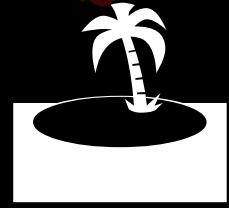


project-tech

Env & Linting & Build

lab 3/8



Window 2 < > https://forms.office.com/Pages/DesignPage.aspx?origin=Neop... Back Computer Mobile

🚀 Tussentijdse Feedback - Blok Tech (Blok-4)

👉 We zijn nu een paar weekjes aan de slag met Blok-Tech. We willen heel graag weten wat je mening over de vakken tot nu toe is. Je helpt ons er heel erg mee als je even de tijd neemt om dit in te vullen! We nemen de resultaten mee in de voortgangsgesprekken aankomende vragen.

Elke sectie heeft een comments tabje, gebruik het voor extra opmerkingen als je iets kwijt wilt. 'Ik vind het leuk' daar hebben we niet zoveel aan ;-). 'Ik vind het stom' ook niet. Laat dan ook ff weten wat er leuk of stom is.

Thanks!

Robert, Ivo, Janno, Sonja & Danny

Hi, Danny. When you submit this form, the owner will see your name and email address.

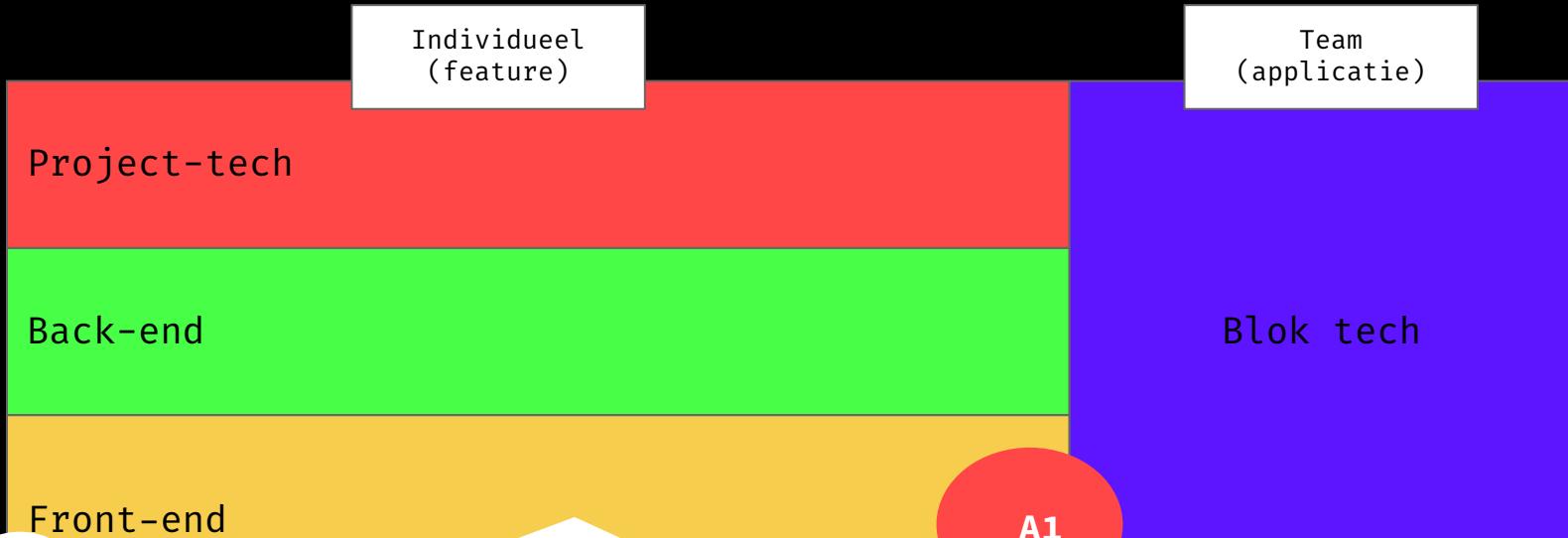
* Required

1. Hoi! Wie ben jij?

Geef je feedback!

Activities of all types shown		<input type="button" value="▼"/>	<input type="button" value="<"/>	<input type="button" value="Today"/>	<input type="button" value=">"/>	<input type="button" value="+ Add timetable"/>	<input type="button" value="Download"/>
Thu 12 May	Fri 13 May						
		08:30 - 11:50 Project: Tech TTH 02B08 Design Lab Group(s): FDMCI_TECH 4 supervision / consultancy	09:20 - 12:40 Project: Tech TTH 01B25 Group(s): FDMCI_TECH 5 supervision / consultancy	2021/2022 <ul style="list-style-type: none"> ▶ <input type="checkbox"/> Benvenuti, L. ▶ <input type="checkbox"/> Buis, P. ▶ <input type="checkbox"/> Dokkum, C.R. van ▶ <input type="checkbox"/> Faber, J. ▶ <input type="checkbox"/> Frank, N.M. ▶ <input type="checkbox"/> Gemert, V. van ▶ <input type="checkbox"/> Giannuzzi, W. ▶ <input type="checkbox"/> Kapritsias, J. ▶ <input type="checkbox"/> Kapritsias, J. ▶ <input type="checkbox"/> Nijhuis, I. ▶ <input type="checkbox"/> Nijhuis, I. ▶ <input type="checkbox"/> Rouwhorst, S.E. ▶ <input type="checkbox"/> Schravendijk, A. van ▶ <input type="checkbox"/> Spier, R. ▶ <input type="checkbox"/> Thung, D.H. ▶ <input type="checkbox"/> Vries, D.A. de ▶ <input type="checkbox"/> Wijnmaalen, J.B. ▶ <input type="checkbox"/> Algemeen - FDMCI/CMD/CMD/algemeen 			

