

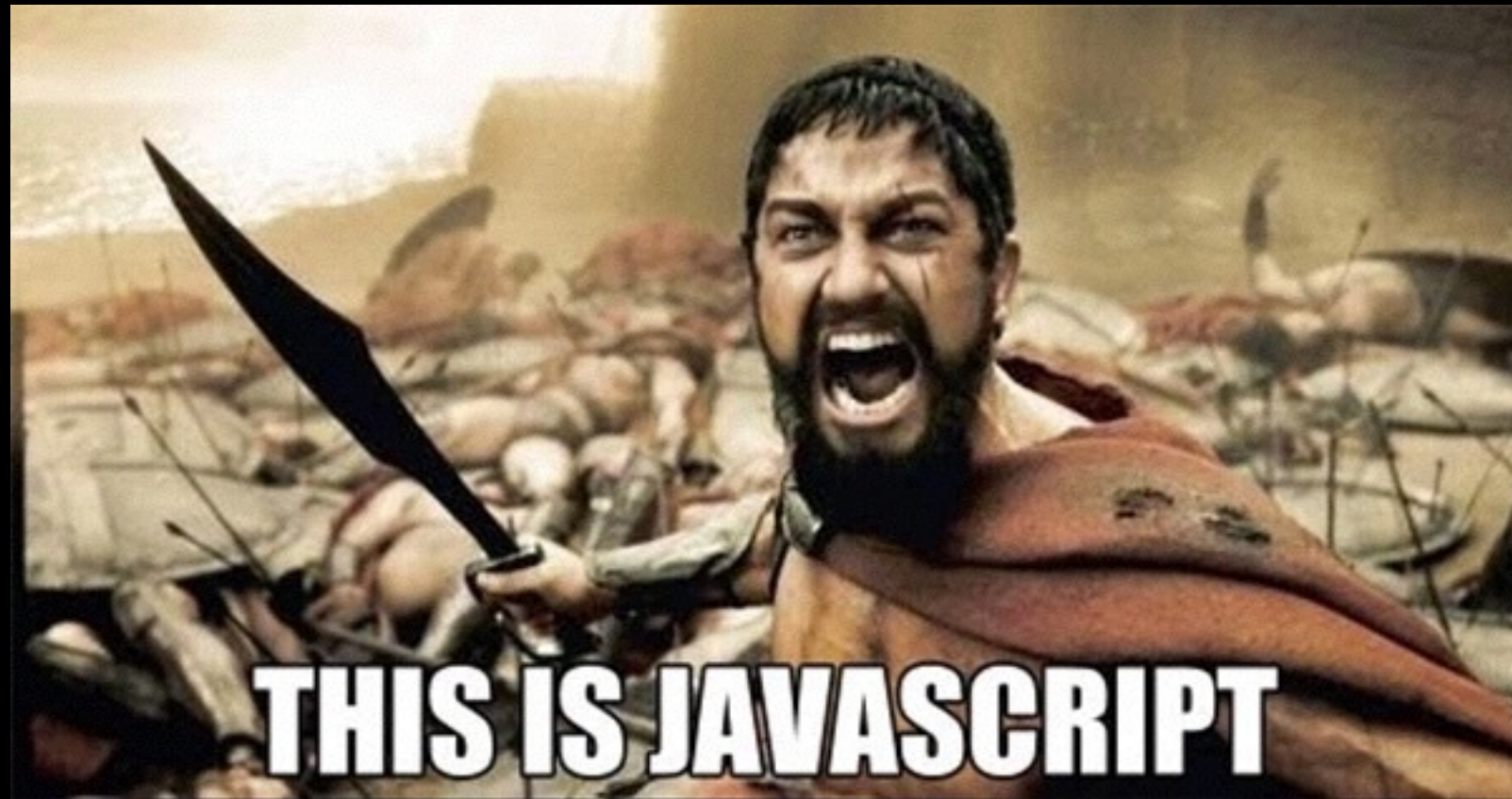
JavaScript Basics



History

- 1995 Famously created in ten days by Brendan Eich for Netscape.
- 2005 Not taken seriously until 2005 when Google used it to write Google Maps.
- 2009 Ryan Dahl writes Node, creating server-side JavaScript.
- 2015 Laurel Schwulst teaches CCA undergrads the basics of JavaScript.

“What language should
I learn?”



Why JS?

Built for front-end interaction so perfect for a graphic designer!

Also used on the server-side so the syntax will be familiar.

100% JavaScript stack!

Ever more frameworks use it:

jQuery, Node, Mongo Db, Angular.js, etc, etc...

JavaScript

The programming language of HTML and the Web. Interaction with the user, animation, etc, all done with JavaScript.

jQuery

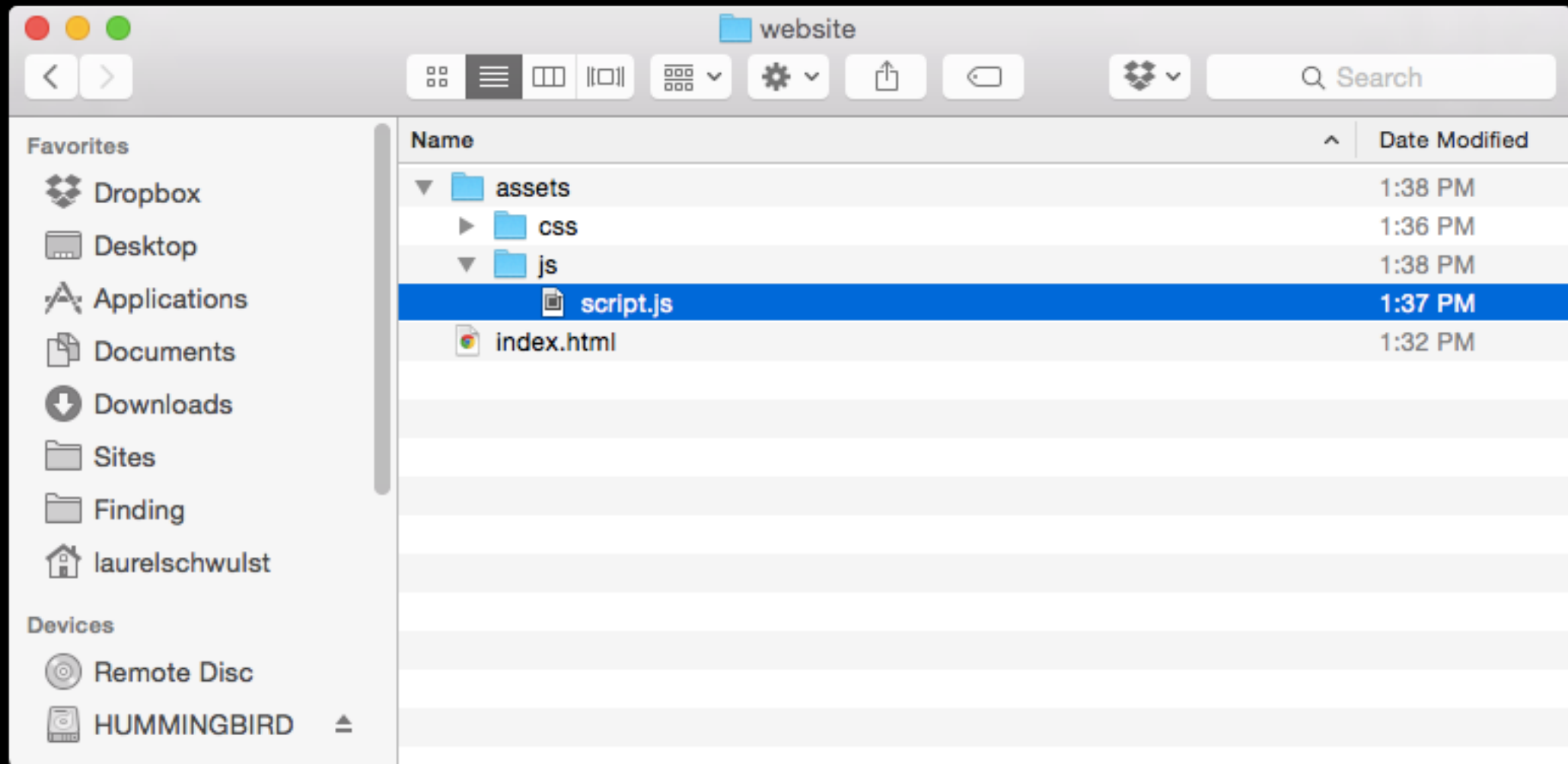
JavaScript library designed to simplify the client-side scripting of HTML.

