size of the current window.
833 * @returns {Promise<Array>} - Resolves with an array [width, height].
834 */
835  async getSize() {
836   await this.#jsl.promiseOrStoped(this.ready);
837   return this.#jsl.env.getWindowSize(this.wid);
838 }
839
840 /**
841 * Retrieves the position of the current window.
842 * @returns {Promise<Array>} - Resolves with an array [left, top].
843 */
844  async getPos() {
845   await this.#jsl.promiseOrStoped(this.ready);
846   return this.#jsl.env.getWindowPos(this.wid);
847 }

```



```
846
847  /**
848   * Closes the current window.
849   * @returns {Promise} - Resolves when the window is closed.
850   */
851  async close() {
852    await this.#jsl.promiseOrStoped(this.ready);
853    return this.#jsl.env.closeWindows(this.wid);
854  }
855
856  /**
857   * Opens the developer tools for the current window asynchronously.
858   * @returns {Promise<boolean>} A promise that resolves to true when dev
859   * tools are opened.
860   */
861  async openDevTools() {
862    await this.#jsl.promiseOrStoped(this.ready);
863    return this.#jsl.env.openWindowDevTools(this.wid);
864  }
865
866  /**
867   * Returns media source id for this window.
868   * @returns {String|boolean} Media source id if there is window; otherwise,
869   * false.
870   */
871  async getMediaSourceId() {
872    await this.#jsl.promiseOrStoped(this.ready);
873    return this.#jsl.env.getWindowMediaSourceId(this.wid);
874  }
875
876  /**
877   * Starts video recording of this window.
878   * @param {Object} - Optional settings.
879   * @returns {Object|boolean} - Returns recorder object.
880   */
881  async startVideoRecording(opts) {
882    await this.#jsl.promiseOrStoped(this.ready);
883    return await this.#jsl.device.startVideoRecording(this.#jsl.env.
884      getWindowMediaSourceId(this.wid), opts);
885  }
886
887  /**
888   * Appends a script to the document head.
889   * @param {string} script_path The script's URL.
890   */
891  addScript(script_path) {
892    const script = this.context.document.createElement("script");
893    script.src = script_path;
894    this.context.document.head.appendChild(script);
895  }
896
897  /**
898   * Appends a stylesheet link to the document head.
899   * @param {string} stylesheet_path The stylesheet's URL.
900   */
901
```

```

898 addLinkStylesheet(stylesheet_path) {
899     const link = this.context.document.createElement("link");
900     link.rel = "stylesheet";
901     link.href = stylesheet_path;
902     this.context.document.head.appendChild(link);
903 }
904
905 /**
906 * Loads the UI script.
907 */
908 addUI() {
909     this.addScript("../js/windows/ui.js");
910     this.addLinkStylesheet("../css/ui.css");
911 }
912
913 /**
914 * Handles actions to perform when the window is ready.
915 * @returns {void}
916 */
917 _onReady() {
918     var style = this.dom.createElement('style');
919     this.dom.head.appendChild(style);
920     this.lang_styles = style.sheet;
921     this.lang_styles.insertRule("lang { display: none; }", 0);
922     this._updateLanguage();
923     this._readyResolve(true);
924 }
925
926 /**
927 * Updates the language of the text elements within the DOM.
928 * @param {boolean} [flag=true] - Whether to update the language.
929 * @returns {void}
930 */
931 _updateLanguage(flag = true) {
932     if(flag) {
933         this.dom.querySelectorAll('str').forEach(function(el) {
934             var id = el.getAttribute('sid');
935             el.innerHTML = language.string(id);
936         });
937     }
938
939     if(this.lang_styles.cssRules.length > 1) {
940         this.lang_styles.deleteRule(1);
941     }
942     this.lang_styles.insertRule("lang."+language.lang+" { display: initial }",
943         1);
944
945     this.dom.querySelectorAll('[title-str]').forEach(function(el) {
946         var id = el.getAttribute('title-str');
947         if(id in language.s) {
948             el.setAttribute('title', language.s[id][language.lang]);
949         }
950     });
951
952     this.dom.querySelectorAll('textarea[str]').forEach(function(el) {

```

```

952     var id = el.getAttribute('str');
953     if(id in language.s) {
954         el.setAttribute('placeholder', language.s[id][language.lang]);
955     }
956 });
957
958 this.dom.querySelectorAll('input[str]').forEach(function(el) {
959     var id = el.getAttribute('str');
960     if(id in language.s) {
961         el.setAttribute('placeholder', language.s[id][language.lang]);
962     }
963 });
964
965 this.dom.querySelectorAll('option[str]').forEach(function(el) {
966     var id = el.getAttribute('str');
967     if(id in language.s) {
968         el.textContent = language.s[id][language.lang];
969     }
970 });
971 }
972 }
```

Listing 131 - windows.js

6.6 windows

```

1 window.MathJax = {
2     tex: {
3         macros: {
4             bm: ["\\mathbf{#1}", 1], // \bm{X} \mathbf{X}
5             norm: ["\\left\\| #1 \\right\\|", 1],
6             abs: ["\\left\\| #1 \\right\\|", 1],
7             diff: ["\\mathrm{d}\\#1", 1]
8         }
9     },
10    startup: {
11        typeset: false
12    }
13};
```

Listing 132 - mathjax-config.js

```

1 /**
2 * @file JSLAB library plot script
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 /**
9 * Class for JSLAB plot.
10 */
11 class PRDC_JSLAB_PLOT {
12
13    /**

```

```

14  * Initializes an instance of the PRDC_JSLAB_PLOT class.
15  */
16 constructor() {
17   var obj = this;
18   this.figure_content = document.getElementById('figure-content');
19   this.plot_cont;
20
21   this.update_queue = new Map();
22   this.frame_requested = false;
23 }
24
25 /**
26 * Set a new plot container to the figure content.
27 */
28 setCont() {
29   this.plot_cont = document.createElement('div');
30   this.plot_cont.className = 'plot-cont';
31   this.figure_content.appendChild(this.plot_cont);
32 }
33
34 /**
35 * Creates a new plot using Plotly in the plot container.
36 * @param {Object} plot_in - An object containing plot initial settings.
37 * @param {Array} traces - Data traces for the plot.
38 * @param {Object} layout - Layout configuration for the plot.
39 * @param {Object} config - Additional configuration settings for the plot.
40 */
41 async newPlot(plot_in, traces, layout, config) {
42   await Plotly.newPlot(this.plot_cont, traces, layout, config);
43 }
44
45 /**
46 * Queues trace-update objects for the next animation-frame batch flush.
47 * @param {Object} traces - The trace data to apply styling changes to.
48 * @param {number} N - The index or length of data for styling adjustments.
49 */
50 updateData(traces, N) {
51   var obj = this;
52   traces = Array.isArray(traces) ? traces : [traces];
53   var idxs = typeof N === 'undefined'
54     ? this.plot_cont.data.map((_, i) => i)
55     : (Array.isArray(N) ? N : [N]);
56
57   idxs.forEach(function(idx, k) {
58     if(idx === null || Number.isNaN(idx)) return;
59     var update = (traces.length === idxs.length) ? traces[k] : traces[0];
60     obj.update_queue.set(idx, { ...obj.update_queue.get(idx), ...update });
61   });
62
63   if(!this.frame_requested) {
64     this.frame_requested = true;
65     requestAnimationFrame(function() {
66       obj._flushUpdates();
67     });
68   }
}

```