Voortgangsgesprek



*Week-3
(halfway)*



Show what
you did

Stand-up!

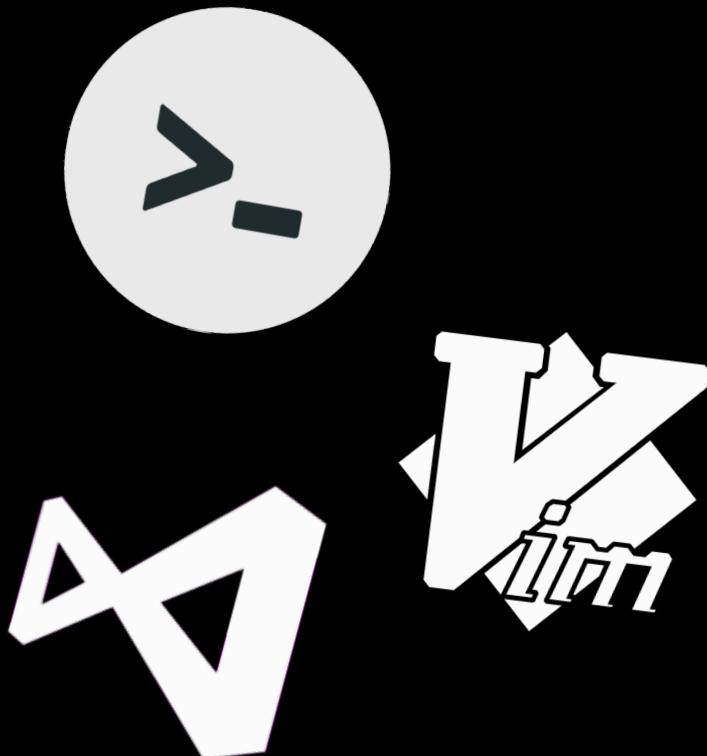
today

I. Standup

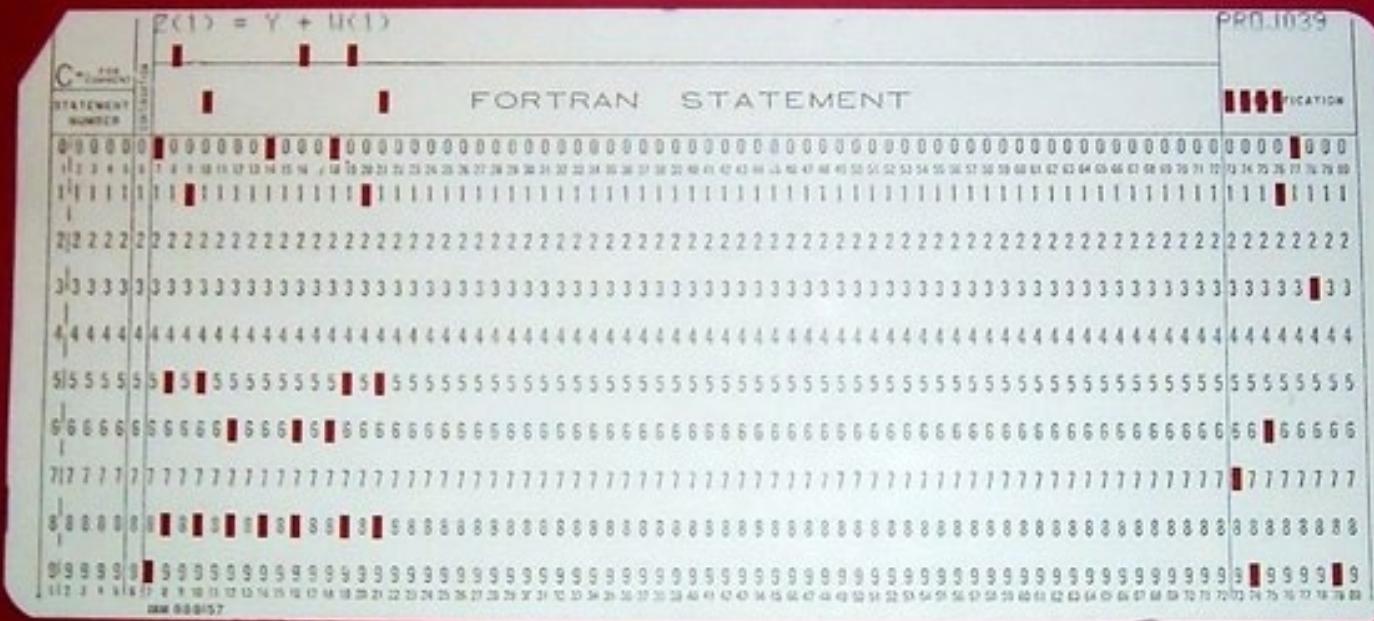
II. Local Dev environment

III. Linters & Formatters

IV. Build Tools



Dev Tooling



Steve Baker

<https://qph.fs.quoracdn.net/main-qimg-4da0567cc18f277fd5bf46fd9667c963-c>



Bucketlist

Code-Editors

Code Editors

- ❖ Code is “just” text
- ❖ Computers don’t care what your code looks like
- ❖ Your code-editor is the most important tool in your arsenal

```
files.js
async function getCount(context) {
  const count = await apiV8.get("/files", {
    params: {
      limit: 1,
      meta: "total_count",
    },
  });
  context.fileCount = count.data.meta.total_count;
  context.fileMap = {};
}

async function uploadFiles(context) {
  const pages = Math.ceil(context.fileCount / 100);

  const tasks = [];
  for (let i = 0; i < pages; i++) {
    tasks.push({
      title: `Uploading files ${i * 100 + 1}-${(i + 1) * 100}`,
      task: uploadBatch(i),
    });
  }
  return new Listr(tasks);
}

function uploadBatch(page) {
  return async (context, task) => {
    const records = await apiV8.get("/files", {
      params: {
        offset: page * 100,
        limit: 100,
      },
    });

    for (const fileRecord of records.data.data) {
      task.output = fileRecord.filename_download;

      const savedFile = await apiV9.post("/files/import", {
        url: fileRecord.data.full_url,
        data: {
          filename_download: fileRecord.filename_download,
          title: fileRecord.title,
          description: fileRecord.description,
        },
      });

      context.fileMap[fileRecord.id] = savedFile.data.data.id;
    }
  };
}
```

```
type-map.js
export const typeMap = {
  alias: 'alias',
  array: 'json',
  binary: 'binary',
  boolean: 'boolean',
  datetime: 'dateTime',
  date: 'date',
  time: 'time',
  file: 'uuid',
  group: 'integer',
  hash: 'hash',
  integer: 'integer',
  decimal: 'decimal',
  json: 'json',
  lang: 'string',
  m2o: 'integer',
  o2m: 'alias',
  slug: 'string',
  sort: 'integer',
  status: 'string',
  string: 'string',
  translation: 'integer',
  uuid: 'uuid',
  datetime_created: 'timestamp',
  datetime_updated: 'timestamp',
  user_created: 'uuid',
  user_updated: 'uuid',
  user: 'uuid',
  owner: 'uuid',
}
```

The image shows two side-by-side code editors on a Mac OS X desktop. Both editors are running in the Nova code editor application.

Left Editor (files.js):

```
async function getCount(context) {