Load a JS file

Best to load before the closing body tag

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <div></div>
    <script type='text/javascript' src='assets/js/
script.js'></script>
  </body>
</html>
```


Load a JS file



Load a JS file

Order matters!

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <div></div>
    <script type='text/javascript' src='assets/js/
jquery.min.js'></script>
    <script type='text/javascript' src='assets/js/
script.js'></script>
  </body>
</html>
```

Comments in JS

As you know, comments allow you and others keep track of what your code does. The computer ignores it.

You make a single line comment with: //

And a multi-line comment with: / */*

```
// The below function returns all usernames
```

```
/*
```

```
    The below code is used to get  
    the users 10 most recent tweets
```

```
*/
```

Debugging

Coding should be done incrementally.

Debugging is a way for you to check your code as you write it.

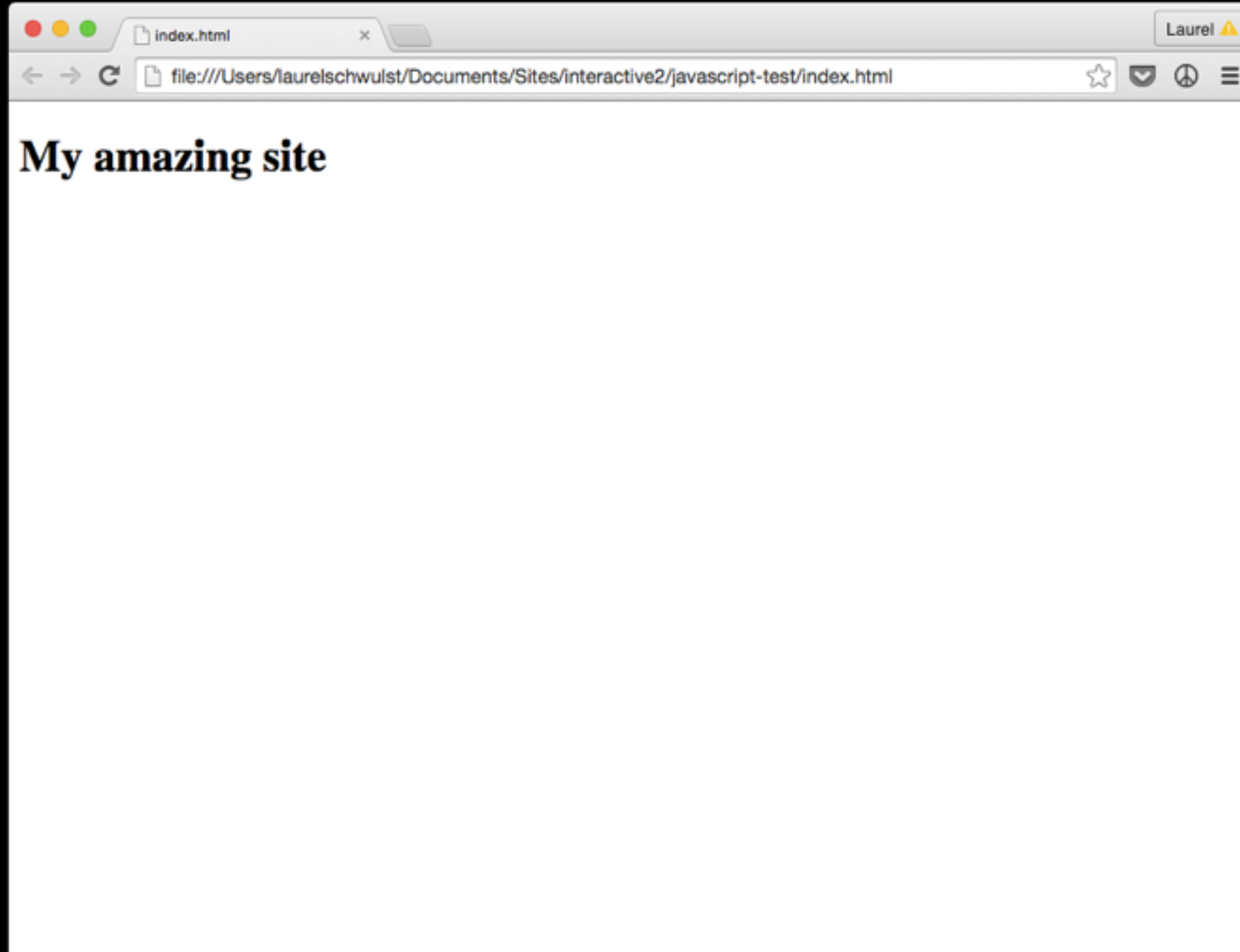
Any time you make a change use `console.log()` to be sure you are getting what you expect.

Debugging

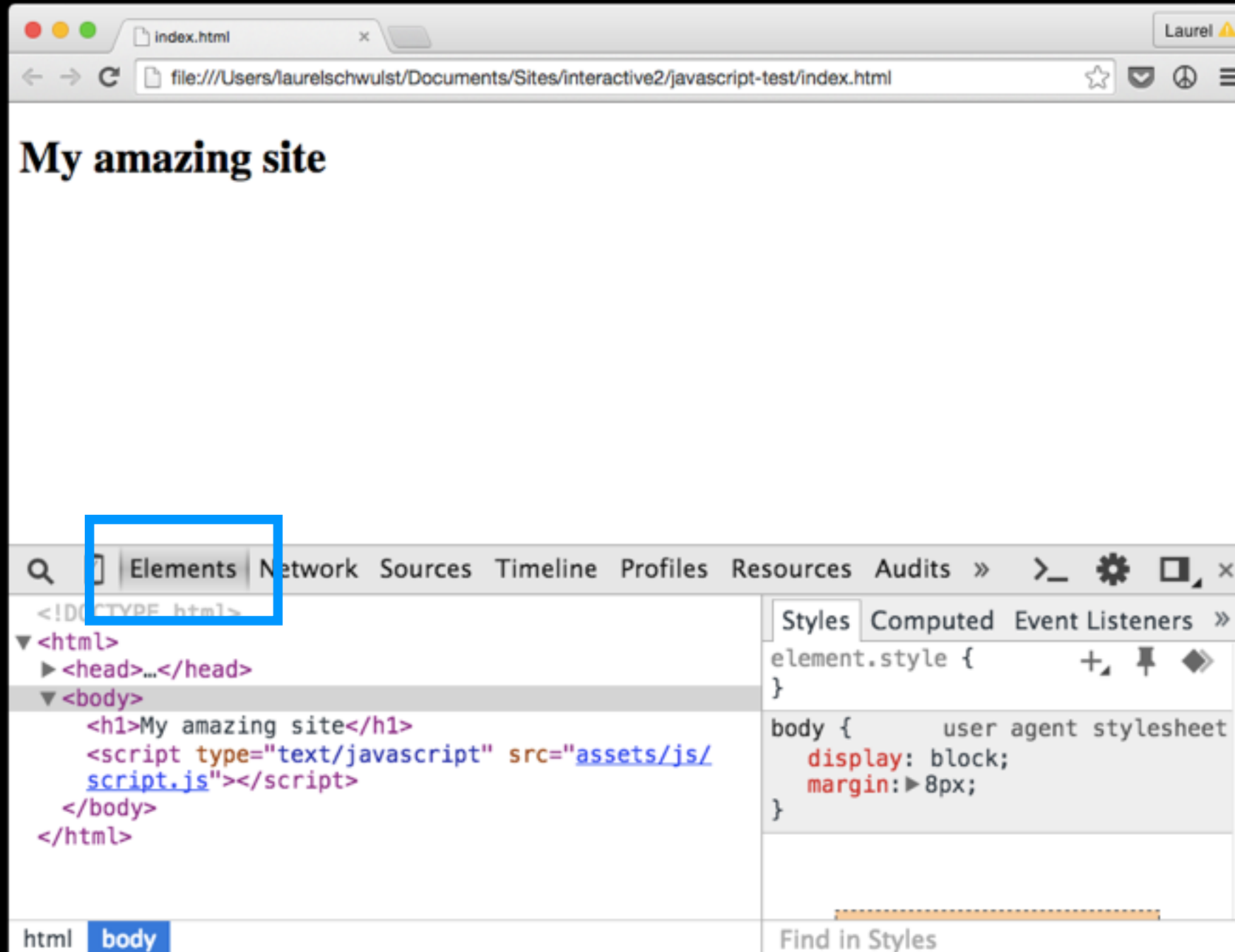
`console.log()` will take whatever is inside the parentheses and log it to the JavaScript console in your browser's developer tools.