```

122
123     if(idx !== -1 && obj.plot_cont.data[idx]) {
124         var slot = restyle_indices.push(idx) - 1;
125         for(var prop in flat) {
126             if(!restyle_props[prop]) restyle_props[prop] = [];
127             var v = flat[prop];
128             restyle_props[prop][slot] =
129                 Array.isArray(v) && v.length === 1 && Array.isArray(v[0]) ? v[0] :
130                 v;
131         }
132     });
133
134     if(restyle_indices.length) {
135         Plotly.restyle(this.plot_cont, restyle_props, restyle_indices);
136     }
137     this.update_queue.clear();
138 }
139
140 /**
141 * Updates the layout of the existing plot.
142 * @param {Object} layout - Layout configuration to apply to the plot.
143 */
144 relayout(layout) {
145     Plotly.relayout(this.plot_cont, layout);
146 }
147
148 /**
149 * Restyles the plot with updated trace data.
150 * @param {Object} traces - The trace data to apply styling changes to.
151 * @param {number} N - The index or length of data for styling adjustments.
152 */
153 restyle(traces, N) {
154     Plotly.restyle(this.plot_cont, traces, N);
155 }
156
157 /**
158 * Resizes the plot to fit its container.
159 */
160 resize() {
161     Plotly.Plots.resize(this.plot_cont);
162 }
163
164 /**
165 * Converts the plot to an image.
166 * @param {string} ext - The image format extension (e.g., 'png', 'jpeg').
167 * @param {Array} size - Optional size of image.
168 * @returns {String} Image of the plot.
169 */
170 async toImage(ext, size) {
171     var width = this.plot_cont.clientWidth;
172     var height = this.plot_cont.clientHeight;
173     if(typeof size !== 'undefined') {
174         width = size[0];
175         height = size[1];

```

```

176     }
177     return await Plotly.toImage(this.plot_cont, {format: ext,
178         width: width,
179         height: height
180     });
181 }
182
183 /**
184 * Converts the plot to an json object.
185 * @param {Function} callback - Callback function that handles the image URL
186
187 * @returns {Object} JSON of the plot.
188 */
189 toJSON() {
190     return Plotly.Plots.graphJson(this.plot_cont,
191         undefined, undefined, undefined, true);
192 }
193
194 /**
195 * Sets the plot data from JSON.
196 * @param {Array} data Data for the plot.
197 */
198 async fromJSON(data) {
199     await Plotly.newPlot(this.plot_cont, data);
200 }
201
202 var plot = new PRDC_JSLAB_PLOT();

```

Listing 133 - plot.js

```

1 /**
2 * @file JSLAB library presentation editor script
3 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4 * PR-DC, Republic of Serbia
5 * info@pr-dc.com
6 */
7
8 const { ipcRenderer } = require('electron');
9
10 const fs = require('fs');
11 const path = require('path');
12 const { ESLint } = require("eslint");
13 require("../../../js/init-config.js");
14
15 const { PRDC_POPUP } = require('../../../lib/PRDC_POPUP/PRDC_POPUP');
16 const { PRDC_PANEL } = require('../../../lib/PRDC_PANEL/PRDC_PANEL');
17
18 /**
19 * Class for JSLAB presentation editor code tab.
20 */
21 class PRDC_JSLAB_PRESENTATION_EDITOR_CODE_TAB {
22
23 /**
24 * Initializes an instance of the PRDC_JSLAB_PRESENTATION_EDITOR class.
25 */

```

```

26  constructor(editor, name, file) {
27    var obj = this;
28    this.editor = editor;
29
30    this.tab = this.editor.tabs.addTab({
31      title: name,
32      favicon: false
33  });
34  this.tab.tab_obj = this;
35
36  this.code_editor = CodeMirror(document.getElementById("code"), {
37    theme: "notepadpp",
38    rulers: [{ color: "#aff", column: 75, lineStyle: "solid" }],
39    indentUnit: 2,
40    tabSize: 2,
41    lineNumbers: true,
42    lineWrapping: true,
43    styleActiveLine: true,
44    matchBrackets: true,
45    gutter: true,
46    gutters: [
47      "CodeMirror-linenumbers",
48      "CodeMirror-foldgutter",
49      "CodeMirror-lint-markers",
50    ],
51    foldGutter: true,
52    searchDialog: true,
53    highlightSelectionMatches: { annotateScrollbar: true },
54  });
55
56  CodeMirror.keyMap.default["Shift-Tab"] = "indentLess";
57  CodeMirror.keyMap.default["Tab"] = "indentMore";
58  CodeMirror.keyMap.default["Ctrl-F"] = "showSearchDialog";
59  CodeMirror.keyMap.default['Ctrl-G'] = 'findNext';
60  CodeMirror.keyMap.default['Shift-Ctrl-G'] = 'findPrev';
61  CodeMirror.keyMap.default['Shift-Ctrl-F'] = 'replace';
62  CodeMirror.keyMap.default['Shift-Ctrl-R'] = 'replaceAll';
63  CodeMirror.keyMap.default['Ctrl-/'] = 'toggleComment';
64
65  // Keypress events
66  this.code_editor.on("keypress", function(cm, event) {
67    if(
68      !cm.state.completionActive &&
69      !event.ctrlKey &&
70      event.key != "Enter" &&
71      event.key != ";" &&
72      event.key != " " &&
73      (event.key != "{" & (event.key != "}"))
74    ) {
75      CodeMirror.commands.autocomplete(cm, null, { completeSingle: false });
76    }
77  });
78
79  this.code_editor.getInputField().setAttribute('title', name);
80

```

```

81   if(name === 'html') {
82     this.code_editor.setOption("mode", "htmlemixed");
83     this.code_editor.setOption("lint", {});
84
85   function collectSlideRanges(source) {
86     const ranges = [];
87     const re = /<\s*slide\b[^>]*>([\s\S]*?)<\/\s*slide\s*>/gi;
88     let m;
89     while ((m = re.exec(source))) ranges.push({ start: m.index, end: re.lastIndex });
90     return ranges;
91   }
92
93   function cursorSlideIndex(pos, ranges) {
94     for(let i = 0; i < ranges.length; i++)
95       if(pos >= ranges[i].start && pos <= ranges[i].end) return i;
96     return -1;
97   }
98
99   this.getSlide = function() {
100    var pos = obj.code_editor.indexFromPos(obj.code_editor.getCursor());
101    var txt = obj.code_editor.getValue();
102    return cursorSlideIndex(pos, collectSlideRanges(txt));
103  }
104  this.setSlide = function(index) {
105    var txt = obj.code_editor.getValue();
106    var rngs = collectSlideRanges(txt);
107    if(index >= 0 && index < rngth.length) {
108      var r = rngth[index];
109      var gt = txt.indexOf('>', r.start);
110      var offset = gt !== -1 && gt < r.end ? gt + 1 : r.start;
111      var pos = obj.code_editor.posFromIndex(offset);
112      obj.code_editor.setCursor(pos);
113      obj.code_editor.scrollIntoView({ line: pos.line, ch: pos.ch }, 80)
114    }
115  }
116  this.foldSlides = function() {
117    var cursor = obj.code_editor.getSearchCursor(/<\s*slide\b[^>]*>/ig,
118                                                 CodeMirror.Pos(0, 0));
119    obj.code_editor.operation(() => {
120      while(cursor.findNext()) {
121        const pos = cursor.from();
122        obj.code_editor.foldCode(pos, null, 'fold');
123      }
124    })
125  this.unfoldSlides = function() {
126    var cursor = obj.code_editor.getSearchCursor(/<\s*slide\b[^>]*>/ig,
127                                                 CodeMirror.Pos(0, 0));
128    obj.code_editor.operation(() => {
129      while(cursor.findNext()) {
130        const pos = cursor.from();
131        obj.code_editor.foldCode(pos, null, 'unfold');
132      }
133    })
134  }

```

```

133     }
134   } else if(name === 'css') {
135     this.code_editor.setOption("mode", "css");
136     this.code_editor.setOption("lint", {});
137   } else if(name === 'js') {
138     this.code_editor.setOption("mode", "javascript");
139     this.code_editor.setOption("lint", {
140       getAnnotations: async function(text, callback) {
141         var results = await obj.editor.eslint.lintText(text);
142         callback(results[0].messages.map(message => ({
143           from: CodeMirror.Pos(message.line - 1, message.column - 1),
144           to: CodeMirror.Pos(
145             message.endLine ? message.endLine - 1 : message.line - 1,
146             message.endColumn ? message.endColumn - 1 : message.column
147           ),
148           severity: message.severity === 2 ? "error" : "warning",
149           message: message.message,
150         ))));
151       },
152       async: true
153     });
154   }
155 }

