JavaScript Logo Project

By John Sandoval

Project Timeline

- 11.22.2014 - Begin the project (Research, mock-ups, plan document)
- 11.23.2014 - Begin development
- 11.29.2014 - Complete development
- 11.30.2014 - Complete project artifacts

Project Challenges

- Interactive image manipulation in a web browser
- Time
- Showing skills

Project Technologies

- **HTML, CSS, JavaScript** (USED)
  - HTML DOM first, then Canvas and/or SVG if time
- **SVG** (NOT USED)
  - SVG coordinates scale automatically, making this a nice responsive option
  - SVG DOM is not entirely accessible by JavaScript, especially for drop events
- **LESS** (USED)
  - Common extension and abstraction of CSS
- **Twitter Bootstrap** (USED)
  - CSS alone makes creating common elements faster; JS will not be used
- **jQuery and jQuery-UI** (USED)
  - Micro-framework saves time wiring up basic DOM manipulations
  - jQuery-UI: only drag, drop, and core libraries will be used (37 KB before gzip)
- **Mocha test framework** (USED)
  - Can be used from CLI, browser, or Grunt
  - Chai assertion library
- **Grunt task runner** (USED)
  - Automates LESS processing, JS/CSS file concatenation, JS/CSS linting, JS/CSS minification, test dependencies, unit testing, test coverage reporting, etc.
- **Dependency package managers** (USED)
  - NPM for Grunt automation, Bower for static assets
- **Git SCM** (USED)
  - Git Bash for Windows
- **GitHub remote repository** (USED)

Project Development Approach

1. Create a standard DOM version first
2. Complete an automation framework
3. Re-factor for modularity and testing
4. Create unit tests and a test reporter
5. Create test coverage reporting
6. Create a 2nd version of the logo app using the Canvas object (🕒)

🕒 = *time permitting*

Project Development Steps

1. Configure NPM and Bower package managers, install dependencies
2. Set up automated dev/build environment
    a. Grunt tasks for watch, pre-processing, concatenation, and minification
3. Create static HTML/CSS layout
4. Create the external JS file with a basic IIFE
    a. Create a “state” object that will track the state of each drag/drop
        i. Start with a mutable singleton (re-factor later)
    b. Add and confirm draggable functionality
        i. Dragstart, dragend
    c. Create DOM drop targets and gain control over droppable events
        i. Dragover, dragleave, drop
    d. Create a function to satisfy each of the functional requirements
        i. Invalid drops should be rejected, notify the user when complete, etc
    e. Create functions for usability
        i. Hover helpers, animations/easing, etc.
5. Evaluate effort to re-code without jQuery-UI (🕒)
    a. Event listeners should be relatively quick
    b. Full control over event bubbling, event timing, dataTransfer(), browser quirks, etc. might take more time than I have.
6. Refactor the IIFE
    a. Convert the singleton to a factory function
    b. Convert the named functions into methods on a public object
    c. Ensure that all functional code is removed from event configurations
7. Create unit tests
    a. Set up a standalone HTML reporter using Mocha and Chai
    b. Write a test for the application state object
    c. Write tests for methods used by drag-type and drop-type events
8. Add tests to automation framework

   a. Install Karma and Grunt packages to project
   b. Configure Karma to launch PhantomJS as a headless browser
   c. Configure Grunt tasks to include tests as part of watch
9. Add test coverage reporting
   a. Install and configure Karma coverage tools
   b. Add coverage reporting to automation framework
   c. Update code/tests based on coverage reports
   d. *We are now Linting, Testing, Reporting Coverage, Concatenating, and Minifying our JS quickly and automatically on each file save!*
10. Make sure that the README markdown file is up to date

Project Lessons

- Elements inside SVG objects are very difficult to access with JS. The SVG DOM is not rendered in the same namespace as the HTML DOM
- Quickly playing around with basic mock-ups during planning showed:
  - HTML DOM dragging slows due to large number of elements
  - Canvas dragging slows due to large size of canvas
  - Chrome and Firefox began slowing before IE

Project Notes

- GitHub repo: https://github.com/jsandoval81/drag-and-drop.git
- Full installation instructions, including dependencies, are available on the GitHub page
- Library and package versions can be found in the `bower.json` and `package.json` files