  const count = await apiV8.get("/files", {
    params: {
      limit: 1,
      meta: "total_count",
    },
  });
  context.fileCount = count.data.meta.total_count;
  context.fileMap = {};
}

async function uploadFiles(context) {
  const pages = Math.ceil(context.fileCount / 100);

  const tasks = [];
  for (let i = 0; i < pages; i++) {
    tasks.push({
      title: `Uploading files ${i * 100 + 1}-${(i + 1) * 100}`,
      task: uploadBatch(i),
    });
  }
  return new Listr(tasks);
}

function uploadBatch(page) {
  return async (context, task) => {
    const records = await apiV8.get("/files", {
      params: {
        offset: page * 100,
        limit: 100,
      },
    });
    for (const fileRecord of records.data.data) {
      task.output = fileRecord.filename_download;

      const savedFile = await apiV9.post("/files/import", {
        url: fileRecord.data.full_url,
        data: {
          filename_download: fileRecord.filename_download,
          title: fileRecord.title,
          description: fileRecord.description,
        },
      });
      context.fileMap[fileRecord.id] = savedFile.data.data.id;
    }
  };
}
```

Right Editor (migration-tool/files.js):

```
async function getCount(context) {
  const count = await apiV8.get("/files", {
    params: {
      limit: 1,
      meta: "total_count",
    },
  });
  context.fileCount = count.data.meta.total_count;
  context.fileMap = {};
}

async function uploadFiles(context) {
  const pages = Math.ceil(context.fileCount / 100);

  const tasks = [];
  for (let i = 0; i < pages; i++) {
    tasks.push({
      title: `Uploading files ${i * 100 + 1}-${(i + 1) * 100}`,
      task: uploadBatch(i),
    });
  }
  return new Listr(tasks);
}

function uploadBatch(page) {
  return async (context, task) => {
    const records = await apiV8.get("/files", {
      params: {
        offset: page * 100,
        limit: 100,
      },
    });
    for (const fileRecord of records.data.data) {
      task.output = fileRecord.filename_download;

      const savedFile = await apiV9.post("/files/import", {
        url: fileRecord.data.full_url,
        data: {
          filename_download: fileRecord.filename_download,
          title: fileRecord.title,
          description: fileRecord.description,
        },
      });
      context.fileMap[fileRecord.id] = savedFile.data.data.id;
    }
  };
}
```

A blue arrow points from the left editor to the right editor, indicating a comparison or migration process between the two versions of the code.