Most important line of code I'll show you!

Debugging — Console

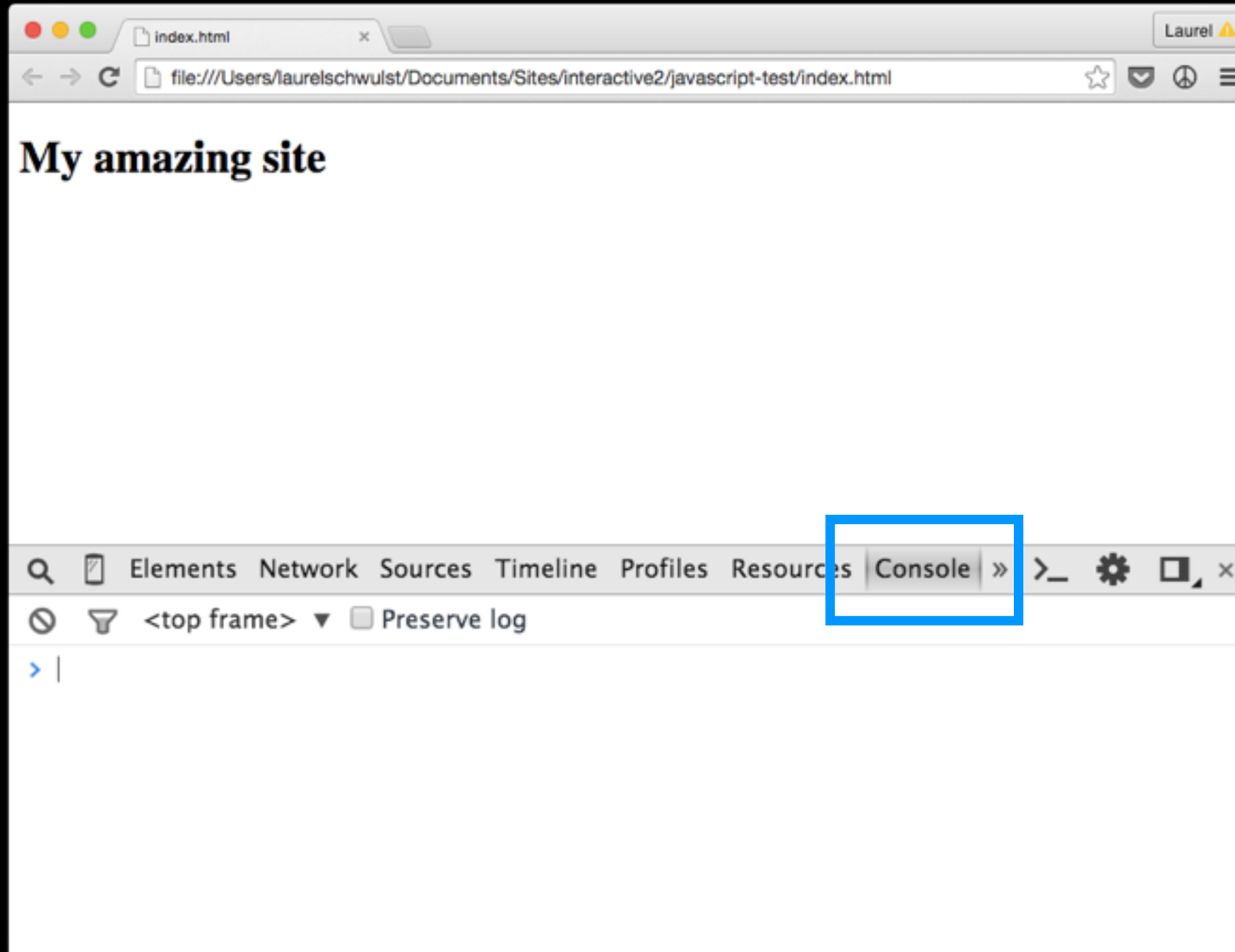


Debugging — Console



This is the part of web inspector you already know — the “Elements” tab...

