

Scientific Experiment Browser

Stakeholders: Daniel Garijo, Yolanda Gil

Deliverable #7: Documentation

April 10, 2017

Huy Ngo

Andrew Nguyen

Tiffany Truong

Table of Contents

| | |
|---------------------------|----------|
| Setting Up | 2 |
| Viewing the project | 2 |
| Setting up project | 2 |
| Some React Specific Notes | 2 |
| HTML Files | 3 |
| index.html | 3 |
| workflow-main.html | 3 |
| CSS Files | 4 |
| landing-page.css | 4 |
| workflow-style.css | 4 |
| Javascript Files | 5 |
| bundle.js/bundle.js.map | 5 |
| main.js | 5 |
| Title.jsx | 5 |
| visualization.js | 5 |
| query.js | 10 |
| change-tabs.js | 11 |
| workflow-searchbar.js | 13 |
| workflowInfoScript.js | 14 |

Setting Up

Viewing the project

The project's repository can be viewed at <https://github.com/tetruong/Portal2.0>. To view the portal itself, click the "[View Portal](#)" link in the README file.

Setting up project

1. git clone <https://github.com/tetruong/Portal2.0.git>
2. If making changes to .jsx files, run command `“./node_modules/.bin/webpack -d --watch”` to recognize changes and automatically recompile .jsx changes
 - a. This will automatically update bundle.js, bundle.js.map
3. In our development, we used Brackets (<http://brackets.io/>)- this allows you to spin up a local instance of the website and test changes live.
4. Note that d3.min.js, dagre-d3.min.js, are external .js files used for the visualization

Some React Specific Notes

To configure the project, look at 'package.json' and 'webpack.config.js'. In these files, you can manage dependencies, entry points to project, and add command-line scripts to aid with development. For example, in package.json, in the 'scripts' section, there is the 'start' script, which will spin up a local instance at localhost:portNumber, where portNumber is specified in webpack.config.js.

HTML Files

index.html

HTML file for home/landing page of the site. This is where one can search for workflows and proceed to view a workflow page.

workflow-main.html

HTML file for displaying a workflow. It has two tabs (Workflow and Execution) on which to view separate visualizations. Here, the user can explore nodes in the visualization by clicking on process/variable nodes and having panels display to the right of the visualization. On the Execution tab, the user can explore different execution traces via a selection menu as well as view the metadata associated with a given execution trace.

CSS Files

landing-page.css

CSS file that contains the CSS styles of index.html.

workflow-style.css

CSS file that contains the CSS styles of workflow-main.html

Javascript Files

bundle.js/bundle.js.map

Auto-generated and updated file by React's webpack.

main.js

This file serves as the entry point to the project, where it renders the Title and search-bar React components. The entry point of the project can be changed within webpack.config.js.

Title.jsx

This file simply creates the title of the landing page and adds content to the html element with id 'title' in index.html.

visualization.js

This file contains the logic required to render the visualization of a workflow or execution trace. Data visualization libraries called D3.js (<https://github.com/d3/d3/wiki>) and Dagre.js (dagre-d3, more specifically <https://github.com/cpetitt/dagre-d3>) are used.

The visualization is comprised of a series of nodes and edges. Nodes are a general term to represent all variables and processes while edges represent the connections between nodes. Each node object, when set in the visualization, corresponds to a specific index inside the visualization. When setting a node object, you can set its label, labelStyle, shape, style in order to customize the look of the node depending on its name and whether or not it is an process or variable (for more on setNode method, look at d3/dagre-d3 specific functions table). When setting edges, you can also style them to your liking.

Since nodes and edges are regular Javascript objects, you can also add key-value pairings to incorporate extra data about a node. For example, a node's URI can be very helpful if some data about that process/variable needs to be queried from a repository's endpoint. Thus, when calling setNode, in addition to specifying other metadata for a node, we also include a corresponding URI.

| Helpful dagre-D3/D3 functions | |
|-------------------------------|--|
| setNode | <p>Input:</p> <ol style="list-style-type: none">1. Integer id, a unique number to identify the node in a graph2. An object containing a node's metadata <p>Output: Object containing information about entire graph</p> <p>Example usage:</p> <pre>graph.setNode(1, { label: 'nodeLabelExample', shape: 'ellipse', style: 'fill: #fff',</pre> |

| | |
|-------------------|---|
| | somethingUseful: 'useful-string-to-include' }); |
| setEdge | Input: <ol style="list-style-type: none"> 1. Integer id of the source node 2. Integer id of the destination node 3. An object containing a node's metadata Output: Object containing information about entire graph Example usage: <pre>graph.setEdge(1, 2, { style: 'stroke: #000; fill:none;', });</pre> |
| graphObject.nodes | Input: none Output: Array of integer node ids |
| graphObject.node | Input: integer id of a node Output: Object containing all metadata/details of a given node |