156 /**
157 * On code changed
158 */
159 codeChanged() {
160   this.tab.classList.add("changed");
161 }

162 /**
163 * Activates this tab and shows code
164 */
165 show() {
166   $(".CodeMirror").hide();
167   this.editor.tabs.setCurrentTab(this.tab);
168   $(this.code_editor.display.wrapper).show();
169   this.code_editor.focus();
170   this.code_editor.refresh();
171 }

172 /**
173 * Save code
174 */
175 save() {
176   if(this.tab.classList.contains("changed")) {
177     this.tab.classList.remove("changed");
178     fs.writeFileSync(this.file_path, this.code_editor.getValue())
179   }
180 }

181 /**
182 * Opens search dialog in the code editor.
183 */
184
185 /**
186 * Opens search dialog in the code editor.
187 */

```

```

188 openSearchDialog() {
189     this.code_editor.execCommand('showSearchDialog');
190 }
191 /**
192 * Sets file path for code
193 * @param {String} file_path - Absolute path to the code file
194 */
195 setPath(file_path) {
196     var obj = this;
197     this.file_path = file_path;
198     this.code = fs.readFileSync(file_path).toString();
199     this.code_editor.setValue(this.code);
200     this.code_editor.clearHistory();
201
202     this.code_editor.on("change", function() {
203         obj.codeChanged();
204     });
205 }
206 }
207 }
208 /**
209 * Class for JSLAB presentation editor.
210 */
211 class PRDC_JSLAB_PRESENTATION_EDITOR {
212
213 /**
214 * Initializes an instance of the PRDC_JSLAB_PRESENTATION_EDITOR class.
215 */
216 constructor() {
217     var obj = this;
218
219     this.webview = document.getElementById('preview');
220     this.webview_wrap = document.getElementById('webview-wrap');
221     this.left = document.getElementById('left-panel-cont');
222
223     this.eslint = new ESLint(config.LINT_OPTIONS);
224     this.editor_more_popup = new PRDC_POPUP(document.getElementById('editor-more-icon'),
225         document.getElementById('editor-more-popup'));
226     this.current_slide = 0;
227     this.total_slides = 0;
228
229     this.columns = new PRDC_PANEL('presentation-editor-columns', 'vertical',
230         document.body, [document.getElementById('left-panel'), document.
231             getElementById('right-panel')], [60, 40], function() {
232         obj.scaleSlide();
233     });
234
235     // Initialize panels
236     window.addEventListener('resize', function() {
237         obj.columns.onResize();
238     });
239     // Tabs

```

```

240   this.tabs_cont = document.querySelector(".tabs");
241   this.tabs = new PRDC_TABS();
242   this.tabs.init(this.tabs_cont);
243
244   // On tab change
245   this.tabs_cont.addEventListener("activeTabChange", function({ detail }) {
246     obj.active_tab = detail.tabEl;
247     if(obj.active_tab.hasOwnProperty('tab_obj')) {
248       obj.active_tab.tab_obj.show();
249     }
250   });
251
252   this.html_editor = new PRDC_JSLAB_PRESENTATION_EDITOR_CODE_TAB(this, 'html');
253   this.js_editor = new PRDC_JSLAB_PRESENTATION_EDITOR_CODE_TAB(this, 'js');
254   this.css_editor = new PRDC_JSLAB_PRESENTATION_EDITOR_CODE_TAB(this, 'css');
255
256   this.html_editor.show();
257
258   document.addEventListener("keydown", function(e) {
259     if(e.ctrlKey && e.key.toLowerCase() === "s" && !e.shiftKey) {
260       obj.saveCode();
261     }
262   });
263
264   this.webview.addEventListener('ipc-message', (e) => {
265     if(e.args[0].ready !== undefined) {
266       obj.total_slides = e.args[0].ready;
267       document.getElementById('total-slides').innerText = '/' + obj.total_slides;
268     } else if(e.args[0].slide !== undefined) {
269       obj.current_slide = e.args[0].slide;
270       document.getElementById('set-slide').value = obj.current_slide + 1;
271     }
272   });
273
274   $("#tab-save").click(function() { obj.saveCode(); });
275   $("#search-dialog-menu").click(function() {
276     obj.active_tab.tab_obj.openSearchDialog();
277     obj.editor_more_popup.close();
278   });
279   $("#fold-slides").click(function() {
280     obj.html_editor.show();
281     obj.html_editor.foldSlides();
282     obj.editor_more_popup.close();
283   });
284   $("#unfold-slides").click(function() {
285     obj.html_editor.show();
286     obj.html_editor.unfoldSlides();
287     obj.editor_more_popup.close();
288   });
289   $("#first-slide").click(function() {
290     obj.webview.send('data',{ show: 0 });
291

```

```

292 });
293 $("#prev-slide").click(function() {
294   obj.webview.send('data',{ show: obj.current_slide - 1 });
295 });
296 $("#next-slide").click(function() {
297   obj.webview.send('data',{ show: obj.current_slide + 1 });
298 });
299 $("#last-slide").click(function() {
300   obj.webview.send('data',{ show: obj.total_slides - 1 });
301 });
302 $("#set-slide").on("change", function() {
303   obj.webview.send('data',{ show: $(this).val() - 1 });
304 });

305 // On IPC message
306 ipcRenderer.on("PresentationEditorWindow", function(event, action, data) {
307   switch(action) {
308     case "go-to-code":
309       obj.html_editor.show();
310       obj.html_editor.setSlide(obj.current_slide);
311       break;
312     case "go-to-slide":
313       obj.webview.send('data', { show: obj.html_editor.getSlide() });
314       break;
315   }
316 }
317 );
318 }

319 /**
320 * Sets file path for presentation editor
321 * @param {String} file_path - Absolute path to the presentation directory.
322 */
323 setPath(file_path, url) {
324   var obj = this;
325   this.file_path = file_path;
326   this.url = url;
327   var name = path.basename(file_path);

328   this.exe_file = name;
329   this.html_editor.setPath(path.join(file_path, 'index.html'));
330   this.js_editor.setPath(path.join(file_path, 'main.js'));
331   this.css_editor.setPath(path.join(file_path, 'main.css'));

332   document.getElementById('presentation-title').innerText = name;
333   this.webview.src = url+'?lazy';
334   this.presentation_config = JSON.parse(fs.readFileSync(path.join(file_path,
335     'res/internal/config.json')).toString());
336
337   // Slide scale
338   this.webview.style.width = this.presentation_config.slide_width + 'px';
339   this.webview.style.height = this.presentation_config.slide_height + 'px';
340   this.scaleSlide();
341
342 }

343 /**
344 */

```



```
346     * Saves code and triggers frame update
347     */
348     saveCode() {
349         this.html_editor.save();
350         this.js_editor.save();
351         this.css_editor.save();
352         this.webview.reload();
353     }
354
355     /**
356     * Scale slide
357     */
358     scaleSlide() {
359         const scale = Math.min((this.left.clientWidth - 20) / this.
360             presentation_config.slide_width, (this.left.clientHeight - 20) / this.
361             presentation_config.slide_height);
362         this.webview.style.transform = `scale(${scale})`;
363         this.webview_wrap.style.height = `${this.presentation_config.slide_height
364             * scale}px`;
365     }
366 }
367
368 var presentation_editor = new PRDC_JSLAB_PRESENTATION_EDITOR();
```

Listing 134 - presentation-editor.js