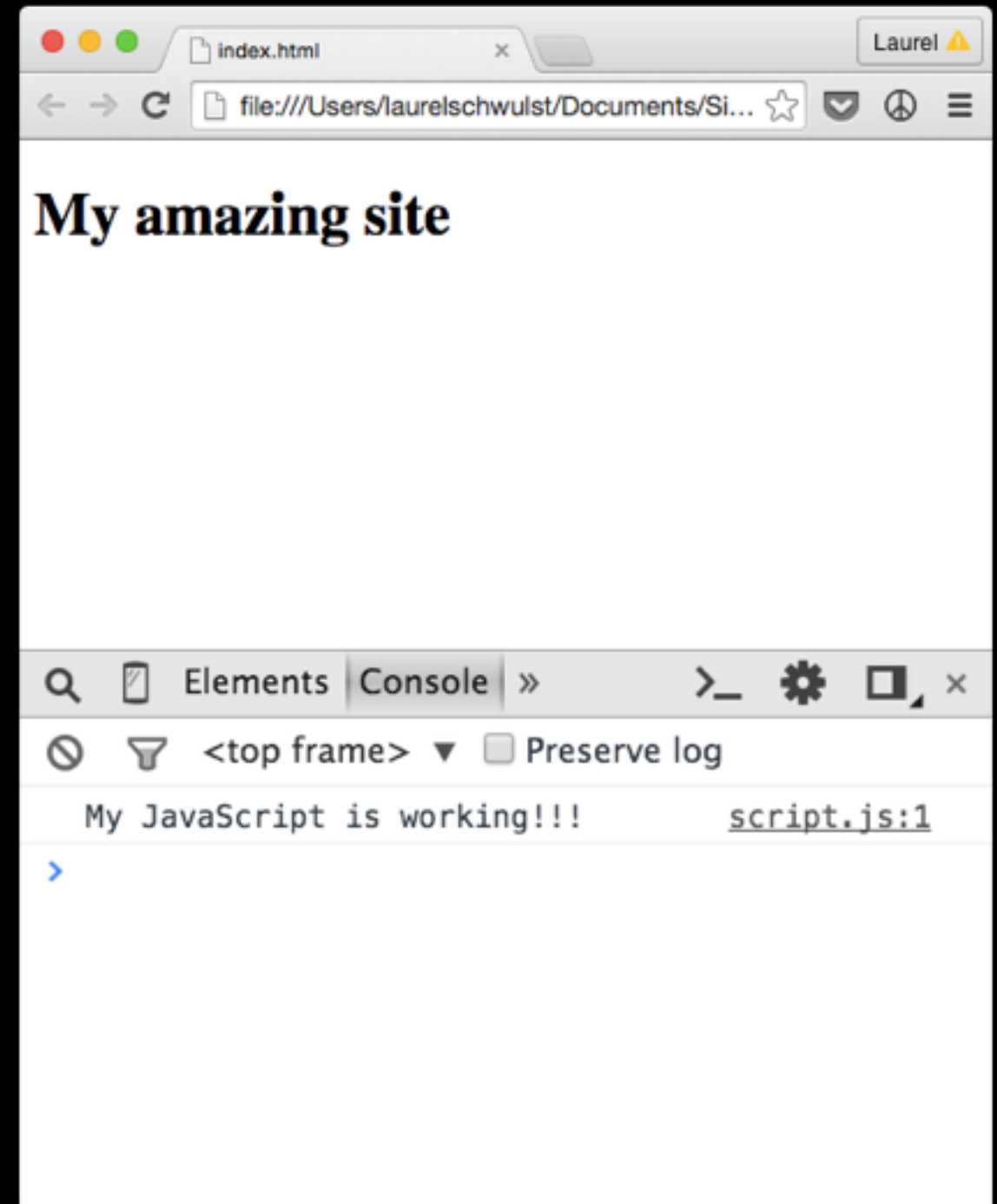
Debugging — Console



To view your “console”, navigate to the Console tab. You can also use the shortcut command + option + J.

Debugging — Example

A screenshot of a code editor window. The title bar shows 'script.js' and 'UNREGISTERED'. The editor has two tabs: 'index.html' and 'script.js'. The 'script.js' tab is active, showing a single line of JavaScript code: `1 console.log("My JavaScript is working!!!");`. The status bar at the bottom indicates 'Line 1, Column 44', 'Tab Size: 4', and 'JavaScript'.



Data Types

These are the most basic types of data the language recognizes:

Integers

Strings

Booleans

Data Types — Integers

Used to represent numerical data.

To make a number in your code, just write a number as numerals without quotes:

```
5 // recent posts to get  
190.12334 // div position from top of browser  
2*50 // new y-position after each animation
```

Data Types — Strings

Used for storing textual information.

To write a string, surround words with quotes:

```
"laurel_schwulst" // username  
"Interactive 2" // course title  
"5" + "7" // makes "57" not 12
```

Data Types — Booleans

Used for representing a binary value (true or false)

Often used in comparisons. Examples:

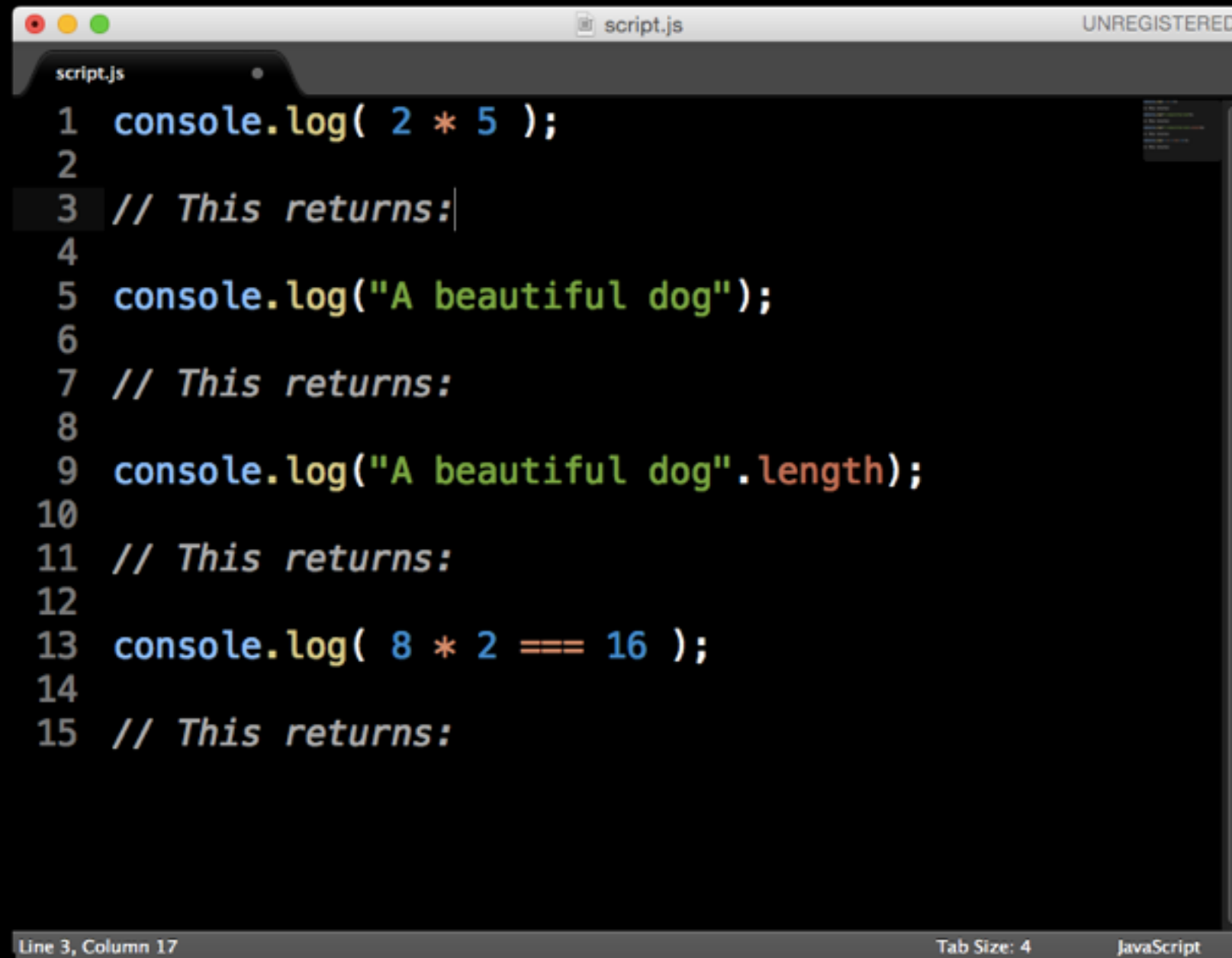
```
23 > 10 // true
```

```
5 < 4 // false
```

```
"abc123" === "abc123" // true
```

```
currentStudent // true
```

