JavaScript Tools: JSLint & JSHint

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- What is JSLint?
- Things that are being flagged by JSLint
- What is JSHint?

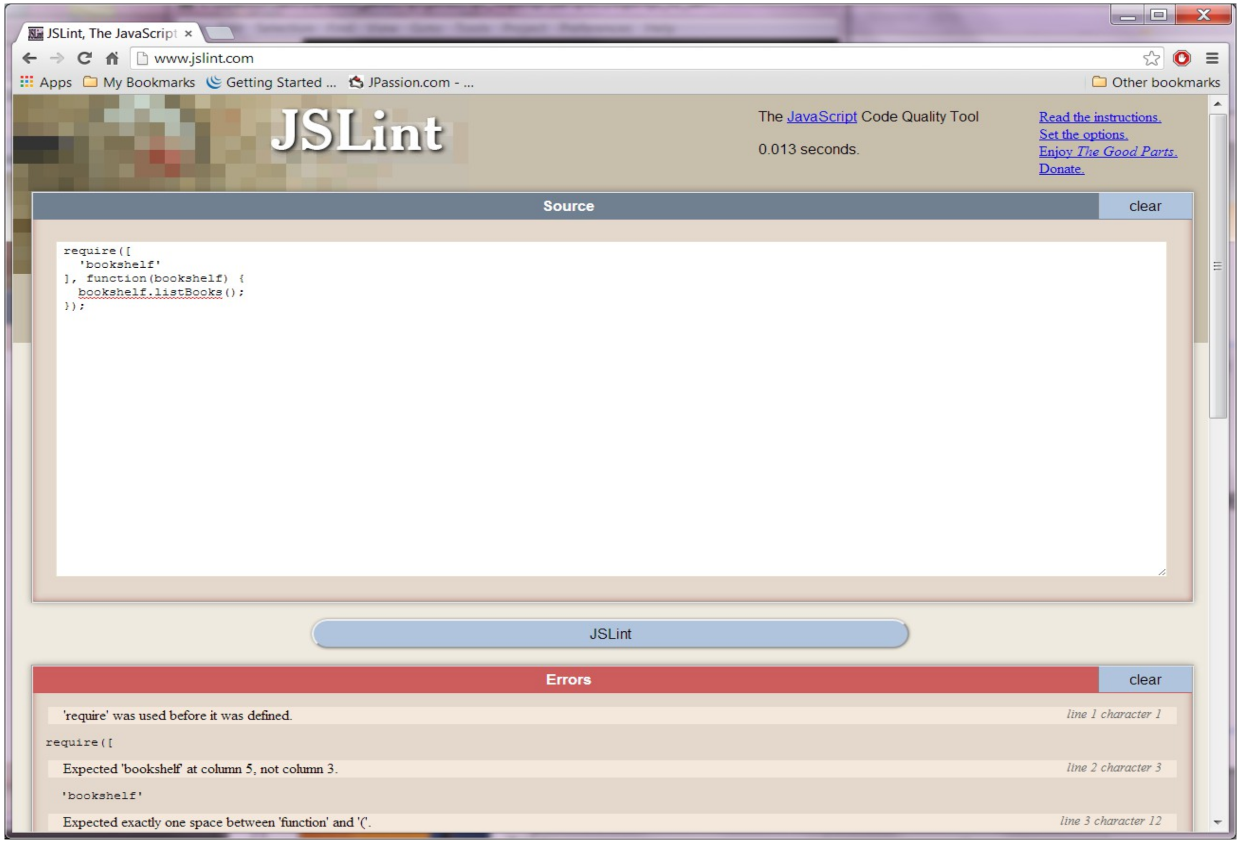
What is JSLint?

What is JSLint?