```
1 /**
2  * @file JSLAB library presentation script
3  * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4  * PR-DC, Republic of Serbia
5  * info@pr-dc.com
6  */
7
8 window.file_buffers = {};
9 var has_node = typeof window.process === 'object' &&
10    !(window.process.versions && window.process.versions.electron);
11 if(has_node) {
12     var { ipcRenderer } = require('electron');
13 }
14
15 var is_iframe = window.parent !== window;
16 var is_lazy = new URLSearchParams(location.search).has('lazy');
17
18 /**
19  * Stores file buffer.
20  * @param {string} file_path The key for the file.
21  * @param {ArrayBuffer} data The file contents.
22  */
23 window.registerFile = function(file_path, data) {
24     window.file_buffers[file_path] = atob(data);
25 }
26
27 /**
28  * Class for JSLAB presentation.
29  */
30 class PRDC_JSLAB_PRESENTATION {
```

```

31
32  /**
33   * Initializes an instance of the PRDC_JSLAB_PRESENTATION class.
34   */
35 constructor() {
36   var obj = this;
37
38   this.config = %presentation_config%;
39   this.slides = document.querySelectorAll('slide');
40   this.slides_cont = document.getElementById('slides-cont');
41   this.current_slide = -1;
42   this.total_slides = this.slides.length;
43   this._interpolated = new WeakSet();
44   this._standalone = window.location.protocol === 'file:';
45   this._animating = false;
46
47   const style = document.createElement('style');
48   style.textContent =
49   '@media print {
50     slide {
51       display: block !important;
52       width: ${this.config.slide_width}px;
53       height: ${this.config.slide_height}px;
54     }
55   }';
56   document.head.appendChild(style);
57
58   this._buildSlideNav();
59
60   this._validTransitions = new Set([
61     'none', 'fade', 'zoom',
62     'cover', 'uncover',
63     'flip', 'flip-x', 'flip-y',
64     'slide', 'slide-left', 'slide-right', 'slide-up', 'slide-down',
65     'cube', 'cube-left', 'cube-right', 'cube-up', 'cube-down',
66     'push', 'push-left', 'push-right', 'push-up', 'push-down',
67   ]);
68   this.transition = this.slides_cont.getAttribute('transition') || 'fade';
69   if(!this._validTransitions.has(this.transition)) this.transition = 'fade';
70
71   document.addEventListener('keydown', (event) => {
72     if(event.ctrlKey && event.key.toLowerCase() === 's'){
73       event.preventDefault();
74       this._toggleSlideNav();
75     }
76     switch(event.key) {
77       case 'ArrowRight':
78       case 'PageDown':
79         this._lastNavKey = 'right';
80         obj.nextSlide();
81         break;
82
83       case 'ArrowLeft':
84       case 'PageUp':
85

```

```

86         this._lastNavKey = 'left';
87         obj.prevSlide();
88         break;
89
90     case 'ArrowDown':
91         this._lastNavKey = 'down';
92         obj.nextSlide();
93         break;
94
95     case 'ArrowUp':
96         this._lastNavKey = 'up';
97         obj.prevSlide();
98         break;
99     case 'F11':
100        if(has_node) {
101            ipcRenderer.sendToHost('data', { key: 'F11' });
102        }
103        if(is_iframe) {
104            window.parent.postMessage({ key: 'F11' }, '*');
105        }
106        break;
107    }
108);
109
110 const WHEEL_DEBOUNCE = 250;
111 var wheelGuard = false;
112 document.addEventListener('wheel', (event) => {
113     if(event.target.closest('.js-plotly-plot') ||
114         event.target.closest('input.ui') ||
115         event.target.closest('scene-3d-json')) return;
116     if(Math.abs(event.deltaY) < 10 || wheelGuard) {
117         return;
118     }
119
120     if(event.deltaY > 0) {
121         this._lastNavKey = 'down';
122         obj.nextSlide();
123     } else {
124         this._lastNavKey = 'up';
125         obj.prevSlide();
126     }
127
128     wheelGuard = true;
129     setTimeout(() => (wheelGuard = false), WHEEL_DEBOUNCE);
130 }, { passive: true });
131
132 window.addEventListener('DOMContentLoaded', () => {
133     var m = window.location.hash.match(/^#\s(\d+)$/);
134     var wanted = m ? parseInt(m[1], 10) - 1 :
135         this.current_slide > -1 ? this.current_slide : 0;
136     this.setSlide(wanted);
137 })
138
139 window.addEventListener('hashchange', () => {
140     const m = location.hash.match(/^#\s(\d+)$/);

```

```

141   if (!m) return;
142   const idx = parseInt(m[1], 10) - 1;
143   if (idx !== this.current_slide) this.showSlide(idx);
144 }
145
146 window.addEventListener('beforeprint', () => {
147   obj._interpolateAllSlides();
148 })
149
150 window.addEventListener('message', (e) =>{
151   if(typeof e.data.set === 'number'){
152     obj.setSlide(e.data.set);
153   } else if(typeof e.data.show === 'number'){
154     obj.showSlide(e.data.show);
155   }
156 });
157
158 if(has_node){
159   ipcRenderer.on('data', (e, data) => {
160     if(typeof data.set === 'number'){
161       obj.setSlide(data.set);
162     } else if(typeof data.show === 'number'){
163       obj.showSlide(data.show);
164     }
165   });
166
167   ipcRenderer.sendToHost('data', { ready: this.total_slides });
168 }
169
170 this.slides_cont.style.width = this.config.slide_width + 'px';
171 this.slides_cont.style.height = this.config.slide_height + 'px';
172
173 function scaleSlides() {
174   const scale = Math.min(document.body.clientWidth / obj.config.
175     slide_width,
176     document.body.clientHeight / obj.config.slide_height);
177   obj.slides_cont.style.transform = `scale(${scale}) translate(-50%, -50%)`;
178 }
179 window.addEventListener('resize', function(e) {
180   scaleSlides();
181 });
182 scaleSlides();
183
184 if(!this._standalone) {
185   const ping = () => fetch('/keepalive',
186     { method: 'HEAD', cache: 'no-store', mode: "no-cors" })
187   .then((data) => {}).catch((err) => {
188     console.log(err);
189   });
190   ping();
191   setInterval(ping, 10_000);
192 }
193

```

```

194     this._attachGestureControl();
195     if(!is_lazy && window.MathJax && typeof MathJax.typesetPromise === 'function') {
196       MathJax.typesetPromise();
197     }
198   }
199
200 /**
201 * Shows the slide at the supplied zero-based index.
202 * @param {number} index      Index of the <slide> element to activate.
203 */
204 setSlide(index) {
205   if(this._animating) this._stopAllAnimations();
206   if(index === this.current_slide) return;
207   if(index < 0 || index >= this.slides.length) return;
208   this.slides.forEach((slide, i) => {
209     slide.classList.toggle('active', i === index);
210     if(i === index) {
211       slide.style.display = 'block';
212     } else {
213       slide.style.display = 'none';
214     }
215   });
216   this.current_slide = index;
217
218   this._updateSlideNav();
219
220   const active = this.slides[index];
221   if(!this._interpolated.has(active)) {
222     this._interpolateSlide(active);
223     this._interpolated.add(active);
224   }
225
226   this._updateHash(index);
227   if(has_node) {
228     ipcRenderer.sendToHost('data', { slide: index });
229   }
230   if(is_iframe) {
231     window.parent.postMessage({ slide: index }, '*');
232   }
233   if(is_lazy) {
234     this._lazyRender(this.slides[index]);
235   }
236 }
237
238 /**
239 * Shows the slide at the supplied zero-based index with animation.
240 * @param {number} index      Index of the <slide> element to activate.
241 */
242 showSlide(index) {
243   if(this._animating) return;
244   if(index === this.current_slide) return;
245   if(index < 0 || index >= this.slides.length) return;
246
247   var outgoing = this.slides[this.current_slide];

```

