simpl.info developer's guide

This site is no longer maintained.

Most demos should continue to work — fixes welcome!

The repo for simpl.info is at github.com/samdutton/simpl.

Before contributing code, please check the CONTRIBUTING file.

Validation

HTML, CSS and JavaScript should follow the Google style guides.

All JavaScript must pass ESLint validation and follow the options in samples/.eslintrc. ESLint plugins are available for Sublime and other editors, and ESLint validation can be run as a Grunt task. If necessary (rarely), include additional ESLint directives within individual JavaScript files.

All JavaScript files must have `'use strict';` at the top, to invoke global strict mode.

All HTML must pass HTMLHint checking. Likewise, CSSLint must not find errors (though some warnings are acceptable).

The W3C validators for CSS and HTML are also useful.

Project structure and style

For each new code sample, create a new top level directory.

Put JavaScript in a separate file. Each demo should have a directory structure like this:

```
index.html
    /js/main.js
```

If your CSS has more than about five rules, put it in a separate file:

```
/css/main.css
```

Declare all global variables and element variables at the top of main.js.

Put all conditional statements in a block, even if only one line.

By default use === and !== for testing equality. This makes equality testing explicit, without coercion.

Put a semicolon at the end of every statement. This is easy to forget with handlers, for example:

```
stream.onFoo = function(){...}; // this is a statement
```

Prefer [] to new Array(), for example:

```
var myArray = [];
```

Prefer dot nation. For example use this:

```
pcConfig.iceServers[i].url
```

...instead of this:

```
pcConfig.iceServers[i]['url'];
```

Double quote string values in HTML, single quote in JavaScript (as per the Google style guide).

JavaScript variable names use camelCase. In general, don't use hyphens in names.

Indent with two spaces. When function calls or tests run onto multiple lines, indent with four spaces.

Validation with Grunt plugins and Travis

This project has been set up to enable automated testing with Grunt plugins and Travis.

Grunt

Grunt uses Node.js to automate tasks written in JavaScript. Two files in a project's top level code directory provide information for Grunt to be used with that project:

- package.json describes dependencies and other project data.
- Gruntfile.js defines options for the tasks which can be run with Grunt for the project.

  For example, the Gruntfile for the simpl.info repo gives options for three Grunt plugins:

CSSLint, HTMLHint, ESLint. These are predefined tasks used to validate code and check coding style. It's also possible to write your own custom tasks using JavaScript – for unit testing, for example.

The `grunt` command can be called with or without parameters, depending on settings defined in the project's Gruntfile. For example, with the simpl.info repo, `grunt htmlhint` will run the HTMLHint Grunt plugin. Calling `grunt` on its own runs all the tasks defined by the `grunt.registerTask()` method in the Gruntfile.js:

```
grunt.registerTask('default',
  ['csslint', 'htmlhint', 'eslint']);
```

The grunt-eslint documentation has simple instructions on how to set up Grunt with ESlint: other plugins are installed in the same way. The Grunt plugins page has more information about popular plugins. Options for Grunt plugins can generally be defined in a project's Gruntfile as well as in config files (such as .eslintrc and .csslintrc).

Travis

Once a GitHub project has been configured on the Travis website, the Travis server can hook into the GitHub API to respond to events for that project. For example, the public project travis-ci.org/samdutton/simpl corresponds to the GitHub project github.com/samdutton/simpl. (Travis can also be used with private projects.)

Once Travis has been given access to the GitHub API for a project, GitHub push or pull events trigger testing (and potentially other build steps) to be done by the Travis server. Whenever the simpl.info repo is pushed or pulled, the Travis server runs the following steps (this build log gives more detail):

1. Install and test Node dependencies as defined by package.json in the project's top level directory.
2. Call the command defined by the scripts.test property in package.json, which for this project is as follows:

   ```
   "scripts": {
     "test": "grunt --verbose"
   }
   ```

   As described above, calling the `grunt` command on its own for this project will run three tasks as defined in Gruntfile.js: CSSLint, HTMLHint and ESLint.

3. Once all tests have been completed, Travis reports success or failure. Travis can be configured to motorola.com/us/error various types of notification. Currently this project is set to send an email to the project owner (samdutton@gmail.com). Travis also updates a project status icon which can be included in documentation on GitHub:

   ![Travis](https://travis-ci.org/samdutton/simpl.svg?branch=master)

   ue 

The travis.yml file at the project's top level directory defines the language environment, which in this case is Node.js, and can be used for other settings.

Travis can also be used to automate build tasks, though this is not currently used by simpl.info.