- JSLint is a static code analysis tool used in software development for checking if JavaScript source code complies with coding rules
- It was developed by Douglas Crockford
- It is provided primarily as an online tool, but there are also command-line version

JSLint Online Tool

- http://www.jslint.com/



```
require([
  'bookshelf'
], function(bookshelf) {
  bookshelf.listBooks();
});
```

'require' was used before it was defined. line 1 character 1

```
require([
```

Expected 'bookshelf' at column 5, not column 3. line 2 character 3

```
 'bookshelf'
```

Expected exactly one space between 'function' and '('. line 3 character 12

JSLint Command Line Tool

- Install “jslint” plugin to your Node.js installation
  > *npm install jslint -g*
- Then run “jslint” command
  > *jslint myapp.js*

JSLint Plugins for text editors and IDEs

- Editors
  - Sublime Text, TextMate, VIM, Emacs, Nodepad++
- IDE's
  - Visual Studio, Eclipse, IntelliJ, NetBeans
- Build tool
  - Maven

Things that are flagged by JSLint

Global Variables

- Issues
  - JavaScript's biggest problem is its dependence on global variables, particularly implied global variables
  - If a variable is not explicitly declared (usually with the “var” statement), then JavaScript assumes that the variable was global. This can mask misspelled names and other problems
- What JSLint expects
  - JSLint expects that all variables and functions are declared before they are used or invoked
  - This allows it to detect implied global variables. It is also good practice because it makes programs easier to read.

```
// The declaration should appear near the top of the file. It must appear
// before the use of the variables it declares.
var getElementByAttribute, breakCycles, hanoi;
```

Scopes

- Issues
  - In many languages, a block introduces a scope. Variables introduced in a block are not visible outside of the block.
  - In JavaScript, blocks do not introduce a scope. There is only function-scope. A variable introduced anywhere in a function is visible everywhere in the function. JavaScript's blocks confuse experienced programmers and lead to errors because the familiar syntax makes a false promise.
- What JSLint expects
  - JSLint expects blocks with function, if, switch, while, for, do, and try statements and nowhere else
  - Because JavaScript does not have block scope, it is wiser to declare all of a function's variables at the top of the function. It is recommended that a single “var” statement be used per function.

var

- Issues
  - > JavaScript allows var definitions to occur anywhere within a function. JSLint is more strict.
- What JSLint expects
  - > JSLint expects that a var will be declared only once, and that it will be declared before it is used.
  - > JSLint expects that a function will be declared before it is used.
  - > JSLint expects that parameters will not also be declared as vars.
  - > JSLint does not expect the arguments array to be declared as a var.
  - > JSLint does not expect that a var will be defined in a block. This is because JavaScript blocks do not have block scope. This can have unexpected consequences. Define all variables at the top of the function.

eval

- Issues
  - eval is evil
  - The eval function (and its relatives, Function, setTimeout, and setInterval) provide access to the JavaScript compiler. This is sometimes necessary, but in most cases it indicates the presence of extremely bad coding. The eval function is the most misused feature of JavaScript.
- What JSLint expects
  - JSLint expects no usage of eval

Constructor Function and “new”

- Issues
  - > Constructor functions are functions that are designed to be used with the new prefix. The new prefix creates a new object based on the function's prototype, and binds that object to the function's implied this parameter. If you neglect to use the new prefix, no new object will be made and this will be bound to the global object. This is a serious mistake.
- What JSLint expects
  - > JSLint enforces the convention that constructor functions be given names with initial uppercase
  - > JSLint does not expect to see a function invocation with an initial uppercase name unless it has the new prefix
  - > JSLint does not expect to see the new prefix used with functions whose names do not start with initial uppercase.

Constructor Function and “new”