Data Types — Example

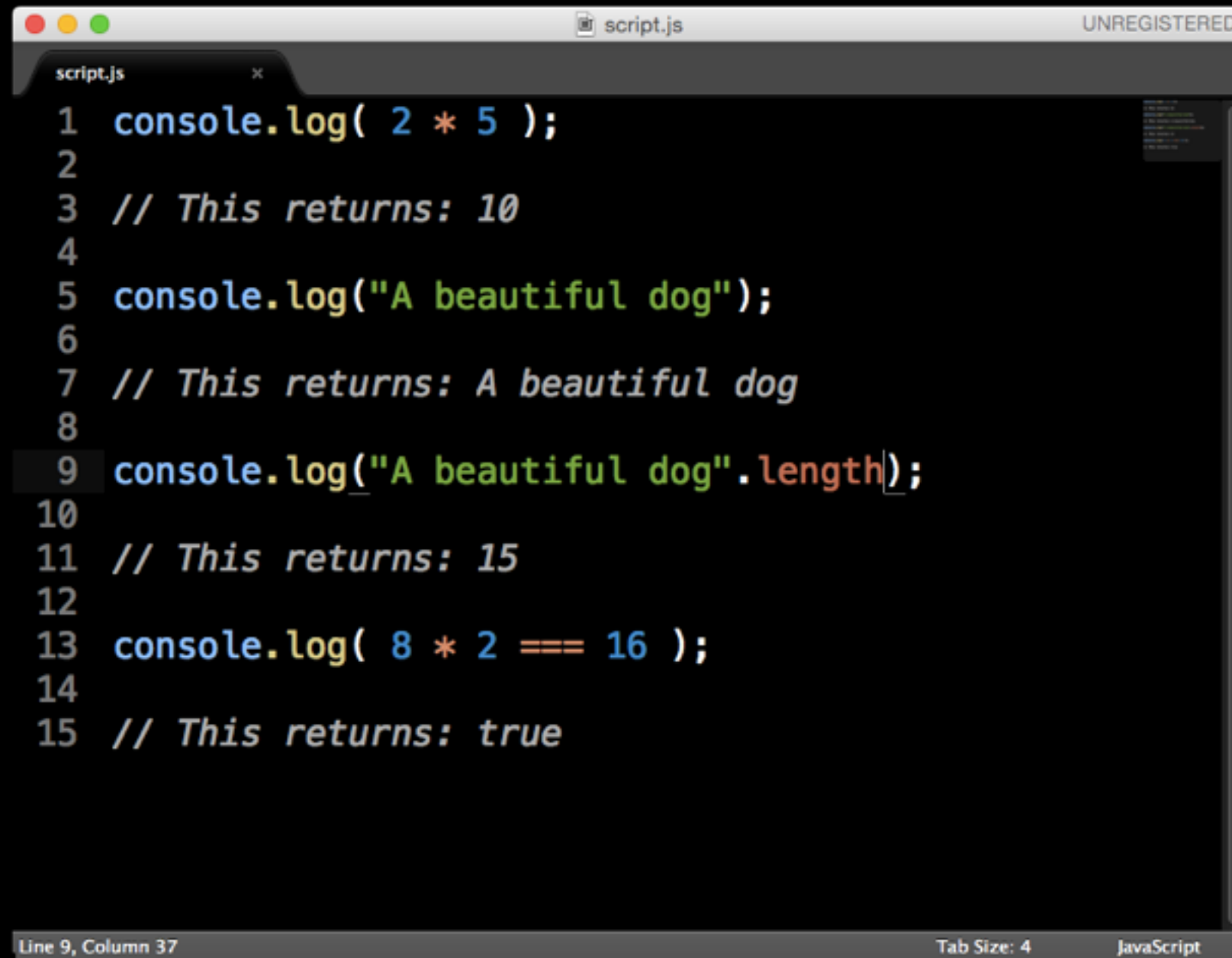


The image shows a code editor window titled 'script.js' with a tab labeled 'script.js' and a status bar indicating 'UNREGISTERED'. The code is as follows:

```
1 console.log( 2 * 5 );  
2  
3 // This returns:  
4  
5 console.log("A beautiful dog");  
6  
7 // This returns:  
8  
9 console.log("A beautiful dog".length);  
10  
11 // This returns:  
12  
13 console.log( 8 * 2 === 16 );  
14  
15 // This returns:
```

The status bar at the bottom shows 'Line 3, Column 17', 'Tab Size: 4', and 'JavaScript'.

Data Types — Example



```
script.js  UNREGISTERED
1 console.log( 2 * 5 );
2
3 // This returns: 10
4
5 console.log("A beautiful dog");
6
7 // This returns: A beautiful dog
8
9 console.log("A beautiful dog".length);
10
11 // This returns: 15
12
13 console.log( 8 * 2 === 16 );
14
15 // This returns: true
```

Line 9, Column 37 Tab Size: 4 JavaScript

Variables

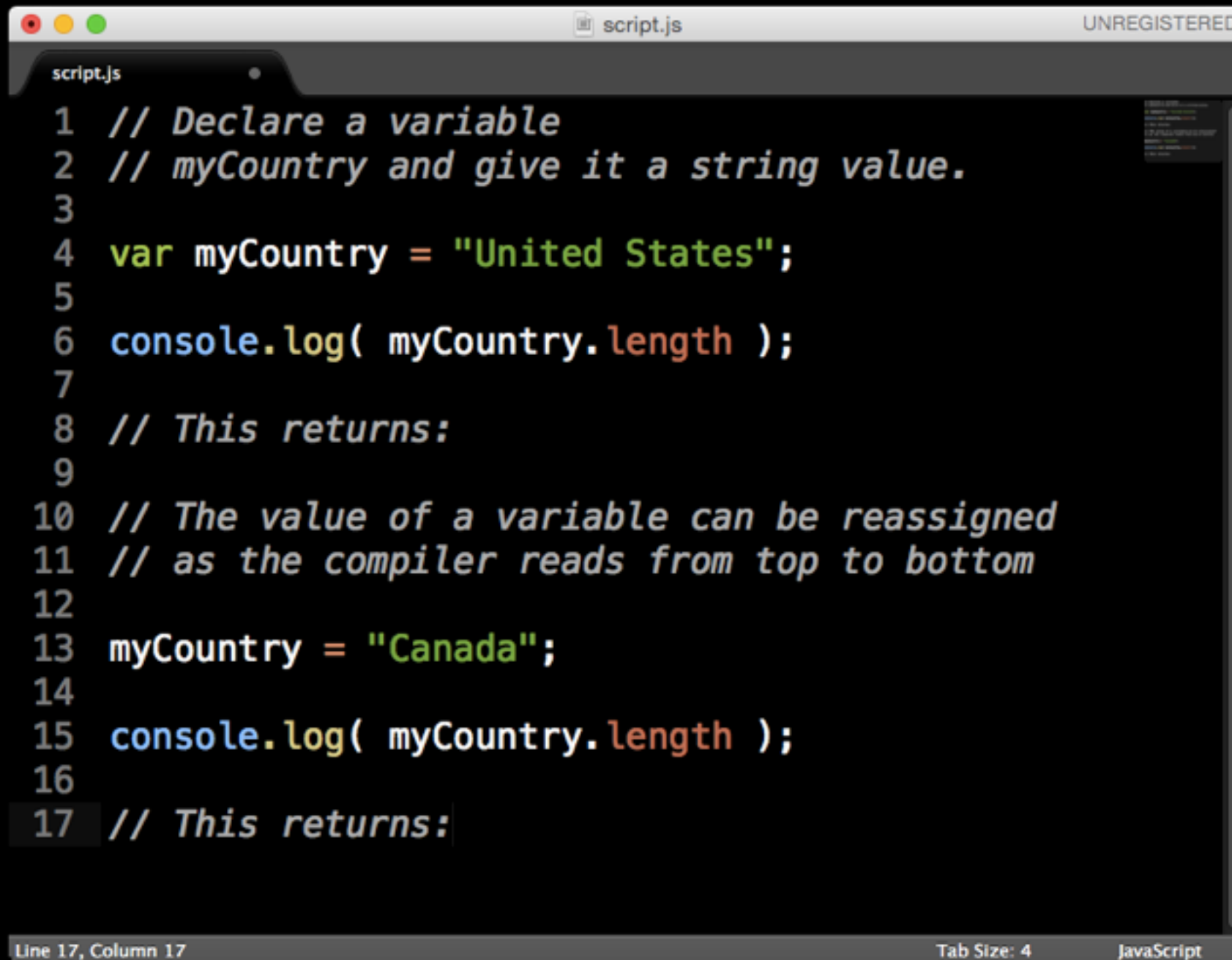
A way to temporarily store values from your coding.

```
var varName = data;
```

