Chapter 102: Linters - Ensuring code quality

Section 102.1: JSHint

JSHint is an open source tool which detects errors and potential problems in JavaScript code.

To lint your JavaScript you have two options.

1. Go to JSHint.com and paste your code in there on line text editor.
2. Install JSHint in your IDE.
    - Atom: linter-jshint (must have Linter plugin installed)
    - Sublime Text: JSHint Gutter and/or Sublime Linter
    - Vim: jshint.vim or jshint2.vim
    - Visual Studio: VSCode JSHint

A benefit of adding it to your IDE is that you can create a JSON configuration file named `.jshintrc` that will be used when linting your program. This is convent if you want to share configurations between projects.

Example `.jshintrc` file

```
{
    "-W097": false, // Allow "use strict" at document level
    "browser": true, // defines globals exposed by modern browsers
http://jshint.com/docs/options/#browser
    "curly": true, // requires you to always put curly braces around blocks in loops and
conditionals http://jshint.com/docs/options/#curly
    "devel": true, // defines globals that are usually used for logging poor-man's debugging:
console, alert, etc. http://jshint.com/docs/options/#devel
    // List global variables (false means read only)
    "globals": {
        "globalVar": true
    },
    "jquery": true, // This option defines globals exposed by the jQuery JavaScript library.
    "newcap": false,
    // List any global functions or const vars
    "predef": [
        "GlobalFunction",
        "GlobalFunction2"
    ],
    "undef": true, // warn about undefined vars
    "unused": true // warn about unused vars
}
```

JSHint also allows configurations for specific lines/blocks of code

```
switch(operation)
{
   case '+'
   {
      result = a + b;
      break;
   }

   // JSHint W086 Expected a 'break' statement
   // JSHint flag to allow cases to not need a break
   /* falls through */
```

```
    case '*':
    case 'x':
    {
        result = a * b;
        break;
    }
}

// JSHint disable error for variable not defined, because it is defined in another file
/* jshint -W117 */
globalVariable = 'in-another-file.js';
/* jshint +W117 */
```

More configuration options are documented at http://jshint.com/docs/options/

Section 102.2: ESLint / JSCS

ESLint is a code style linter and formatter for your style guide much like JSHint. ESLint merged with JSCS in April of 2016. ESLint does take more effort to set up than JSHint, but there are clear instructions on their website for getting started.

A sample configuration for ESLint is as follows:

```
{
    "rules": {
        "semi": ["error", "always"], // throw an error when semicolons are detected
        "quotes": ["error", "double"] // throw an error when double quotes are detected
    }
}
```

A sample configuration file where ALL rules are set to off, with descriptions for what they do can be found here.

Section 102.3: JSLint

JSLint is the trunk from which JSHint branched. JSLint takes a much more opinionated stance on how to write JavaScript code, pushing you towards only using the parts Douglas Crockford deems to be its "good parts", and away from any code that Crockford believes to have a better solution. The following StackOverflow thread may help you decide which linter is right for you. While there are differences (here are some brief comparisons between it and JSHint / ESLint), each option is extremely customizable.

For a more information about configuring JSLint check out NPM or github.