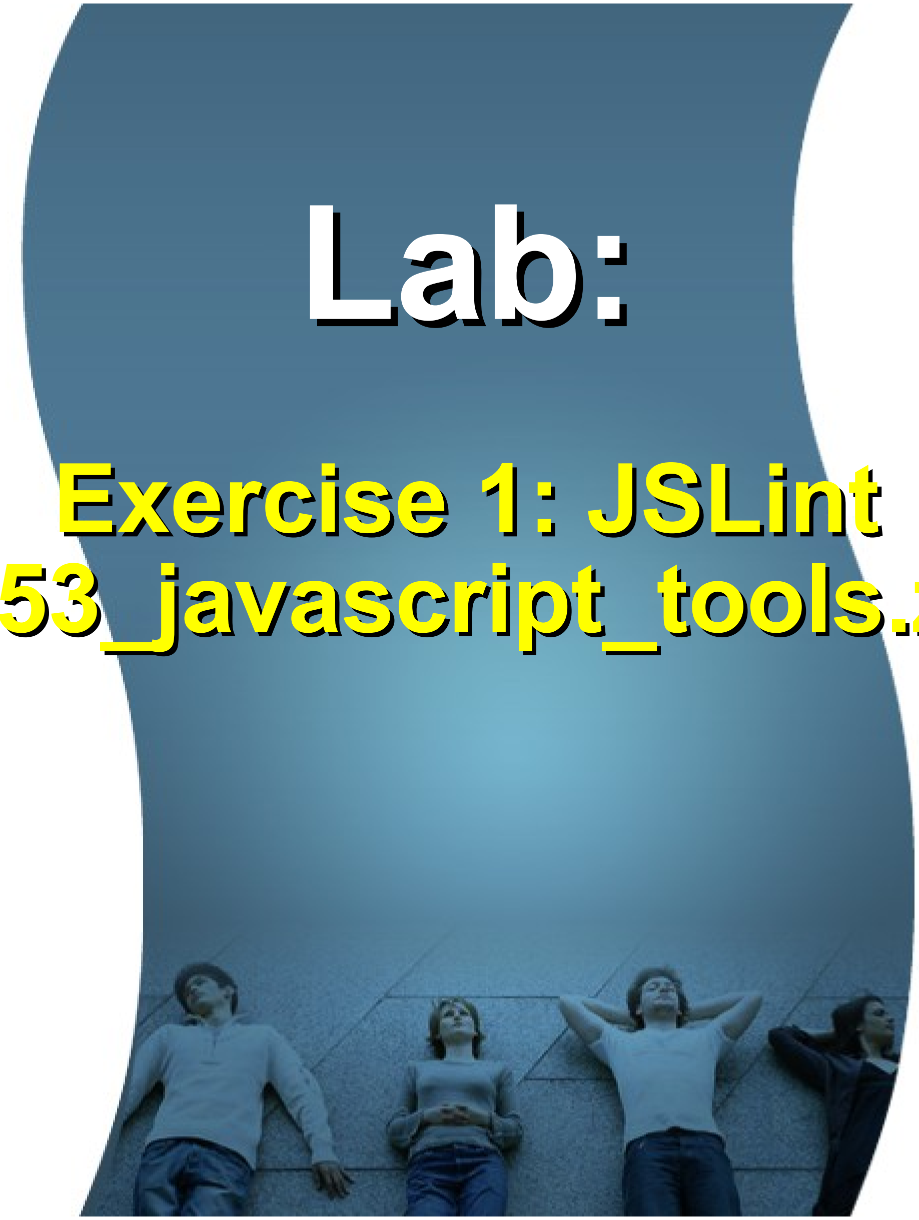
- Issues
  - Constructor functions are functions that are designed to be used with the *new* prefix. The new prefix creates a new object based on the function's prototype, and binds that object to the function's implied this parameter. If you neglect to use the new prefix, no new object will be made and this will be bound to the global object. This is a serious mistake.
- What JSLint expects
  - JSLint enforces the convention that constructor functions be given names with initial uppercase
  - JSLint does not expect to see a function invocation with an initial uppercase name unless it has the new prefix
  - JSLint does not expect to see the new prefix used with functions whose names do not start with initial uppercase.

Properties

- Issues
  - Since JavaScript is a loosely-typed, dynamic-object language, it is not possible to determine at compile time if property names are spelled correctly. JSLint provides some assistance with this.
- What JSLint does
  - At the bottom of its report, JSLint displays a /*properties*/ directive. It contains all of the names and string literals that were used with dot notation, subscript notation, and object literals to name the properties of objects. You can look through the list for misspellings. This is to make misspellings easier to spot.