| Variables | |
|----------------------|---|
| svg/svgGroup | D3 selections used to manipulate UI/UX of the graph |
| vis | The graph object- setNode and setEdge are member functions of this object |
| zoom | Used to setup zoom behavior (https://github.com/d3/d3-zoom) |
| processInputMapping | Object acting like hash table Keys: URI strings of processes Values: Array of URI strings of input variables |
| processOutputMapping | Object acting like hash table Keys: URI strings of processes Values: Array of URI strings of output variables |
| isVariableOfMapping | Object acting like hash table Keys: URI strings of variables that are used by processes as inputs Values: Array URI strings of processes that use the variable as a parameter |

| | |
|--------------------|---|
| outputByMapping | Object acting like hash table Keys: URI strings of variables output by process Values: Array of URI strings of processes that output the variable |
| processNodeIndices | Object acting like hash table Keys: URI strings of processes in visualization Value: integer index of node in visualization |
| putNodeIndices | Object acting like hash table Keys: URI strings of variables in visualization Value: integer index of node in visualization |
| workflowURI | String of URI of workflow/execution trace being rendered |
| results | Object containing query response with URIs for processes/variables required for visualization |

| Functions | |
|---------------------|--|
| renderVisualization | <p>Input:</p> <ol style="list-style-type: none"> 1. AJAX response 'res' 2. boolean 'isTrace' <p>Output: none</p> <p>Encompasses the logic required to set the nodes and edges of visualization and display to an svg element in workflow-main.html</p> |
| addInputProcess | <p>Input:</p> <ol style="list-style-type: none"> 1. String URI of process 'processName' 2. String URI of input 'inputName' <p>Output: none</p> <p>Adds inputName to the list of inputs for a given processName.</p> |
| addOutputProcess | <p>Input:</p> <ol style="list-style-type: none"> 1. String URI of process 'processName' 2. String URI of output 'outputName' <p>Output: none</p> <p>Adds outputName to the list of inputs for a given processName.</p> |

| | |
|----------------------|--|
| mapInputToProcess | <p>Input:</p> <ol style="list-style-type: none"> 1. String URI of input 'inputName' 2. String URI of process 'processName' <p>Output: none</p> <p>Adds processName to the list of processes that use variable inputName</p> |
| mapOutputFromProcess | <p>Input:</p> <ol style="list-style-type: none"> 1. String URI of output 'outputName' 2. String URI of process 'processName' <p>Output: none</p> <p>Adds processName to the list of processes that output variable outputName</p> <ul style="list-style-type: none"> - We believe that an output can only correspond to 1 process exactly, so the implementation of this can change in the future to reflect that |
| mapNodesEdges | <p>Input:</p> <ol style="list-style-type: none"> 1. Graph object 'graph' <p>Output: none</p> <p>Parse 'results' object to set nodes of graph. Adds data to processOutputMapping, processInputMapping, isVariableOfMapping, outputByMapping, putNodeIndices, and processNodeIndices.</p> |
| setGraphEdges | <p>Input:</p> <ol style="list-style-type: none"> 1. Graph object 'vis' <p>Output: none</p> <p>Sets graph edges based on node indices from putNodeIndices and processNodeIndices.</p> |
| formatInputs | <p>Input:</p> <ol style="list-style-type: none"> 1. Graph object 'vis' 2. Function 'callback' <p>Output: none</p> <p>Calls getInputs from query.js to retrieve the inputs of a workflow and restyles the nodes determined to be inputs. After restyling nodes, executes the 'callback' function</p> |

| | |
|------------------------|---|
| renderGraph | <p>Input:</p> <ol style="list-style-type: none"> 1. Graph object 'vis' <p>Output: none</p> <p>Populates svg element in workflow-main.html with visualization. Sets up height, width, zoom feature, centers graph.</p> |
| setupNodeOnClick | <p>Input:</p> <ol style="list-style-type: none"> 1. D3 selection object 'svg' 2. Graph object 'vis' <p>Output: none</p> <p>Setup on-click listeners for each node to display necessary information panels</p> |
| translateVisualization | <p>Input: none</p> <p>Output: none</p> <p>Translates visualization when information panels display</p> |
| addTraces | <p>Input: Object 'traces'</p> <p>Output: none</p> <p>Populates execution trace options in selection menu and adds event listeners for when the selection option is changed between different execution traces.</p> |
| highlightPuts | <p>Input: Array of URIs of inputs/outputs 'putsArray'</p> <p>Output: none</p> <p>Highlights the inputs/outputs of a given process node</p> |
| unhighlightAllPuts | <p>Input: none</p> <p>Output: none</p> <p>Unhighlights the inputs/outputs of all processes</p> |
| unhighlightPuts | <p>Input: Array of URIs of inputs/outputs 'putsArray'</p> <p>Output: none</p> <p>Unhighlights the inputs/outputs of a given process node</p> |
| stripNameFromURI | <p>Input: string 'uri'</p> |