```
248 var incoming = this.slides[index];
249
250 var slideOverride = incoming.getAttribute('transition');
251 var base = slideOverride || this.transition;
252
253 var forward = index > this.current_slide;
254 var navKey = this._lastNavKey || (forward ? 'right' : 'left');
255 var t = this._resolveDir(base, navKey, forward);
256
257 if(t == 'none') {
258   if(outgoing) {
259     outgoing.classList.remove('active');
260     outgoing.style.display = 'none';
261   }
262
263   incoming.style.display = 'block';
264   incoming.classList.add('active');
265 } else {
266   this._animating = true;
267
268   if(outgoing) {
269     outgoing.classList.remove('active');
270     outgoing.classList.add('slide-out', `${t}-out`);
271
272     outgoing.addEventListener('animationend', () => {
273       outgoing.classList.remove('slide-out', `${t}-out`);
274       outgoing.style.display = 'none';
275     }, { once: true });
276   }
277
278   incoming.style.display = 'block';
279   incoming.classList.add('slide-in', `${t}-in`, 'active');
280
281   incoming.addEventListener('animationend', () => {
282     incoming.classList.remove('slide-in', `${t}-in`);
283     this._animating = false;
284   }, { once: true });
285 }
286
287 this.current_slide = index;
288
289 this._updateSlideNav();
290
291 if(!this._interpolated.has(incoming)) {
292   this._interpolateSlide(incoming);
293   this._interpolated.add(incoming);
294 }
295
296 this._updateHash(index);
297 if(has_node) {
298   ipcRenderer.sendToHost('data', { slide: index });
299 }
300 if(is_iframe) {
301   window.parent.postMessage({ slide: index }, '*');
302 }
```

```

303     if(is_lazy) {
304         this._lazyRender(this.slides[index]);
305     }
306 }
307
308 /**
309 * Advances to the next slide (no-op if already on the last one).
310 */
311 nextSlide() {
312     this.showSlide(this.current_slide + 1);
313 }
314
315 /**
316 * Goes back to the previous slide (no-op if already on the first one).
317 */
318 prevSlide() {
319     this.showSlide(this.current_slide - 1);
320 }
321
322 /**
323 * Sets transition animation by name
324 * @param {string} name      transition animation name.
325 */
326 setTransition(name) {
327     if(this._validTransitions.has(name)) this.transition = name;
328 }
329
330 /**
331 * Returns the current slides position as a human-friendly 1-based index.
332 * @returns {number} The 1-based index of the slide that is currently active
333 *
334 */
335 slideNumber() {
336     return this.current_slide + 1;
337 }
338
339 /**
340 * Returns the total number of slides in the presentation.
341 * @returns {number} The total count of <slide> elements detected at startup
342 *
343 */
344 slideCount() {
345     return this.total_slides;
346 }
347
348 /**
349 * Returns size in pixels based on input string
350 * @param {string} Size in vw, vh or % format.
351 * @returns {number} Size in pixels.
352 */
353 toPixels(str, ref) {
354     if(!str) return 0;
355     if(str.endsWith('vw')) return this.config.slide_width * parseFloat(str) /
356         100;
357     if(str.endsWith('vh')) return this.config.slide_height * parseFloat(str) /

```

```

100;
355  if(str.endsWith('%')) {
356    if(ref == 'width') {
357      return this.config.slide_width * parseFloat(str) / 100;
358    } else if(ref == 'height') {
359      return this.config.slide_height * parseFloat(str) / 100;
360    }
361  }
362  return parseFloat(str);
363}
364
365 /**
366 * Returns when global variable becomes defined
367 * @param {string} prop - Name of variable
368 */
369 async waitForGlobal(prop) {
370   if(!window[prop]) {
371     await new Promise(resolve => {
372       const check = () => {
373         if(window[prop]) return resolve();
374         requestAnimationFrame(check);
375       };
376       check();
377     });
378   }
379 }
380
381 /**
382 * Builds slides navigation.
383 */
384 _buildSlideNav() {
385   var html =
386     <div id="first-slide" class="button" title="First slide"> </div>
387     <div id="prev-slide" class="button" title="Previous"> </div>
388     <input id="set-slide" type="number" min="1" step="1">
389     <span id="total-slides">/ 0</span>
390     <div id="next-slide" class="button" title="Next"> </div>
391     <div id="last-slide" class="button" title="Last slide"> </div>;
392
393   this.slide_nav = document.createElement('div');
394   this.slide_nav.id = 'slide-controls';
395   this.slide_nav.innerHTML = html;
396   this.slide_nav.hidden = true;
397   document.body.appendChild(this.slide_nav);
398
399   this.slide_nav.querySelector('#first-slide').onclick = ()=> this.setSlide(0);
400   this.slide_nav.querySelector('#prev-slide').onclick = ()=> this.prevSlide();
401   this.slide_nav.querySelector('#next-slide').onclick = ()=> this.nextSlide();
402   this.slide_nav.querySelector('#last-slide').onclick = ()=> this.setSlide(this.total_slides - 1);
403
404   this.slide_nav_input = this.slide_nav.querySelector('input');

```

```

405     this.slide_nav_input.onchange = e => {
406       this.setSlide((+this.slide_nav_input.value || 1) - 1);
407     };
408     this.slide_nav_total = this.slide_nav.querySelector('#total-slides');
409
410     this._updateSlideNav();
411   }
412
413   /**
414    * Builds slides navigation.
415    */
416   _toggleSlideNav() {
417     this.slide_nav.hidden = !this.slide_nav.hidden;
418     if(!this.slide_nav.hidden){
419       this.slide_nav_input.focus();
420       this.slide_nav_input.select();
421     }
422   }
423
424   /**
425    * Builds slides navigation.
426    */
427   _updateSlideNav() {
428     this.slide_nav_input.value = this.slideNumber();
429     this.slide_nav_total.textContent = `/ ${this.total_slides}`;
430   }
431
432   /**
433    * Synchronises the URL hash with the currently visible slide.
434    * Uses history.replaceState so it never clutters the browser history.
435    * @param {number} idx      zero-based slide index.
436    */
437   _updateHash(idx) {
438     const newHash = `#${idx + 1}`;
439     if(location.hash !== newHash) {
440       history.replaceState(null, '', newHash);
441     }
442   }
443
444   /**
445    * Lazy-render MathJax, <img-pdf> and <plot-json> elements that
446    * live inside the currently visible slide.
447    * @param {HTMLElement} slide      the active <slide> element
448    */
449   _lazyRender(slide) {
450     if(!slide) return;
451     slide.querySelectorAll('img-pdf, plot-json, scene-3d-json').forEach(el =>
452       {
453         if(el._lazyRendered) return;
454         if(typeof el._render === 'function') {
455           el._lazyRendered = true;
456           el._render();
457         }
458       });
459   }

```

```

459     if (window.MathJax && typeof MathJax.typesetPromise === 'function') {
460       MathJax.typesetPromise([slide]).catch(err => console.error(err));
461     }
462   }
463
464   /**
465    * Replaces every ${expr} text placeholder inside *root*
466    * with the evaluated value of *expr* in the window scope.
467    * @param {HTMLElement} root      Slide element to interpolate.
468    */
469   _interpolateSlide(root) {
470     const AVOID = new Set(['SCRIPT', 'STYLE']);
471     const re = /\$\{([^\}]*)\}/g;
472
473     const walker = document.createTreeWalker(
474       root,
475       NodeFilter.SHOW_TEXT,
476       {
477         acceptNode: n =>
478           AVOID.has(n.parentNode.tagName)
479             ? NodeFilter.FILTER_REJECT
480             : NodeFilter.FILTER_ACCEPT
481       }
482     );
483
484     for (let node; (node = walker.nextSibling()); ) {
485       const src = node.nodeValue;
486       if (!re.test(src)) continue;
487       re.lastIndex = 0;
488
489       const out = src.replace(re, (_, expr) => {
490         try {
491           const fn = new Function('with (window) { return (${expr}); }');
492           const val = fn.call(window);
493           return val === null ? '' : String(val);
494         } catch (err) {
495           return `\\${expr}`;
496         }
497       });
498
499       if (out !== src) node.nodeValue = out;
500     }
501   }
502
503   /**
504    * Calls _interpolateSlide method for each slide
505    */
506   _interpolateAllSlides() {
507     var current_slide = this.current_slide;
508     for (var i = 0; i < this.slideCount(); i++) {
509       this.setSlide(i);
510     };
511     this.setSlide(current_slide);
512   }
513

```