Examples with data types:

```
var username = "Laurel";  
var classYear = 2015;  
var currentStudent = true;
```

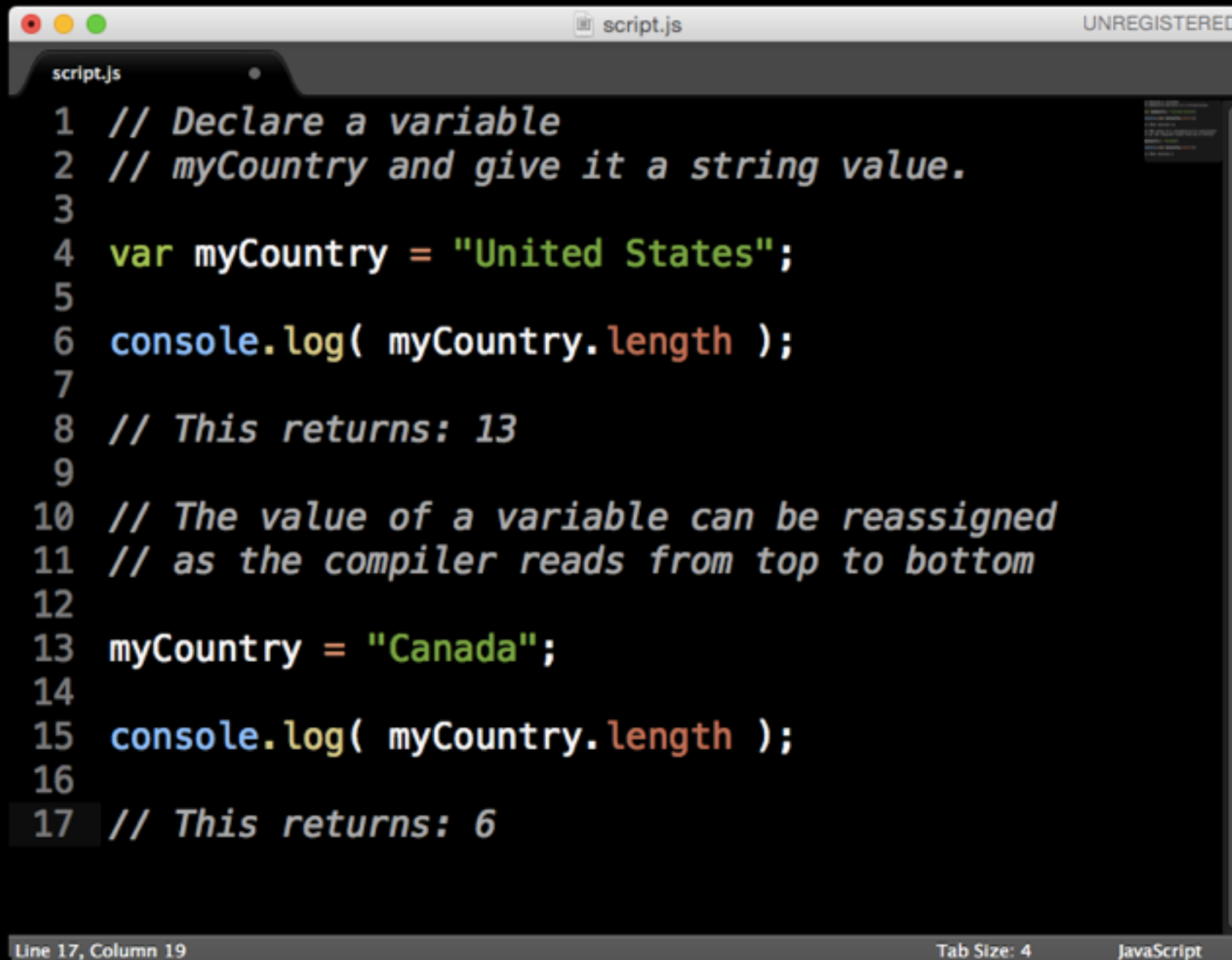

Variables — Example



```
script.js UNREGISTERED
1 // Declare a variable
2 // myCountry and give it a string value.
3
4 var myCountry = "United States";
5
6 console.log( myCountry.length );
7
8 // This returns:
9
10 // The value of a variable can be reassigned
11 // as the compiler reads from top to bottom
12
13 myCountry = "Canada";
14
15 console.log( myCountry.length );
16
17 // This returns:
```

Line 17, Column 17 Tab Size: 4 JavaScript

Variables — Example



```
script.js UNREGISTERED
1 // Declare a variable
2 // myCountry and give it a string value.
3
4 var myCountry = "United States";
5
6 console.log( myCountry.length );
7
8 // This returns: 13
9
10 // The value of a variable can be reassigned
11 // as the compiler reads from top to bottom
12
13 myCountry = "Canada";
14
15 console.log( myCountry.length );
16
17 // This returns: 6
```

Line 17, Column 19 Tab Size: 4 JavaScript

Concatenation

A way to add/connect strings using: +

Example:

```
var username = "Laurel";  
var greeting = "Hi " + username + "!";
```

and now...

Make the computer
make decisions!

Computers think in 1s and 0s.
How can I turn my ideas into a
yes/no logic?

Should a div be visible on load?

Did the user click a button?

Did the user enter the right
password?

