Torbit: Front End Programming Challenge
Cecchi MacNaughton

Functionality

The functionality of the dashboard is fairly straightforward. The graph is initially populated with random data, and can be updated using the textarea below the graph. Individual data lines can be toggled on and off using the legend (clicking the check icon will toggle that line, while clicking the name itself will cause only that line to be shown). The view finder is used by dragging the handles from either edge inward to "crop" the range of time over which data should be displayed. The graph supports two graphing modes: "stacked" and "independent". In "stacked" mode, the frontend loading speed is used as the "zero" point for the backend loading speed. In other words, the combined areas displayed on the graph represent the total loading time. In "independent" mode, the two data lines are graphed independently of each other as in a standard line graph.

Code Design

The dashboard is based on a module pattern. The app contains one main object, Dashboard, which is an empty module but in a more complex, hypothetical version of the dashboard this would include central code related to the dashboard as a whole. There are five submodules:

`Dashboard.Elements`

The Elements module is the simplest module, and simply stores references to commonly used DOM nodes.

`Dashboard.Controls`

The Controls module keeps track of user input and interaction. In particular this module adds event listeners to handle when the user changes the graphing mode (stacked or independent) or updates the data.

`Dashboard.Data`

The Data module handles everything relating to parsing the input into a format usable by Rickshaw, and storing the data in a way such that the main graph and the view finder can access it independently. In addition the Data module has a variety of methods for creating a random set of data.

`Dashboard.Graph`

The Graph module is the main module for initializing, displaying, and updating the main graph.

`Dashboard.ViewFinder`

The ViewFinder module creates a second graph, a clone of the main graph, which can be used to zoom in on the main graph. Rickshaw has a similar functionality built in but it relies on jQuery and is a simple slider as opposed to a graphical view finder.

Limitations

The code makes use of array iteration methods introduced in JavaScript 1.6 and 1.8, meaning the application will not work natively in some older browsers. These methods can easily be added with shims (i.e. https://github.com/kriskowal/es5-shim) to increase browser support, however the graphing library used, Rickshaw, only supports back to Internet Explorer 9.

3rd Party Code

The dashboard is built on top of Rickshaw, a graphing library developed at Shutterstock. Rickshaw is in turn built on top of d3.js. Beyond that, the code is entirely mine except where otherwise mentioned in the comments. The following methods `ViewFinder.getOffset()` and `ViewFinder.preventHighlight()` are based on snippets from StackOverflow answers. I modified `Rickshaw.Graph.Legend` slightly to remove its jQuery dependency.