Code Editors

- ❖ Specialized for writing software
- ❖ Make writing and reading source code easier
- ❖ Syntax Highlighting, Find and Replace, Extensions
- ❖ Personal Preference!



Visual Studio Code



Brackets



WebStorm



Sublime Text



ATOM



Visual Studio



GNU Emacs



Xcode 12

A screenshot of the Visual Studio Code interface. The title bar says "permissions.ts — directus". The left sidebar shows a tree view of files in the "src" directory, including "activity.ts", "assets.ts", "auth.ts", "collections.ts", "extensions.ts", "fields.ts", "files.ts", "folders.ts", "graphql.ts", "items.ts", "not-found.ts", "presets.ts", "relations.ts", "revisions.ts", "roles.ts", "server.ts", "settings.ts", "users.ts", "utils.ts", "webhooks.ts", "database", "exceptions", "mail", "middleware", "services", "types", "util", "app.ts", "cache.ts", "constants.ts", "emitter.ts", "env.ts", "extensions.ts", "grant.ts", and "logger.ts". The main editor area displays the "permissions.ts" file. The code is a TypeScript file for a REST API endpoint. It imports express, asyncHandler, PermissionsService, MetaService, lodash, and various exception classes. It defines a router, sets up middleware, and handles POST requests to "/permissions". It uses a service to read by key or query, and a service to read by primary key. It also handles GET requests to "/permissions". The code includes several overloads for the service methods. The status bar at the bottom shows "Ln 1, Col 1 Tab Size: 3 UTF-8 LF TypeScript 4.1.2 Prettier".

```
ts permissions.ts
api > src > controllers > TS permissions.ts > ...
1 import express from 'express';
2 import { asyncHandler } from '../../utils/async-handler';
3 import { PermissionsService, MetaService } from '../services';
4 import { clone } from 'lodash';
5 import { InvalidCredentialsException, ForbiddenException, InvalidPayloadException } from '../exceptions';
6 import useCollection from './middleware/use-collection';
7 import { respond } from './middleware/respond';
8 import { PrimaryKey } from '../types';
9
10 const router = express.Router();
11
12 router.use(useCollection('directus_permissions'));
13
14 router.post(
15   '/',
16   asyncHandler(async (req, res, next) => {
17     const service = new PermissionsService({
18       accountability: req.accountability,
19       schema: req.schema,
20     });
21     const primaryKey = await service[method](keys: PrimaryKey[], query?: Query | undefined, action?: "create" | "read" | "update" | "delete" | "comment" | "explain" | undefined);
22     try {
23       const item = await service.readByKey(primaryKey, req.sanitizedQuery);
24       res.locals.payload = { data: item || null };
25     } catch (error) {
26       if (error instanceof ForbiddenException) {
27         return next();
28       }
29       throw error;
30     }
31     return next();
32   })
33   respond
34 );
35
36 router.get(
37   '/',
38   asyncHandler(async (req, res, next) => {
39     const service = new PermissionsService({
40       accountability: req.accountability,
41       schema: req.schema,
42     });
43     const metaService = new MetaService({
44       accountability: req.accountability,
45       schema: req.schema,
46     });
47   })
48 );
```

Visual Studio Code

```
graphql.ts                                     permissions.ts

1 import express from 'express';
2 import asyncHandler from '../utils/async-handler';
3 import { PermissionsService, MetaService } from '../services';
4 import { clone } from 'lodash';
5 import { InvalidCredentialsException, ForbiddenException, InvalidPayloadException } from '../exceptions';
6 import useCollection from '../middleware/use-collection';
7 import { respond } from '../middleware/respond';
8 import { PrimaryKey } from '../types';
9
10 const router = express.Router();
11
12 router.use(useCollection('directus_permissions'));
13
14 router.post(
15   '/',
16   asyncHandler(async [req, res, next] => {
17     const service = new PermissionsService({
18       accountability: req.accountability,
19       schema: req.schema,
20     });
21     const primaryKey = await service.createRecord<string, any>.readByKey(keys: PrimaryKey[], query?: Query | undefined, action: "create" | "read" | "update" | "delete" | "comment" | "explain" | undefined);
22     try {
23       const item = await service.readByKey(primaryKey, req.sanitizedQuery);
24       res.locals.payload = { data: item || null };
25     } catch (error) {
26       if (error instanceof ForbiddenException) {
27         return next();
28       }
29     }
30     throw error;
31   })
32   .then(next)
33 );
34
35 router.get(
36   '/',
37   asyncHandler(async [req, res, next] => {
38     const service = new PermissionsService({
39       accountability: req.accountability,
40       schema: req.schema,
41     });
42     const metaService = new MetaService({
43       accountability: req.accountability,
44       schema: req.schema,
45     });
46
47     const item = await service.readByQuery(req.sanitizedQuery);
48     const meta = await metaService.getMetaForQuery('directus_permissions', req.sanitizedQuery);
49   })
50   .then(next)
51 );
```