| | |
|---------------|---|
| | <p>Output: substring of 'uri' containing label of URI</p> <p>Parses URI for the relevant label to display</p> |
| addDimensions | <p>Input: dagre-d3 renderer object 'render'</p> <p>Output: none</p> <p>Adds dimensions functionality for workflows by creating custom shapes for nodes (https://github.com/cpettt/dagre-d3/blob/master/lib/shapes.js).</p> |

query.js

This file contains functions containing SPARQL queries and AJAX calls required to populate the workflow page's dynamic content. There is a main endpoint set to query from at the top of the file.

| Functions | |
|-------------------|---|
| populateSearchBar | <p>Input:</p> <ol style="list-style-type: none"> Function 'handler' <p>Output: none</p> <p>Queries for the suggestions for auto-complete feature</p> |
| getInputs | <p>Input:</p> <ol style="list-style-type: none"> String 'workflow' URI Function 'handler' <p>Queries for the inputs of a workflow so their nodes can be styled properly in visualization</p> |
| getWorkflowData | <p>Input:</p> <ol style="list-style-type: none"> String 'workflowURI' Function 'handler' <p>Queries for the data of each node/edge in visualization of a workflow</p> |
| getExecutionID | <p>Input:</p> <ol style="list-style-type: none"> String 'workflowURI' Function 'handler' <p>Queries for the execution IDs of a given workflow and calls addTraces to populate options for selection box of execution traces</p> |

| | |
|----------------------|---|
| getExecutionData | Input: <ol style="list-style-type: none"> 1. String 'executionID' 2. Function 'handler' Queries for the data of each node/edge in visualization of an execution |
| getExecutionMetadata | Input: <ol style="list-style-type: none"> 1. String 'executionID' 2. Function 'handler' Queries for the metadata of an execution (status, time started, ended, when execution account was created) |

change-tabs.js

This file contains the logic in how to handle when a user switches between the 'Workflow' and 'Execution' tab. When the Execution tab is selected, icons indicating an execution trace's metadata and a selection box will show on top of a new visualization.

| Functions | |
|---------------------|---|
| setWorkflowMetadata | Input: <ol style="list-style-type: none"> 1. Object 'res' Output: none Upon receiving query results, populates the metadata of an execution |

search-bar.jsx

This file contains the react component for the autocomplete search bar in the index.html page. The main feature of this react component is the autocomplete functionality. In query.js, there is an AJAX function (populateSearchBar) that queries the endpoint and fetches a json file of the workflow URIs and labels. When our react component is first mounted, it calls populateSearchBar and parses the json and stores the workflow label and URI in an array (workflowSuggestions). Each entry in the array is a map in the following format: label:<workflowLabel>, uri:<workflowURI>. This list of suggestions is all that is needed to enable the autocomplete functionality in our react component. More documentation on the react component we used can be found at <https://github.com/moroshko/react-autosuggest>.

| Variables | |
|---------------------|---|
| workflowSuggestions | Array containing the workflow labels and corresponding URIs. Each entry in the array is |

| | |
|--|--|
| | stored in the format: {label:<workflowLabel>, uri:<workflowURI>} |
|--|--|

| Functions | |
|-----------------------|---|
| escapeRegexCharacters | <p>Input: string 'str' Output: regex escaped string</p> <p>Escapes user input to be treated as a literal string within a regular expression</p> |
| getSuggestions | <p>Input: user input in the search box 'value' Output: list of suggestions that match the user input</p> <p>Given a user input, find all the values in workflowSuggestions that match the user input and return them.</p> |
| getSuggestionValue | <p>Input: suggestion from the list of suggestions 'suggestion' Output: string</p> <p>Returns an input value when a suggestion is clicked. Right now, we are returning the label of the suggestion.</p> |
| renderSuggestion | <p>Input: selected suggestion 'suggestion', user input 'query' Output: string or react element</p> <p>Defines how suggestions are rendered. Currently, we encapsulate the matching portion of the suggestion with tags and return it as a react element.</p> |
| parseAutocompleteData | <p>Input: json file 'res' Output: none</p> <p>We parse res and insert an entry for every label+uri pair into our workflowSuggestions array.</p> |
| componentDidMount | <p>Input: none Output: none</p> <p>Invoked immediately after the component is mounted. Here, we initialize our</p> |