```

514  /**
515   * Removes all -in/-out classes from slide and hard-stops its
516   * animation.
517   * @param {HTMLElement} el      slide element.
518   */
519  _clearAnimClasses(el) {
520    el.classList.forEach(c => {
521      if(c.endsWith('-in') || c.endsWith('-out')) el.classList.remove(c);
522    });
523    el.style.animation = 'none';
524    void el.offsetWidth;
525    el.style.animation = '';
526  }
527  /**
528   * Stops all slides that may be mid-transition.
529   */
530  _stopAllAnimations() {
531    this.slides.forEach(s => this._clearAnimClasses(s));
532    this._animating = false;
533  }
534  /**
535   * Expands a base transition into a direction-specific variant.
536   * @param base  {string} e.g. "slide", "flip", "cover"
537   * @param navKey {string} "left" | "right" | "up" | "down"
538   * @param forward {boolean} true when index grows
539   */
540  _resolveDir(base, navKey, forward) {
541    if(/-(left|right|up|down)$/.test(base)) return base;
542    if(base === 'flip') {
543      return (navKey === 'up' || navKey === 'down') ? 'flip-x' : 'flip-y';
544    }
545    if(base === 'slide' || base === 'cube' || base === 'push') {
546      if(navKey === 'up') return `${base}-down`;
547      if(navKey === 'down') return `${base}-up`;
548      return `${base}-${forward ? 'left' : 'right}'`;
549    }
550    if(base === 'cover' || base === 'uncover') {
551      if(navKey === 'up') return `${base}-up`;
552      if(navKey === 'down') return `${base}-down`;
553      return `${base}-${forward ? 'left' : 'right}'`;
554    }
555  }
556  return base;
557}
558 /**
559  * Attaches gesture control
560  */
561 _attachGestureControl() {
562  let startX = 0, startY = 0, tracking = false;
563  const PX_THRESHOLD = 40;
564  const slidesArea = this.slides_cont;
565
566  slidesArea.addEventListener('pointerdown', e => {

```

```

568   if(e.target.closest('.js-plotly-plot') ||
569     e.target.closest('input.ui') ||
570     e.target.closest('scene-3d-json')) return;
571   if(e.pointerType !== 'mouse' || e.buttons === 1) {
572     tracking = true;
573     startX = e.clientX;
574     startY = e.clientY;
575   }
576 }, { passive: true });

577
578 slidesArea.addEventListener('pointerup', e => {
579   if(e.target.closest('.js-plotly-plot') ||
580     e.target.closest('input.ui') ||
581     e.target.closest('scene-3d-json')) return;
582   if(!tracking) return;
583   tracking = false;

584   const dx = e.clientX - startX;
585   const dy = e.clientY - startY;

586   if(Math.abs(dx) > Math.abs(dy) && Math.abs(dx) > PX_THRESHOLD) {
587     if(dx < 0) {
588       this._lastNavKey = 'right';
589       this.nextSlide();
590     } else {
591       this._lastNavKey = 'left';
592       this.prevSlide();
593     }
594   } else if(Math.abs(dy) > Math.abs(dx) && Math.abs(dy) > PX_THRESHOLD) {
595     if(dy < 0) {
596       this._lastNavKey = 'up';
597       this.prevSlide();
598     } else {
599       this._lastNavKey = 'down';
600       this.nextSlide();
601     }
602   }
603 }, { passive: true });
604 slidesArea.addEventListener('pointercancel', () => { tracking = false; });
605 }
606 }
607 }
608 }

609
610 var presentation = new PRDC_JSLAB_PRESENTATION();

611 /**
612 * Loads file from buffer
613 * @param {string} buf_url      URL to buffered file.
614 */
615 async function loadFileBuf(buf_url) {
616   var name = encodeURIComponent(buf_url);
617   if(window.file_buffers[name]) return Promise.resolve(window.file_buffers[
618     name]);
619   return new Promise((resolve, reject) => {
620     const s = document.createElement('script');
621     s.src = buf_url + '.buf.js';

```

```

622     s.onload = () => {
623         resolve(window.file_buffers[name]);
624     }
625     document.head.appendChild(s);
626 });
627 }
628
629 /**
630 * Class for ImagePDF HTML element
631 */
632 class ImagePDF extends HTMLElement {
633
634     static observedAttributes = [ 'src', 'page', 'width', 'height' ];
635
636 /**
637 * Initializes an instance of the ImagePDF class.
638 */
639 constructor() {
640     super();
641     this._canvas = document.createElement('canvas');
642     this._context = this._canvas.getContext('2d');
643     this.appendChild(this._canvas);
644 }
645
646 /**
647 * Callback called when element is added to page
648 */
649 connectedCallback() {
650     if(!isLazy) this._render();
651     this.isConnected = true;
652 }
653
654 /**
655 * Callback called when element's attribute is changed
656 */
657 attributeChangedCallback() {
658     if(!this.isConnected) return;
659     if(!isLazy || this.closest('slide')?.classList.contains('active')) {
660         this._render();
661     }
662 }
663
664 /**
665 * Renders element
666 */
667 async _render() {
668     var srcAttr = this.getAttribute('src');
669     if(!srcAttr) return;
670
671     let loadingTask;
672     try {
673         if(this.src != srcAttr) {
674             if(presentation._standalone){
675                 var buf = await loadFileBuf(srcAttr);
676                 loadingTask = pdfjsLib.getDocument({ data: buf });

```

```

677     } else {
678       loadingTask = pdfjsLib.getDocument(src_attr);
679     }
680     this.src = src_attr;
681     this.pdf = await loadingTask.promise;;
682   }
683
684   var page_n = parseInt(this.getAttribute('page') || '1', 10) || 1;
685   var page = await this.pdf.getPage(page_n);
686
687   var vp0 = page.getViewport({ scale: 1 });
688   var wanted_w = presentation.toPixels(this.getAttribute('width'), 'width'
689     ) || vp0.width;
690   var wanted_h = presentation.toPixels(this.getAttribute('height'), 'height'
691     ) || vp0.height;
692
693   var scale = wanted_w ? wanted_w / vp0.width : wanted_h ? wanted_h / vp0.
694     height : 1;
695   var vps = page.getViewport({ scale });
696
697   this._canvas.style.display = 'block';
698   this._canvas.width = vps.width;
699   this._canvas.height = vps.height;
700
701   await page.render({ canvasContext: this._context, viewport: vps }).promise;
702   this._finished_loading = true;
703 } catch(err){
704   console.error('img-pdf:', err);
705 }
706
707 customElements.define('img-pdf', ImagePDF);
708
709 /**
710 * Class for PlotJSON HTML element
711 */
712 class PlotJSON extends HTMLElement {
713
714   static observedAttributes = ['src', 'width', 'height'];
715
716   /**
717    * Initializes an instance of the PlotJSON class.
718    */
719   constructor() {
720     super();
721     this._cont = this.appendChild(document.createElement('div'));
722   }
723
724   /**
725    * Callback called when element is added to page
726    */
727   connectedCallback() {
728     if(!is_lazy) this._render();

```