Nova

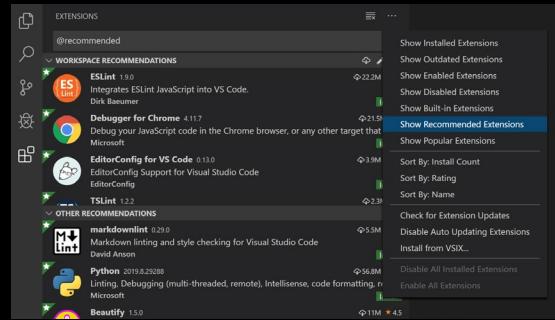
Code Editor Customisation



1) Look and Feel

```
{  
  "files.autoSave": "afterDelay",  
  "diffEditor.renderSideBySide": false,  
  "breadcrumbs.enabled": false,  
  "editor.minimap.enabled": false,  
  "typescript": [  
    {  
      "diffEditor.ignoreTrimWhitespace": true,  
      "diffEditor.maxComputationTime": 1000,  
      "diffEditor.renderIndicators": true,  
      "diffEditor.renderSideBySide": true,  
      "editor.acceptSuggestionOnCommitCharacter": true,  
      "editor.acceptSuggestionOnEnter": true,  
      "editor.accessibilityPageSize": 100,  
      "editor.accessibilitySupport": true,  
      "editor.autoClosingBrackets": true,  
      "editor.autoClosingOvertype": true,  
      "editor.autoClosingQuotes": true,  
      "editor.autoIndent": true  
    }  
  ]  
}
```

2) Settings



3) Extensions

Assignment ($\pm 30\text{m}$)

Research the functionalities of your IDE development and document the research in your wiki. Look into the following subjects:

- Themes (f.e. [color themes](#), [fonts](#))
- Settings (f.e. [settings.json](#))
- Extensions (f.e. [extensions marketplace](#))



Linters & Formatters

linters

?

A **linter** catches inconsistencies throughout your project and reduces your *chances of making logical errors*.

Syntax: Linters check the source code for programmatic mistakes (bugs, errors)

formatter

?

A **formatter** can help maintain a *consistent coding style*. It's basically a ‘style guide’ for how you write code.

Stylistic: beautifies (and formats) your code to make it more readable.

```
vagrant@precise32:~$ eslint uploader.js
```

```
uploader.js
```

```
1:13 error 'angular' is not defined      no-undef
1:28 error Strings must use doublequote quotes
7:15 error Strings must use doublequote quotes
7:28 error Strings must use doublequote quotes
7:34 error Missing "use strict" statement strict
9:20 error Expected '===' and instead saw '==' eqeqeq
10:17 error Expected '===' and instead saw '==' eqeqeq
14:20 error Strings must use doublequote quotes
27:16 error 'FormData' is not defined      no-undef
34:17 error 'XMLHttpRequest' is not defined no-undef
35:11 error Trailing spaces not allowed    no-trailing-spaces
36:12 error Strings must use doublequote quotes
45:1  error Unexpected blank line at end of file eol-last
```

```
✖ 13 problems
```

```
vagrant@precise32:~$ █
```

Syntax (eslint)

```
1 function HelloWorld({greeting = "Hello", greeted = "World", silent = false, onMouseOver,}) {
2
3   if(!greeting){return null};
4
5   // TODO: Don't use random in render
6   let num = Math.floor(Math.random() * 1e7).toString().replace(/\.\d+$/g, "");
7
8   return <div className="HelloWorld" title={"You are visitor number ${ num }"} onMouseOver={onMouseOver}>
9
10  <strong>{ greeting.slice(0, 1).toUpperCase() + greeting.slice(1).toLowerCase() }</strong>
11  {greeting.endsWith(',') ? " " : <span style={{color: 'grey'}}>,</span>}
12  <br>
13  { greeted }
14  ,
15  { silent }
16  ? "."
17  : "|"
18
19  .function HelloWorld({
20   greeting = "Hello",
21   greeted = "World",
22   silent = false,
23   onMouseOver,
24 }) {
25   if (!greeting) {
26     return null;
27   }
28   // TODO: Don't use random in render
29   let num = Math.floor(Math.random() * 1e7)
30   .toString()
31   .replace(/\.\d+$/g, "");
32
33   return (
34     <div
35       className="HelloWorld"
36       title={"You are visitor number ${num}"}
37       onMouseOver={onMouseOver}
38     >
39       <strong>
40         {greeting.slice(0, 1).toUpperCase() + greeting.slice(1).toLowerCase()}
41       </strong>
42       {greeting.endsWith(',') ? (
43         " "
44       ) : (
45         <span style={{ color: "grey" }}>,</span>
46       )}
47       <br>
48       {greeted}
49       {silent ? ":" : "|"}
50     </div>
51   );
52 }
53 }
```