| | |
|-----------------------------|--|
| | workflowSuggestions. |
| onChange | <p>Input: Action Event 'event', input 'newValue', string 'method'</p> <p>Output: none</p> <p>Changes the 'value' variable in our react component to match the user input.</p> |
| onSuggestionsFetchRequested | <p>Input: user input 'value'</p> <p>Output: none</p> <p>Called every time we need to update the suggestions (each time the user modifies the input).</p> |
| onSuggestionsClearRequested | <p>Input: none</p> <p>Output: none</p> <p>Clears the dropdown list of suggestions</p> |
| onSuggestionSelected | <p>Input: the selected suggestion 'suggestion', the value of the suggestion 'suggestionValue', index in the array 'suggestionIndex'</p> <p>Output: none</p> <p>Called when a suggestion is selected. We store workflowSuggestions and the selected suggestion's URI and label in local storage, then change pages to workflow-main.html.</p> |

workflow-searchbar.js

This file contains the jQuery UI Autocomplete functions used in the search bar in the workflow-main.html page. The functionality is the same as the React component in the index.html page, but it uses jQuery instead.

| Functions | |
|-------------------------|---|
| monkeyPatchAutocomplete | <p>Input: none</p> <p>Output: none</p> <p>Overriding the _renderItem function in the default jQuery UI Autocomplete widget so that we can highlight the part of the suggestions that the user input matches</p> |

workflowInfoScript.js

This file contains all of the functions necessary for adding the panels of information to the page when a user clicks on one of the nodes in a visualization on a workflow page. This adds panels to the page in two separated sections, one for process information, and one section for variable information. These get added to the right side of the page and will dynamically resize the visualization graph as needed.

| Variables | |
|-------------------------|---|
| processInfosCount | Counter for how many process information panels are currently showing on the screen (range 0-4) |
| processInfosIndex | Counter for how many process information sections have been added to the page. Counts up so that every panel added has a unique index. |
| variableInfosCount | Counter for how many variable information panels are currently showing on the screen (range 0-4) |
| variableInfosIndex | Counter for how many variable information sections have been added to the page. Counts up so that every panel added has a unique index. |
| variableSectionsShowing | List of the names of the variable information panels that are currently showing. |
| processSectionsShowing | List of the names of the process information panels that are currently showing. |
| \$template | jQuery object for the template of each process information panel. |
| \$templateVariables | jQuery object for the template of each variable information panel. |
| \$vis | jQuery object for the visualization container to be used for resizing it dynamically. |

| Functions | |
|----------------|---|
| addProcessInfo | Inputs: process URI string 'processURI', array of process input variables' URIs 'inputsArray', array of process output variables' URIs 'outputsArray' |

| | |
|--------------------|---|
| | <p>This checks for the number of already showing process information panels:</p> <ul style="list-style-type: none"> - If 0 variable and process panels are showing, it resizes the visualization to make room for the information panels - If 4 are showing the top one (least recently added panel) is removed to make room for the new panel - Additionally, the process name is checked for duplicates so it does not get displayed twice <p>A new panel is created by making a copy of the template and populating the information about the process's name, and input and output variables list before it is added to the page.</p> |
| addVariableInfo | <p>Inputs: variable URI string 'variableURI', array of process URIs that use the variable 'usedBy', process URI of the process that generates this variable 'generatedBy', and 'variableType'</p> <p>This checks for the number of already showing variable information panels:</p> <ul style="list-style-type: none"> - If 0 variable and process panels are showing, it resizes the visualization to make room for the information panels - If 4 are showing the top one (least recently added panel) is removed to make room for the new panel - Additionally, the variable name is checked for duplicates so it does not get displayed twice <p>A new panel is created by making a copy of the template and populating the information about the variable's name and information before it is added to the page.</p> |
| clearAllPanels | <p>Inputs: none Outputs: none</p> <p>Removes all of the information panels (variable and process) from the page, resets counters and arrays keeping track of displayed information sections.</p> |
| removeVariableInfo | <p>Input: name of the variable who's panel is to be removed 'removeID'</p> |

| | |
|-------------------|---|
| | Removes the panel associated with that variable ID from the page and from the arrays keeping track of their information. |
| removeProcessInfo | Input: name of the process who's panel is to be removed 'removeID' Removes the panel associated with that process ID from the page and from the arrays keeping track of their information. |