A Drip of JavaScript

The Virtue of JavaScript Linting

Originally published in the A Drip of JavaScript newsletter.

One of the great benefits of being a developer is the extent to which we can automate away many of the necessary but tedious parts of our jobs. And sometimes the value of automating something is actually greater than doing it by hand. This is particularly true of linting.

A linter is a program that can analyze source code in a particular programming language and flag potential problems like syntax errors, deviations from a prescribed coding style, or using constructs known to be unsafe. For example a JavaScript linter would flag the first use of `parseInt` below as unsafe:

```
// Unsafe without a radix argument
var count1 = parseInt(countString);

// Safe because a radix is specified
var count2 = parseInt(countString, 10);
```

Of course, you could always examine all of your code by hand to determine whether an error was introduced, but that requires a great deal of expertise. And it quickly becomes impractical as the size of your codebase increases, requiring ever greater amounts of caffeine to allow a human being to maintain sufficient attention to read it. (You should probably stop well before reaching a lethal dose.) A linter can check the entire codebase in seconds and help you keep the number of bugs to a minimum.

The two best known linters for JavaScript are JSLint and JSHint. JSLint was originally authored by Douglas Crockford, but it was sometimes seen as too inflexible and

dogmatic. Because of this concern, JSHint was born as a more flexible linting tool which gives users more control over what sort of issues it should flag as potential problems.

Getting started with linting is easy. Both JSHint and JSLint have web-based tools that will let you paste in code to check. But you can also use their command-line tools to do project-wide linting.

Let's walk through what it takes to set that up with JSHint. To get started, you'll want to install Node.js, which will give you the ability to execute JavaScript programs (like JSHint) from the command-line.

After you have Node installed, you can install JSHint by entering `npm install -g jshint` at the command-line. Depending on your permissions and operating system you may need to preface that command with `sudo`.

Once you have JSHint installed, navigate to a directory which contains some JavaScript files and enter `jshint *.js`. Chances are you will soon have a list of warnings and errors printing to your terminal. It should look something like this:

```
example.js: line 1, col 34, Missing radix parameter.
example.js: line 5, col 14, Missing semicolon.
example.js: line 7, col 17, Use '!==' to compare with 'null'.

3 errors
```

Of course, you don't necessarily have to use JSHint's default settings. Suppose that you typically prefer to use `===` but in a particular project you want to allow the use of `x != null` (as mentioned in a previous drip).

Fortunately, JSHint supports a project settings file to specify how your project should be linted. Simply create a file named `.jshintrc` in the top directory of your project. The options should be specified as properly formatted JSON like the following:

```
{
    "eqnull": true
}
```

From now on, any time JSHint is run anywhere inside your project directory, it will use the options you specified. For a list of the available options, see the JSHint documentation.

Using a tool like JSHint or JSLint is a great way to save time and improve the quality of your code all at once.

Thanks for reading!

Joshua Clanton