Formatter (prettier)

Configuration File Formats

ESLint supports configuration files in several formats:

- JavaScript - use `.eslintrc.js` and export an object containing your configuration.
- JavaScript (ESM) - use `.eslintrc.cjs` when running ESLint in JavaScript packages that specify `"type": "module"` in their `package.json`. Note that ESLint does not support ESM configuration at this time.
- YAML - use `.eslintrc.yaml` or `.eslintrc.yml` to define the configuration structure.
- JSON - use `.eslintrc.json` to define the configuration structure. ESLint's JSON files also allow JavaScript-style comments.
- `package.json` - create an `eslintConfig` property in your `package.json` file and define your configuration there.

If there are multiple configuration files in the same directory, ESLint will only use one. The priority order is as follows:

1. `.eslintrc.js`
2. `.eslintrc.cjs`
3. `.eslintrc.yaml`
4. `.eslintrc.yml`
5. `.eslintrc.json`
6. `package.json`

Installing an extension or package in your project is usually not enough.

More often than not these linters and formatters *only work if you actually configure them.*

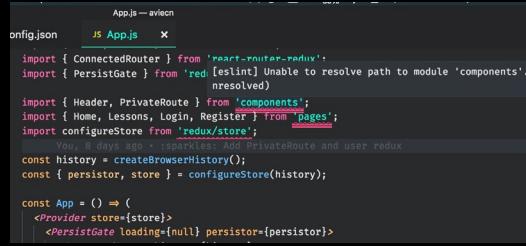
```
vagrant@precise32:~$ eslint uploader.js
```

```
uploader.js
  1:13  error  'angular' is not defined          no-undef
  1:28  error  Strings must use doublequote      quotes
  7:15  error  Strings must use doublequote      quotes
  7:28  error  Strings must use doublequote      quotes
  7:34  error  Missing 'use strict' statement      strict
  9:28  error  Expected '===' and instead saw '=='  eqeqeqq
 10:17  error  Expected '===' and instead saw '=='  eqeqeqq
 14:20  error  Strings must use doublequote      quotes
 27:16  error  'FormData' is not defined          no-undef
 34:17  error  'XMLHttpRequest' is not defined    no-undef
 35:1   error  Trailing spaces not allowed       no-trailing-spaces
 36:12  error  Strings must use doublequote      quotes
 45:1   error  Unexpected blank line at end of file eol-last

  8 13 problems
```

```
vagrant@precise32:~$
```

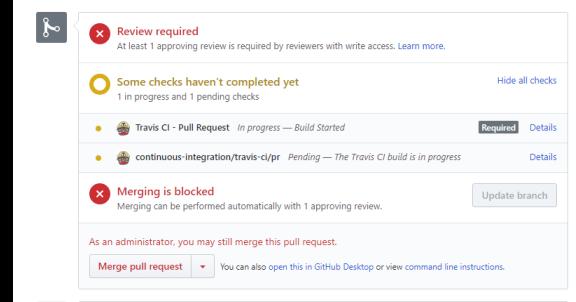
CLI Output (run scripts)



A screenshot of a code editor showing ESLint errors. The file is named 'App.js'. The errors are:

- Line 1:13: 'angular' is not defined (no-undef)
- Line 1:28: Strings must use doublequote (quotes)
- Line 7:15: Strings must use doublequote (quotes)
- Line 7:28: Strings must use doublequote (quotes)
- Line 7:34: Missing 'use strict' statement (strict)
- Line 9:28: Expected '===' and instead saw '==' (eqeqeqq)
- Line 10:17: Expected '===' and instead saw '==' (eqeqeqq)
- Line 14:20: Strings must use doublequote (quotes)
- Line 27:16: 'FormData' is not defined (no-undef)
- Line 34:17: 'XMLHttpRequest' is not defined (no-undef)
- Line 35:1: Trailing spaces not allowed (no-trailing-spaces)
- Line 36:12: Strings must use doublequote (quotes)
- Line 45:1: Unexpected blank line at end of file (eol-last)

IDE
(extensions)



GitHub
(testing)

Linters & formatters

why

Ensuring a good coding style has a couple upsides:

- It helps you avoid bugs
- Help other developers read through your code
- Save time and avoid stress
- Well organized, cleanly written code can be its own documentation!
- Improves team collaboration

ESLint

- ❖ Checks for a wide range of syntax- and stylistic errors
- ❖ Can help you write better code by enforcing best practices
- ❖ Makes sure everybody codes in the same “dialect”

<https://eslint.org>

enforce a maximum depth that blocks can be nested (max-depth)

Many developers consider code difficult to read if blocks are nested beyond a certain depth.