“use strict”

- What you see when you do not use “use strict” in your function
  - Problem at line 1 character 1: Missing "use strict" statement.
- What is “use strict” for?
  - Strict Mode is a new feature in ECMAScript 5 that allows you to place a program, or a function, in a "strict" operating context. This strict context prevents certain actions from being taken and throws more exceptions
- Strict mode helps out in a couple ways
  - It catches some common coding mistakes, throwing exceptions.
  - It prevents, or throws errors, when relatively "unsafe" actions are taken (such as gaining access to the global object).
  - It disables features that are confusing or poorly thought out



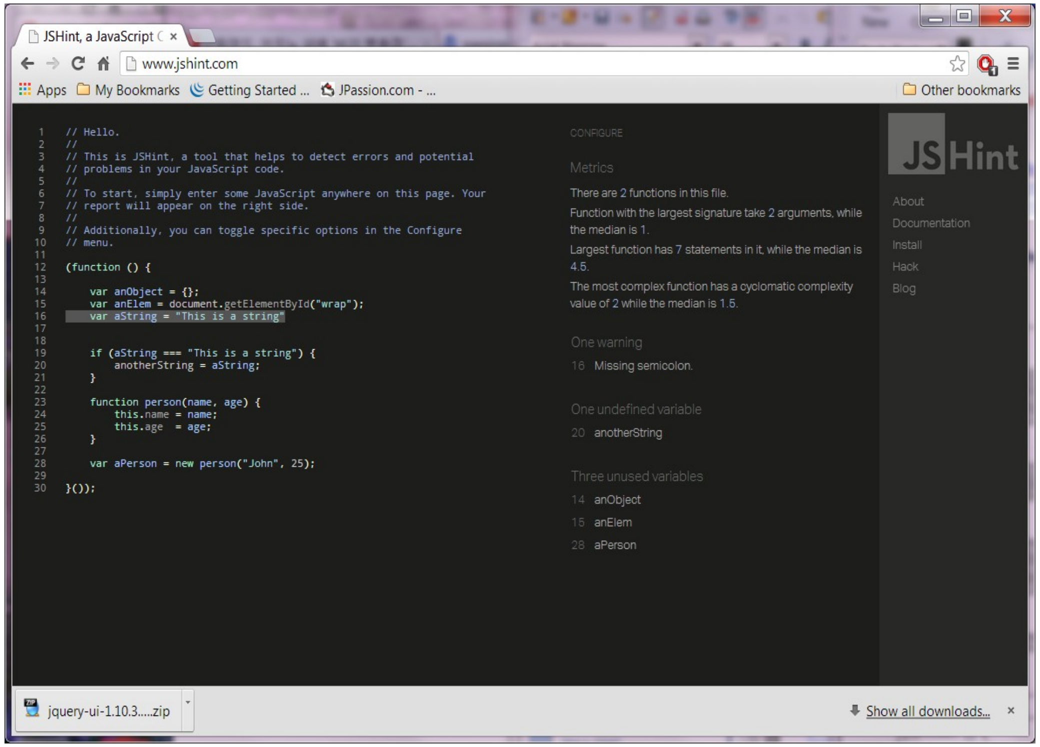
Lab:

Exercise 1: JSLint
4253_javascript_tools.zip

What is JSHint?

What is JSHint?

- http://jshinit.com
- A fork from JSLint – initially created to provide more flexible way of using JSLint (for example, turning off some checks)
- More intuitive UI than JSLint



```
 1  // Hello.
 2  //
 3  // This is JSHint, a tool that helps to detect errors and potential
 4  // problems in your JavaScript code.
 5  //
 6  // To start, simply enter some JavaScript anywhere on this page. Your
 7  // report will appear on the right side.
 8  //
 9  // Additionally, you can toggle specific options in the Configure
10  // menu.
11
12  (function () {
13
14      var anObject = {};
15      var anElem = document.getElementById("wrap");
16      var aString = "This is a string"
17
18
19      if (aString === "This is a string") {
20          anotherString = aString;
21      }
22
23      function person(name, age) {
24          this.name = name;
25          this.age  = age;
26      }
27
28      var aPerson = new person("John", 25);
29
30  }());
```

CONFIGURE

Metrics

There are 2 functions in this file.

Function with the largest signature take 2 arguments, while the median is 1.

Largest function has 7 statements in it, while the median is 4.5.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.5.