```

728     this.is_connected = true;
729   }
730
731   /**
732    * Callback called when element's attribute is changed
733    */
734   attributeChangedCallback() {
735     if(!this.is_connected) return;
736     if(!is_lazy || this.closest('slide')?.classList.contains('active')) {
737       this._render();
738     }
739   }
740
741   /**
742    * Renders element
743    */
744   async _render() {
745     var src_attr = this.getAttribute('src');
746     if(!src_attr) return;
747
748     try {
749       if(this.src != src_attr) {
750         if(presentation._standalone){
751           const buf = await loadFileBuf(src_attr);
752           this.data = JSON.parse(buf);
753         } else {
754           var resp = await fetch(src_attr, { cache: 'no-store' });
755           this.data = await resp.json();
756         }
757         this.src = src_attr;
758       }
759
760       var w = presentation.toPixels(this.getAttribute('width'), 'width') || 0;
761       var h = presentation.toPixels(this.getAttribute('height'), 'height') ||
762         0;
763       if(w) this.data.layout.width = w;
764       if(h) this.data.layout.height = h;
765
766       await Plotly.newPlot(this._cont, this.data);
767       this._finished_loading = true;
768     } catch(err){
769       console.error('plot-json:', err);
770     }
771   }
772
773 customElements.define('plot-json', PlotJSON);
774
775 /**
776  * Class for Scene3dJSON HTML element
777 */
778 class Scene3dJSON extends HTMLElement {
779
780   static observedAttributes = ['src', 'width', 'height'];
781

```

```

782  /**
783   * Initializes an instance of the Scene3dJSON class.
784   */
785  constructor() {
786    super();
787    this._canvas = document.createElement('canvas');
788    this.appendChild(this._canvas);
789  }
790
791 /**
792  * Callback called when element is added to page
793 */
794 connectedCallback() {
795   if(!is_lazy) this._render();
796   this.is_connected = true;
797 }
798
799 /**
800  * Callback called when element's attribute is changed
801 */
802 attributeChangedCallback() {
803   if(!this.is_connected) return;
804   if(!is_lazy || this.closest('slide')?.classList.contains('active')) {
805     this._render();
806   }
807 }
808
809 /**
810  * Renders element
811 */
812 async _render() {
813   var src_attr = this.getAttribute('src');
814   if(!src_attr) return;
815
816   try {
817     if(this.src !== src_attr) {
818       if(presentation._standalone) {
819         var buf = await loadFileBuf(src_attr);
820         this.data = JSON.parse(buf);
821       } else {
822         var resp = await fetch(src_attr, { cache: 'no-store' });
823         this.data = await resp.json();
824       }
825       this.src = src_attr;
826     }
827
828     var w = presentation.toPixels(this.getAttribute('width'), 'width') ||
829           640;
830     var h = presentation.toPixels(this.getAttribute('height'), 'height') ||
831           480;
832
833     await presentation.waitForGlobal('THREE');
834
835     var loader = new window.THREE.ObjectLoader();
836     this.scene = loader.parse(this.data);

```

```

835   this.camera = new window.THREE.PerspectiveCamera(45, w / h, 0.1, 1000);
836   this.renderer = new window.THREE.WebGLRenderer({ canvas: this._canvas,
837     alpha: true, antialias: true });
838   this.renderer.setSize(w, h);
839   this.renderer.setAnimationLoop(() => {
840     this.renderer.render(this.scene, this.camera)
841   });
842
843   var script = this.querySelector('script[type="x-scene-setup"]');
844   if(script) {
845     var AsyncFunction = Object.getPrototypeOf(
846       async function () {} ).constructor;
847     var fn = new AsyncFunction(
848       'presentation',
849       script.textContent
850     ).bind(this);
851     await fn(presentation);
852   }
853
854   this.renderer.render(this.scene, this.camera);
855   this._finished_loading = true;
856 } catch(err) {
857   console.error('scene-3d-json:', err);
858 }
859 }
860
861 customElements.define('scene-3d-json', Scene3dJSON);

```

Listing 135 - presentation.js

```

1  /**
2   * @file JSLAB library terminal script
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB ui.
10  */
11 class PRDC_JSLAB_TERMINAL {
12
13 /**
14  * Initializes an instance of the PRDC_JSLAB_TERMINAL class.
15  */
16 constructor() {
17   var obj = this;
18
19   this.messages = document.getElementById('messages-container');
20   this.message_input = document.getElementById('message-input');
21   this.autoscroll = true;
22   this.show_timestamp = true;
23   this.write_timestamps = true;
24   this.log = [];
25   this.log_dialog = document.getElementById('log-dialog');

```

```

26   this.settings_dialog = document.getElementById('settings-dialog');
27   this.dialogs = [this.log_dialog, this.settings_dialog]
28   this.N_messages = 1;
29   this.N_messages_max = Infinity;
30   this.last_class = null;
31   this.last_tic = 0;
32
33   for(var dialog of this.dialogs) {
34     dialog.addEventListener('input', function(e) {
35       if(e.key == 'Escape') {
36         // ESC
37         obj.closeDialog(dialog);
38         e.stopPropagation();
39         e.preventDefault();
40       }
41     });
42   }
43
44   // Terminal settings button click
45   document.getElementById('settings').addEventListener('click', () => {
46     this.openDialog(this.settings_dialog);
47   });
48
49   // Terminal timestamp button click
50   document.getElementById('timestamp').addEventListener('click', () => {
51     this.setTimestamp(!this.show_timestamp);
52   });
53   this.setTimestamp(this.show_timestamp);
54
55   // Terminal auto scroll button click
56   document.getElementById('autoscroll').addEventListener('click', () => {
57     this.setAutoscroll(!this.autoscroll);
58   });
59   this.setAutoscroll(this.autoscroll);
60
61   // Terminal clear button click
62   document.getElementById('clear').addEventListener('click', () => {
63     this.clear();
64   });
65
66   // Terminal save log button click
67   document.getElementById('log').addEventListener('click', () => {
68     this.openDialog(this.log_dialog);
69   });
70
71   // Terminal scroll to bottom button click
72   document.getElementById('to-bottom').addEventListener('click', () => {
73     this.scrollToBottom();
74     this.message_input.focus();
75     const len = this.message_input.value.length;
76     this.message_input.setSelectionRange(len, len);
77   });
78
79   document.querySelector('#settings-dialog .options-close').addEventListener(
80     'click', () => {

```



```
80     this.closeDialog(this.settings_dialog);
81 });
82
83 document.getElementById('N-messages-max').value = this.N_messages_max;
84 document.querySelector('#settings-dialog .change-settings').
85   addEventListener('click', () => {
86   obj.closeDialog(obj.settings_dialog);
87   obj.setNMessagesMax(Number(document.getElementById('N-messages-max').
88     value));
89 })
90 this.setNMessagesMax(this.N_messages_max);
91
92 document.querySelector('#log-dialog .options-close').addEventListener(
93   'click', () => {
94   this.closeDialog(this.log_dialog);
95 })
96 document.getElementById('write-timestamps').addEventListener('click',
97   function() {
98   obj.setWriteTimestamps(this.checked);
99 })
100 this.setWriteTimestamps(this.write_timestamps);
101
102 /**
103 * Sets options.
104 */
105 setOptions(opts) {
106   if(opts.hasOwnProperty('show_timestamp')) {
107     this.setTimestamp(opts.show_timestamp);
108   }
109   if(opts.hasOwnProperty('show_timestamp')) {
110     this.setAutoscroll(opts.autoscroll);
111   }
112   if(opts.hasOwnProperty('N_messages_max')) {
113     this.setNMessagesMax(opts.N_messages_max);
114   }
115   if(opts.hasOwnProperty('write_timestamps')) {
116     this.setWriteTimestamps(opts.write_timestamps);
117   }
118 }
119
120 /**
121 * Auto resizes input
122 */
123 autoResizeInput() {
124   this.message_input.style.height = 'auto';
125   this.message_input.style.height = this.message_input.scrollHeight + 'px';
126 }
```