This rule enforces a maximum depth that blocks can be nested to reduce code complexity.

ⓘ Examples of incorrect code for this rule with the default `{"max": 4}` option:

```
/*eslint max-depth: ["error", 4]*/
/*eslint-env es6*/

function foo() {
  for (;;) { // Nested 1 deep
    while (true) { // Nested 2 deep
      if (true) { // Nested 3 deep
        if (true) { // Nested 4 deep
          if (true) { // Nested 5 deep
            }
          }
        }
      }
    }
  }
}
```

ⓘ Examples of correct code for this rule with the default `{"max": 4}` option:

```
/*eslint max-depth: ["error", 4]*/
/*eslint-env es6*/

function foo() {
  for (;;) { // Nested 1 deep
    while (true) { // Nested 2 deep
      if (true) { // Nested 3 deep
        if (true) { // Nested 4 deep
          }
        }
      }
    }
  }
}
```

Prettier

- ❖ Opinionated code formatter
- ❖ Rewrites your code to be consistent and adhering to the rules
- ❖ Very useful, but can cause friction

<https://prettier.io/>

```
1 function HelloWorld({greeting = "hello", greeted = "World", silent = false, onMouseOver,}) {  
2  
3   if(!greeting){return null};  
4  
5   // TODO: Don't use random in render  
6   let num = Math.floor(Math.random() * 1e+7).toString().replace(/\.\d+ig, "")  
7  
8   return <div className='HelloWorld' title={ You are visitor number ${ num } } onMouseOver={onMouseOver}>  
9  
10  <strong>{ greeting.slice( 0, 1 ).toUpperCase() + greeting.slice(1).toLowerCase() }</strong>  
11  {greeting.endsWith(",") ? " " : <span style={{color: '\grey'}}>,"</span> }  
12  <em>  
13  { greeted }  
14  </em>  
15  { (silent)  
16  ? "."  
17  : "!"}  
18  
19 </div>;  
20 }  
21 }
```

Before

```
1 function HelloWorld({  
2   greeting = "hello",  
3   greeted = "World",  
4   silent = false,  
5   onMouseOver,  
6 }) {  
7   if (!greeting) {  
8     return null;  
9   }  
10  // TODO: Don't use random in render  
11  let num = Math.floor(Math.random() * 1e7)  
12  .toString()  
13  .replace(/\.\d+ig, "");  
14  
15  return (  
16    <div  
17      className="HelloWorld"  
18      title={ You are visitor number ${num} }  
19      onMouseOver={onMouseOver}  
20    >  
21      <strong>  
22        {greeting.slice(0, 1).toUpperCase() + greeting.slice(1).toLowerCase()}  
23      </strong>  
24      {greeting.endsWith(",") ? (" ")  
25       : (<span style={{ color: "grey" }}>,"</span> )}  
26      <em>{greeted}</em>  
27      {silent ? (".") : ("!")}  
28    </div>  
29  );  
30 }  
31  
32 };
```

After

.editorconfig

- ❖ Makes sure everybody's editor settings are the same
- ❖ Ensures you never end up with mixed tabs/spaces

<https://editorconfig.org>

```
# EditorConfig is awesome:  
https://EditorConfig.org  
  
# top-most EditorConfig file  
root = true  
  
# Unix-style newlines with a newline ending  
every file  
[*]  
end_of_line = lf  
insert_final_newline = true  
  
# Matches multiple files with brace expansion  
notation  
# Set default charset  
[*.js,py]  
charset = utf-8  
  
# 4 space indentation  
[*.py]  
indent_style = space  
indent_size = 4  
  
# Tab indentation (no size specified)  
[Makefile]  
indent_style = tab  
  
# Indentation override for all JS under lib  
directory  
[lib/**.js]  
indent_style = space  
indent_size = 2  
  
# Matches the exact files either package.json  
or .travis.yml  
[{package.json,.travis.yml}]  
indent_style = space  
indent_size = 2
```

Assignment ($\pm 30\text{m}$)



Research one linter and **one formatter** and see what the functionalities are.

Additionally try to get one up and running in your project;

- Linters; help you catch **syntax errors** (f.e. ESLint, Stylelint, Markdownlint)
- Formatters; helps you **format** your code (f.e. Prettier, Editorconfig)

Danny de Vries, 2 years ago | 1 author (Danny de Vries)

```
1  module.exports = {  
2    env: {  
3      browser: true,  
4      commonjs: true,  
5      es6: true  
6    },  
7    extends: "eslint:recommended",  
8    rules: {  
9      'no-console': 1,  
10     },  
11   };
```

Danny de Vries, 2 years ago • Initial co

live demo linters & formatters

Build Tools

build tools

?