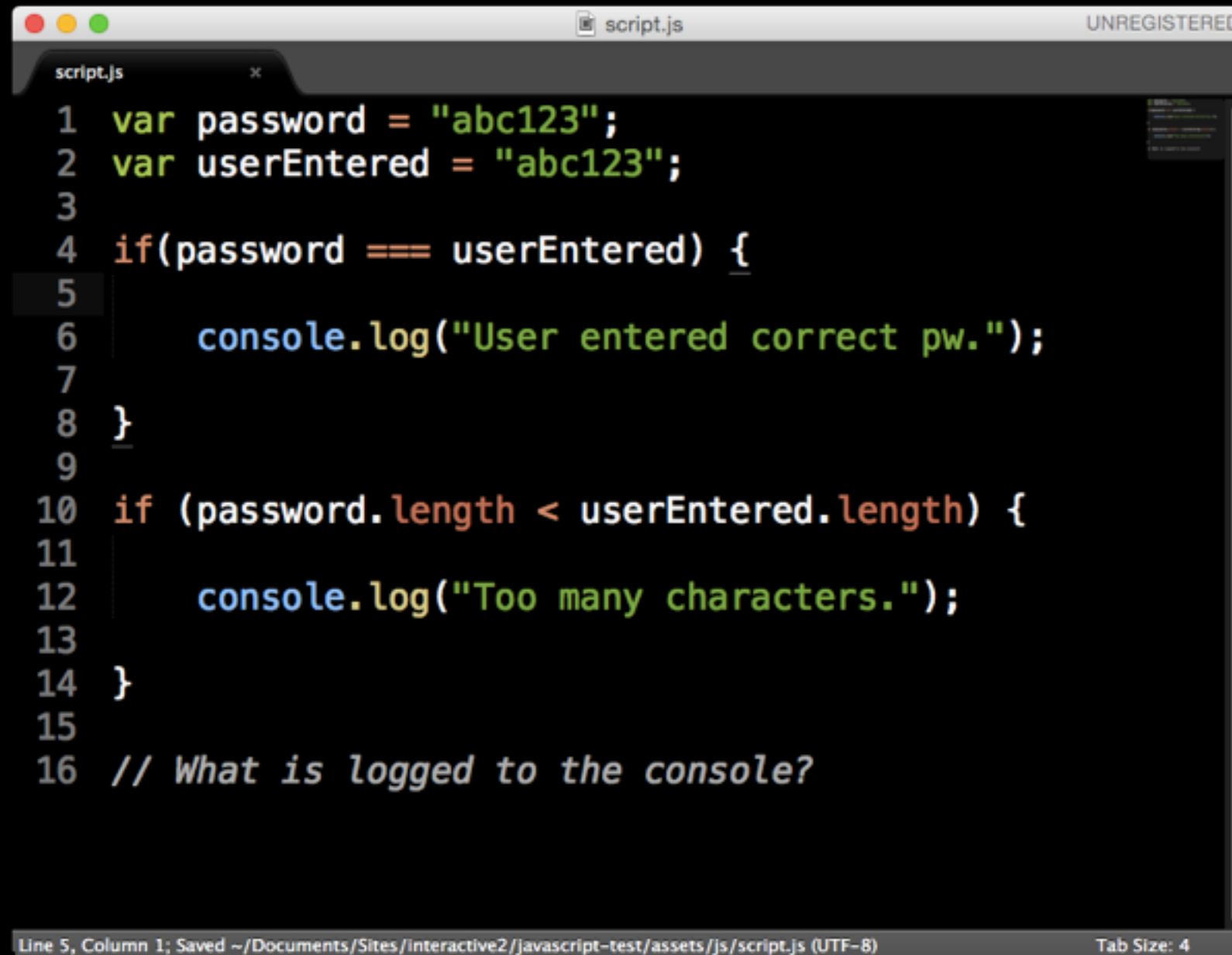
If Statement — basic yes

Made up of the `if` keyword, a `condition`, and a pair of curly braces `{ }`. If the answer to the condition is yes, the code inside the curly braces will run.

Syntax:

```
if ( condition ) {  
    // if the condition returns true then  
    // execute code inside these brackets.  
    // if false, skip this code.  
}
```

If Statement — Example

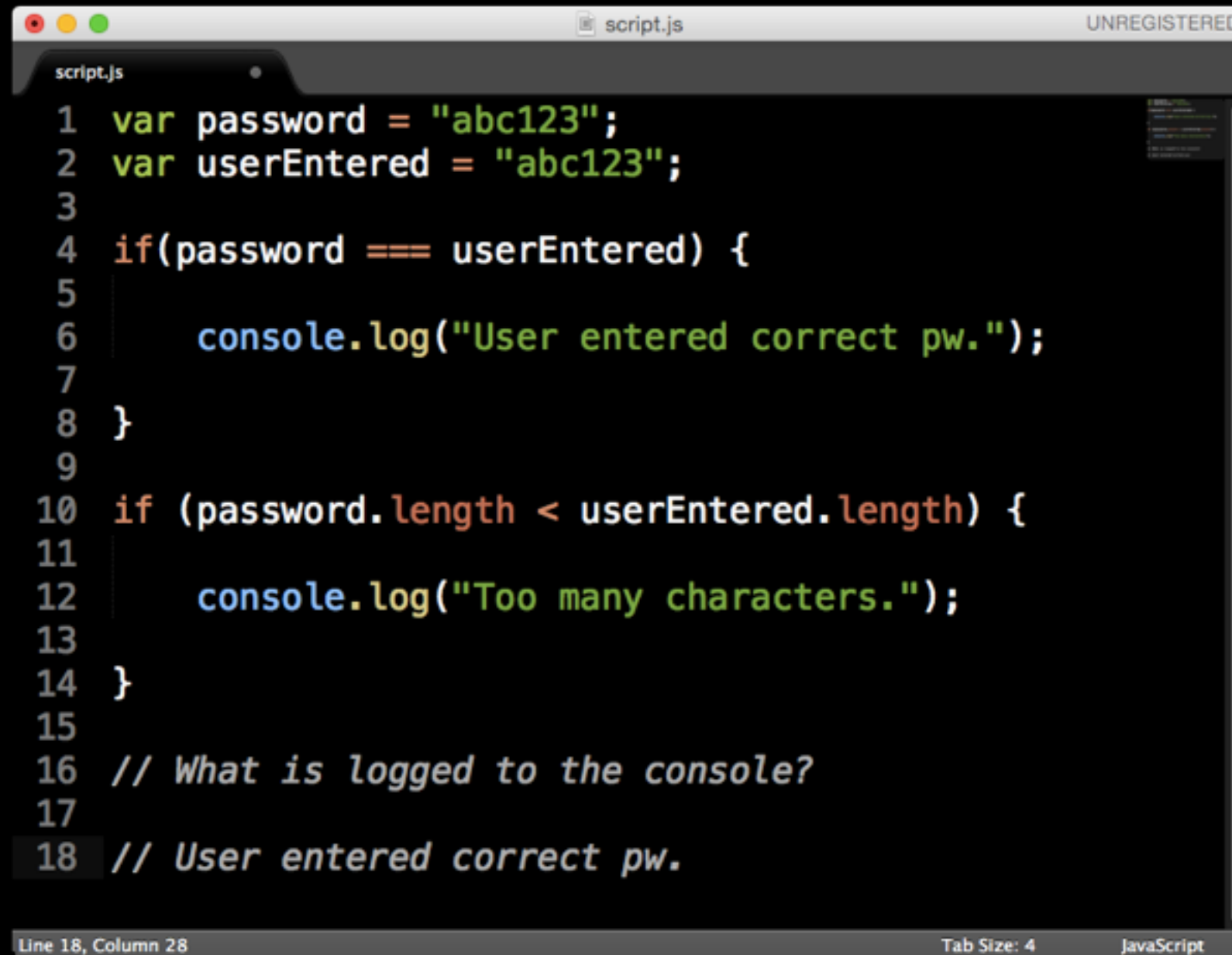


A screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The editor contains the following JavaScript code:

```
1 var password = "abc123";
2 var userEntered = "abc123";
3
4 if(password === userEntered) {
5     console.log("User entered correct pw.");
6 }
7
8 if (password.length < userEntered.length) {
9     console.log("Too many characters.");
10 }
11
12 // What is logged to the console?
```

The status bar at the bottom shows 'Line 5, Column 1; Saved ~/Documents/Sites/Interactive2/javascript-test/assets/js/script.js (UTF-8)' and 'Tab Size: 4'.

If Statement — Example

A screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The code is written in JavaScript and uses syntax highlighting. It defines two variables, 'password' and 'userEntered', both set to 'abc123'. The first if statement checks for equality and logs a success message. The second if statement checks if the password is shorter than the user input and logs an error message. The editor shows line numbers from 1 to 18, and the status bar at the bottom indicates 'Line 18, Column 28', 'Tab Size: 4', and 'JavaScript'.

```
1 var password = "abc123";
2 var userEntered = "abc123";
3
4 if(password === userEntered) {
5
6     console.log("User entered correct pw.");
7
8 }
9
10 if (password.length < userEntered.length) {
11
12     console.log("Too many characters.");
13
14 }
15
16 // What is logged to the console?
17
18 // User entered correct pw.
```

If Statement — basic yes/no

In addition to doing something when the condition is true, we can do something else if the condition is false.