```

131     }
132
133     /**
134      * Clears all messages from the command window.
135      */
136     clear() {
137         this.N_messages = 1;
138         this.messages.innerHTML = '';
139     }
140
141     /**
142      * Scrolls the command window to the bottom.
143      */
144     scrollToBottom() {
145         this.messages.scrollTop = this.messages.scrollHeight;
146     }
147
148     /**
149      * Returns a formatted timestamp string.
150      */
151     getTimestamp() {
152         const date = new Date();
153         const pad = (num, size) => ('000' + num).slice(size * -1);
154         const time = parseFloat(date.getTime() / 1000).toFixed(3);
155         const hours = date.getHours();
156         const minutes = Math.floor(time / 60) % 60;
157         const seconds = Math.floor(time - minutes * 60);
158         const milliseconds = time.slice(-3);
159         return pad(hours, 2) + ':' + pad(minutes, 2) + ':' +
160                 pad(seconds, 2) + '.' + pad(milliseconds, 3);
161     }
162
163     /**
164      * Toggles the visibility of timestamps in the command window.
165      */
166     setTimestamp(show_timestamp) {
167         this.show_timestamp = show_timestamp;
168         const timestampButton = document.getElementById('timestamp');
169         if(this.show_timestamp) {
170             if(this.messages.classList.contains('no-timestamp')) {
171                 this.messages.classList.remove('no-timestamp');
172                 timestampButton.classList.add('active');
173                 timestampButton.setAttribute('title-str', 41);
174             }
175         } else {
176             if(!this.messages.classList.contains('no-timestamp')) {
177                 this.messages.classList.add('no-timestamp');
178                 timestampButton.classList.remove('active');
179                 timestampButton.setAttribute('title-str', 166);
180             }
181         }
182     }
183
184     /**
185      * Toggles the autoscroll feature.

```

```
186  */
187 setAutoscroll(autoscroll) {
188     this.autoscroll = autoscroll;
189     const autoscrollButton = document.getElementById('autoscroll');
190     if(this.autoscroll) {
191         if(!autoscrollButton.classList.contains('active')) {
192             autoscrollButton.classList.add('active');
193             autoscrollButton.setAttribute('title-str', 42);
194         }
195     } else {
196         if(autoscrollButton.classList.contains('active')) {
197             autoscrollButton.classList.remove('active');
198             autoscrollButton.setAttribute('title-str', 167);
199         }
200     }
201 }
202
203 /**
204 * Sets the maximum number of messages to display.
205 */
206 setNMessagesMax(N_messages_max) {
207     if(!N_messages_max) return;
208     this.N_messages_max = N_messages_max;
209     const nMessagesMaxInput = document.getElementById('N-messages-max');
210     if(this.N_messages_max < 5) {
211         this.N_messages_max = 5;
212     }
213     if(this.N_messages >= this.N_messages_max) {
214         while(this.N_messages > this.N_messages_max) {
215             this.messages.firstChild.remove();
216             this.N_messages -= 1;
217         }
218     }
219     if(nMessagesMaxInput.value != this.N_messages_max) {
220         nMessagesMaxInput.value = this.N_messages_max;
221     }
222 }
223
224 /**
225 * Toggles whether timestamps are written to the log file.
226 */
227 setWriteTimestamps(write_timestamps) {
228     this.write_timestamps = write_timestamps;
229     const writeTimestampsInput = document.getElementById('write-timestamps');
230     if(this.write_timestamps) {
231         if(!writeTimestampsInput.checked) {
232             writeTimestampsInput.checked = true;
233         }
234     } else {
235         if(writeTimestampsInput.checked) {
236             writeTimestampsInput.checked = false;
237         }
238     }
239 }
```

```

241 /**
242 * Adds a message to the terminal output.
243 * @param {string} msg_class - The CSS class for the message.
244 * @param {string} data - The HTML content of the message.
245 * @param {string} [raw] - The raw text data (defaults to data if undefined)
246 *
247 * @param {boolean} [merge_messages=true] - Whether to merge with the
248 * previous message.
249 * @returns {HTMLElement} The created message element.
250 */
251 addMessage(msg_class, data, raw, merge_messages = true) {
252   if(typeof raw === 'undefined') {
253     raw = data;
254   }
255   const t = performance.now();
256   let el;
257
258   if(msg_class !== this.last_class) {
259     this.last_class = msg_class;
260     this.last_tic = t;
261     const ts = this.getTimestamp();
262     if(this.N_messages < this.N_messages_max) {
263       this.N_messages++;
264     } else if(this.messages.firstChild) {
265       this.messages.firstChild.remove();
266     }
267
268     el = document.createElement('div');
269     el.className = msg_class;
270     el.innerHTML = '<span class="timestamp">$\{ts\}</span>$\{data\}';
271     this.messages.appendChild(el);
272     this.log.push({ 'class': msg_class, 'timestamp': ts, 'data': raw });
273
274   } else {
275     if(!merge_messages || t - this.last_tic > 5) {
276       this.last_tic = t;
277       const ts = this.getTimestamp();
278       if(this.N_messages <= this.N_messages_max) {
279         this.N_messages++;
280       } else if(this.messages.firstChild) {
281         this.messages.firstChild.remove();
282       }
283
284       el = document.createElement('div');
285       el.className = msg_class;
286       el.innerHTML = '<span class="timestamp">$\{ts\}</span>$\{data\}';
287       this.messages.appendChild(el);
288       this.log.push({ 'class': msg_class, 'timestamp': ts, 'data': raw });
289
290     } else {
291       // Merge data with the last message element
292       el = this.messages.querySelector('div:last-child');
293       if(el) {
294         el.innerHTML += data;
295       }
296       this.log[this.log.length - 1].data += raw;
297     }
298   }
299 }

```

```
294     }
295 }
296
297     if(this.autoscroll) {
298         this.scrollToBottom();
299     }
300     return el;
301 }
302
303 /**
304 * Gets the current log of the command window to a file.
305 */
306 getLog() {
307     var obj = this;
308     var data = '';
309     this.log.forEach(function(x) {
310         if(x.class) {
311             data += x.class + ' ';
312         }
313         if(obj.write_timestamps) {
314             data += '[' + x.timestamp + '] ';
315         }
316         data += x.data + '\r\n';
317     });
318     return data;
319 }
320
321 /**
322 * Opens a specified dialog (e.g., settings or log).
323 * For simplicity, fade animations are replaced by immediate show/hide.
324 * @param {HTMLElement} dialog - The dialog element to show.
325 */
326 openDialog(dialog) {
327     if(window.getComputedStyle(dialog).display === 'none') {
328         document.querySelectorAll('.terminal-dialog').forEach(el => {
329             el.style.display = 'none';
330         });
331         dialog.style.display = 'block';
332         dialog.focus();
333     }
334 }
335
336 /**
337 * Closes a specified dialog.
338 * @param {HTMLElement} dialog - The dialog element to hide.
339 */
340 closeDialog(dialog) {
341     dialog.style.display = 'none';
342     this.message_input.focus();
343     const len = this.message_input.value.length;
344     this.message_input.setSelectionRange(len, len);
345 }
346 }
```

348 **var** terminal = **new** PRDC_JSLAB_TERMINAL();

Listing 136 - terminal.js

```

1  /**
2   * @file JSLAB library ui script
3   * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4   * PR-DC, Republic of Serbia
5   * info@pr-dc.com
6   */
7
8  /**
9   * Class for JSLAB ui.
10  */
11 class PRDC_JSLAB_UI {
12
13 /**
14  * Initializes an instance of the PRDC_JSLAB_UI class.
15  */
16 constructor() {
17     // Tabs
18     var container = document.querySelector('.ui.tabs-cont');
19     if(container) {
20         var headers = [...container.querySelectorAll('.ui.tab-name')];
21         var panels = [...container.querySelectorAll('.ui.tab')];
22
23         function showTab(idx) {
24             headers.forEach((h, i) => h.classList.toggle('active', i === idx));
25             panels.forEach((p, i) => p.style.display = i === idx ? 'block' : 'none'
26                 );
27         };
28         var start_idx = headers.findIndex(h => h.classList.contains('active'));
29         if(start_idx === -1) {
30             start_idx = 0;
31         }
32         showTab(start_idx);
33         headers.forEach((h, i) => h.addEventListener('click', () => showTab(i)))
34             ;
35     }
36 }
37
38 var ui = new PRDC_JSLAB_UI();

```

Listing 137 - ui.js