Build tools are additional tooling you can use in your project and are mostly used for **automating tasks or transforming code.**

preprocessors, bundlers, postprocessor, compiling #buzzword #buzzword, ai, machine learning

example

sass

.scss gets processed
(compiled) to .css

SCSS

```
1 section {  
2     height: 100px;  
3     width: 100px;  
4     .class-one {  
5         height: 50px;  
6         width: 50px;  
7         .button {  
8             color: #074e68;  
9         }  
10    }  
11 }  
12 }  
13 }
```

CSS

```
1 section {  
2     height: 100px;  
3     width: 100px;  
4 }  
5 section .class-one {  
6     height: 50px;  
7     width: 50px;  
8 }  
9 section .class-one .button {  
10    color: #074e68;  
11 }  
12 }  
13 }
```

example

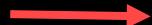
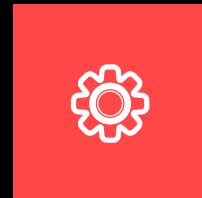
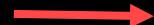
build

// Input

// Build

// Output

```
css/  
└── typography.scss  
└── layout.scss  
└── colors.scss
```



```
css/  
└── style.css
```

example

images

The screenshot shows the GitHub page for the svgo project. The README.md file contains the following content:

SVGO

SVGO Optimizer is a Node.js-based tool for optimizing SVG vector graphics files.

Why?

SVG files, especially those exported from various editors, usually contain a lot of redundant and useless information. This can include editor metadata, comments, hidden elements, default or non-optimal values and other stuff that can be safely removed or converted without affecting the SVG rendering result.

Installation

```
# Via npm
npm -g install svgo
# Via yarn
yarn global add svgo
```

CLI usage

```
# Processing single files:
svgo one.svg two.svg -o one.min.svg two.min.svg
# Processing directory of svg files, recursively using '-f', '--folder' :
svgo -f ./path/to/folder/with/svg/files -o ./path/to/folder/with/svg/output
# Help for advanced usage
svgo --help
```

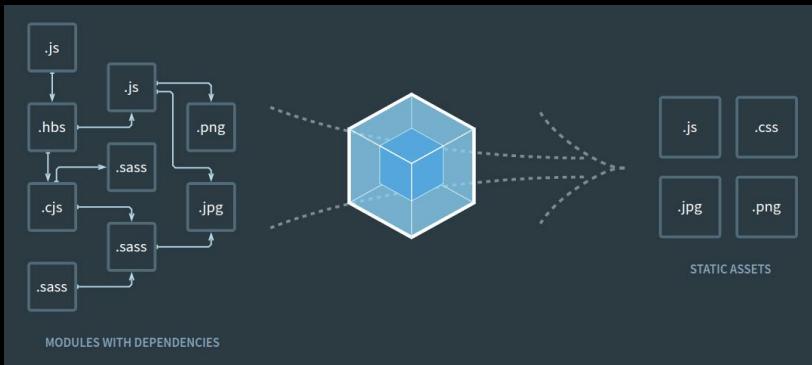
Configuration

SVGO has a plugin-based architecture, separate plugins allows various xml svg optimizations. See [built-in plugins](#).
SVGO automatically loads configuration from `svgo.config.js` or from `--config ./path/myconfig.js`. Some general options can be configured via CLI.

```
// svgo.config.js
module.exports = {
```

file	before	after	profit
user-icon.svg	15.666 Kib	7.354 Kib	53.1%
desc.svg	12.23 Kib	4.621 Kib	62.2%
arrow_first.svg	2.825 Kib	0.373 Kib	86.8%
arrow_left.svg	2.521 Kib	0.294 Kib	88.3%
question.svg	1.092 Kib	0.656 Kib	39.9%
arrow-collapsed-ltr.svg	1.018 Kib	0.122 Kib	88%
arrow-expanded.svg	1.017 Kib	0.124 Kib	87.8%
arrow-down-icon.svg	1.017 Kib	0.124 Kib	87.8%
arrow-down-focus-i...	1.017 Kib	0.124 Kib	87.8%
arrow-collapsed-rtl.svg	1.017 Kib	0.124 Kib	87.8%

Types



Bundlers (automation)



Processors (transpiling)

```
Danny de Vries, 2 years ago | 1 author (Danny de Vries)
1  module.exports = {
2    env: {
3      browser: true,
4      commonjs: true,
5      es6: true
6    },
7    extends: "eslint:recommended",
8    rules: {
9      'no-console': 1,
10     },
11   };
```

live demo build tools

Non-Virtual “Tooling”

- ❖ Stay hydrated
- ❖ Get enough sleep
- ❖ Reduce your brightness
- ❖ Make your workspace comfortable
- ❖ Take enough breaks



building

Start building out the interface (html & css) of your matching-app feature.

Synopsis

- Time: 6:00h
- Due: before week 3

Assignment

Based on the concept, job story, requirement list and wireframe from the previous week start building out the front-end of the feature you are going to make for the matching-application. You can build the interface with the **templating engine** as you learned in back-end as opposed to 'plain' HTML & CSS.

- Turn your wireframe **into HTML pages**. Do a HTML breakdown of your wireframe and use semantic HTML as learned in previous courses.

work on local, linting, build

exit;

see you in lab-4!