One warning

16 Missing semicolon.

One undefined variable

20 anotherString

Three unused variables

14 anObject

15 anElem

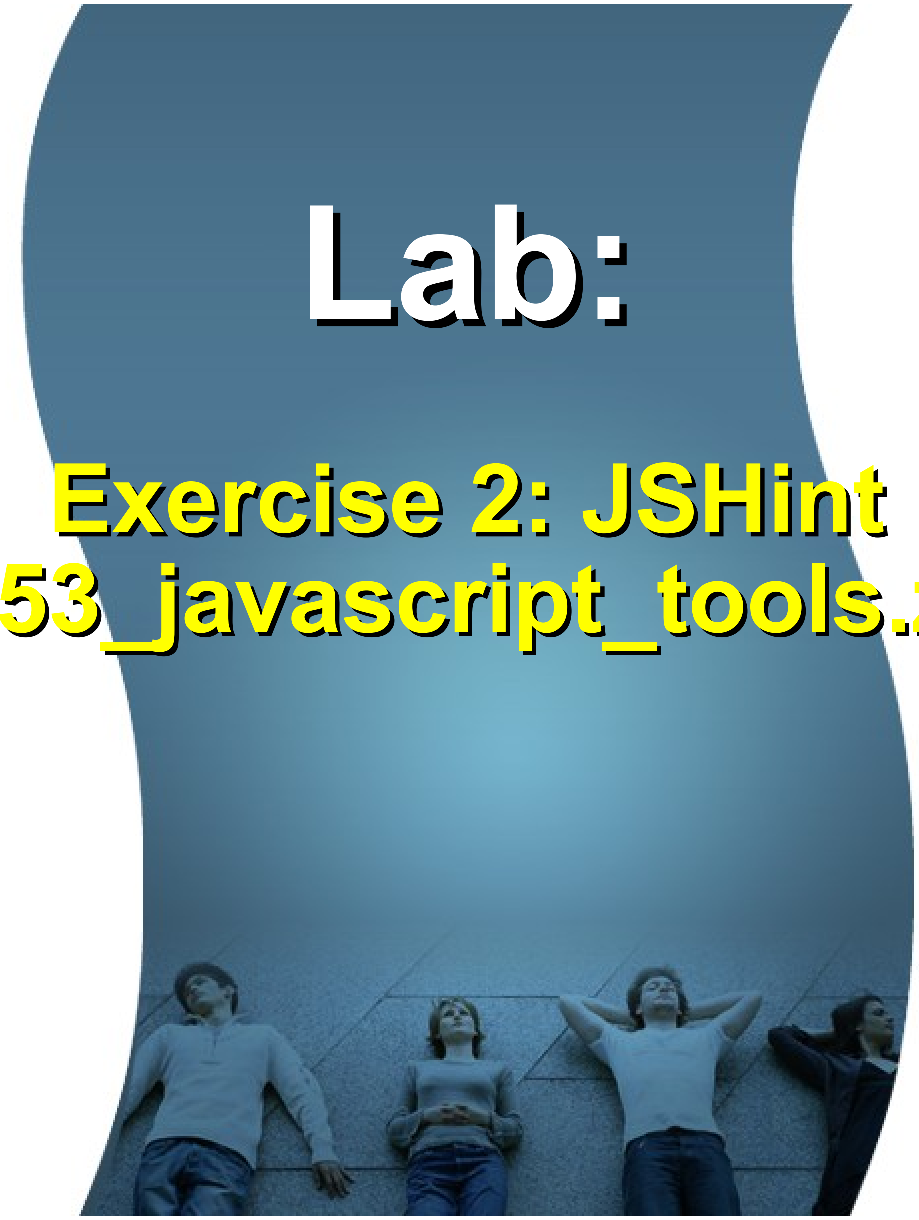
28 aPerson

JSHint Command Line Tool

- Install “jshint” plugin to your Node.js installation
  - npm install jshint -g
- Then run “jslint” command
  - jshint myapp.js

JSHint Plugins for text editors and IDEs

- Editors
  - Sublime Text, TextMate, VIM, Emacs, Nodepad++
- IDE's
  - Visual Studio, Eclipse, IntelliJ, NetBeans
- Build tool
  - Maven



Lab:

Exercise 2: JSHint
4253_javascript_tools.zip

Code with Passion!
JPassion.com