Syntax:

```
if ( currentStudent === true ) {  
    console.log("You are currently enrolled");  
}  
  
else {  
    console.log("You are not enrolled");  
}
```

Logical operands

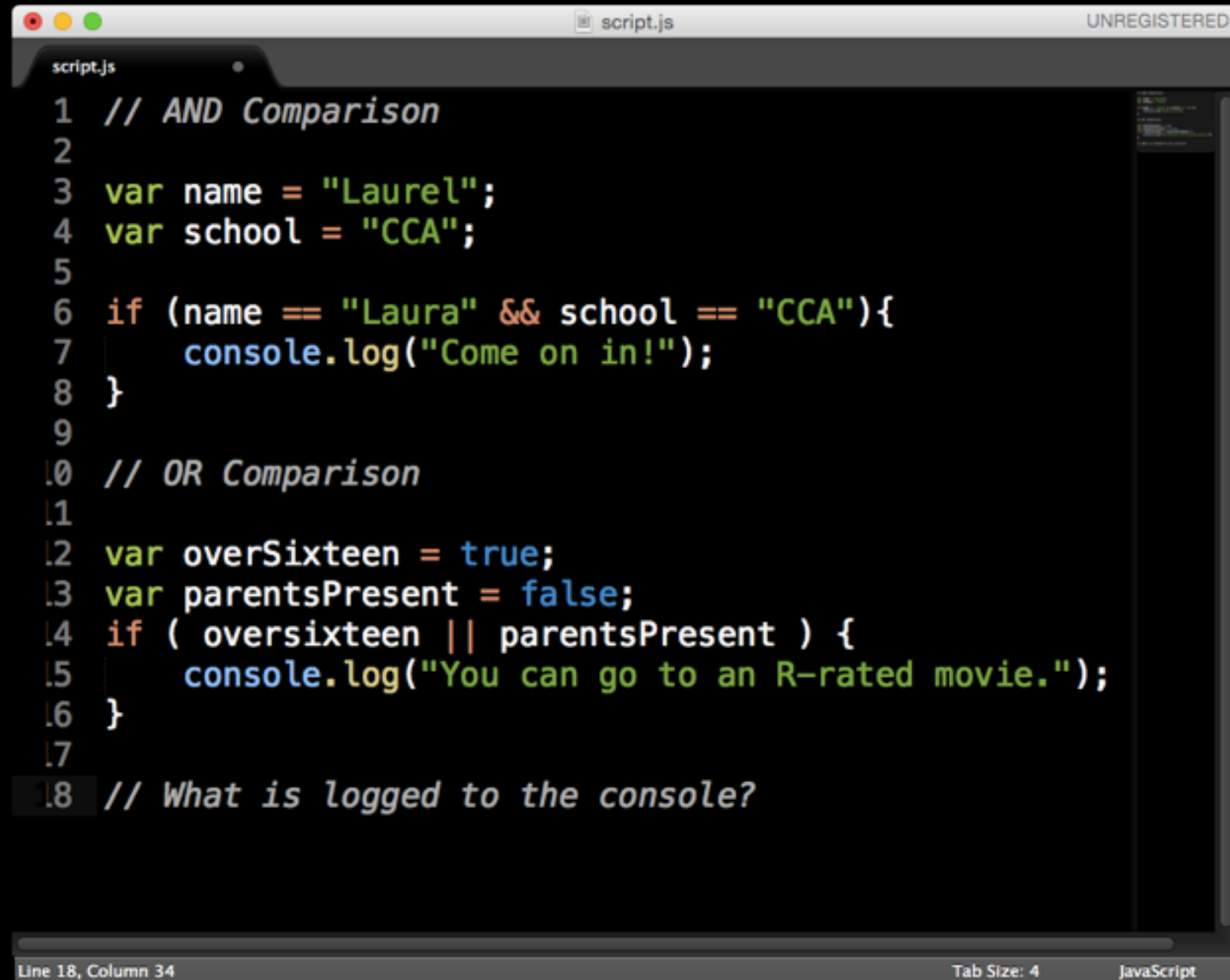
Used when several conditions need to be evaluated at once.

&& (and)

|| (or)

!= (not equal)

Logical operands — Example

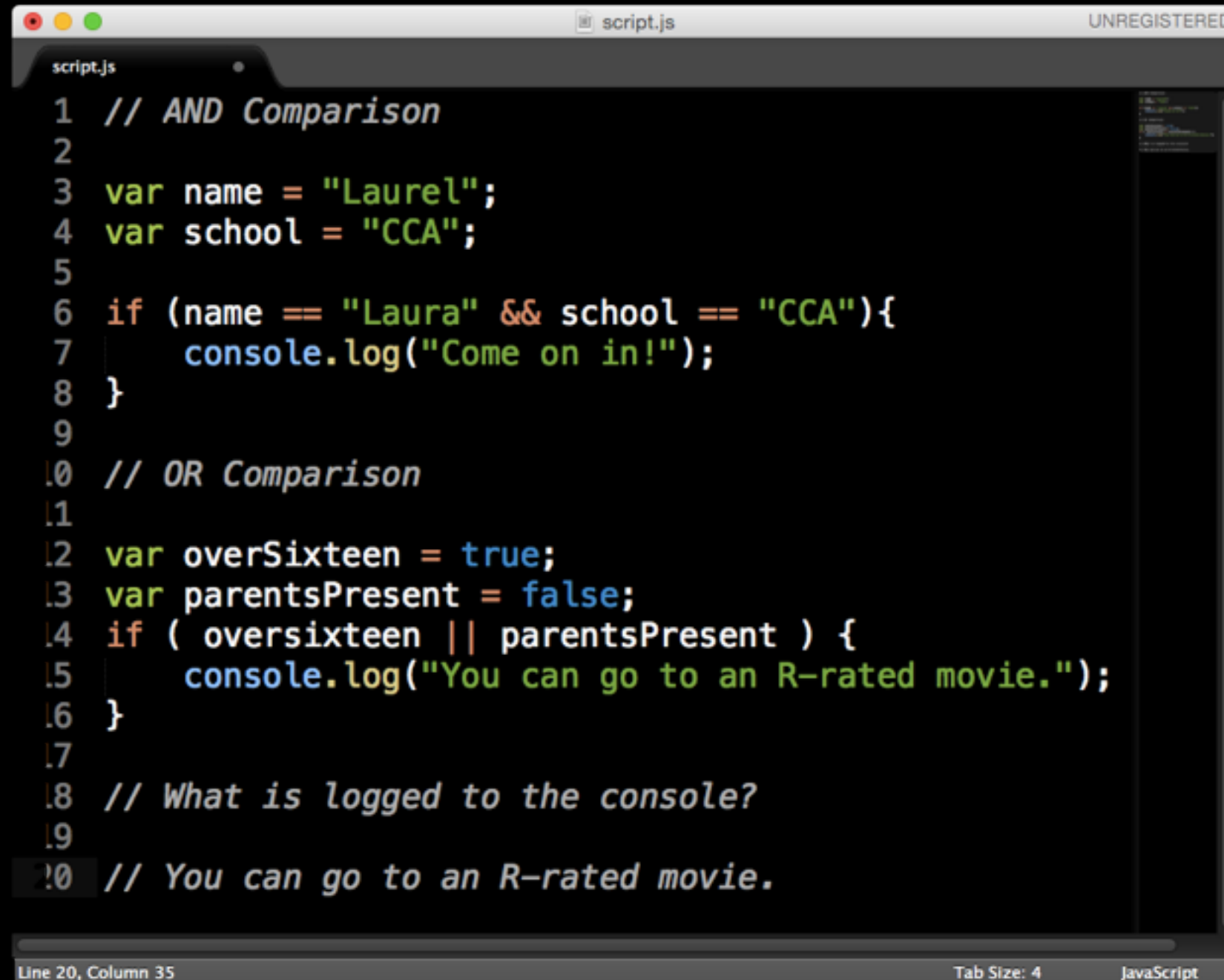


The image shows a screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The code is written in JavaScript and demonstrates the use of logical operands. It is divided into three sections: an AND comparison, an OR comparison, and a final comment. The AND comparison section (lines 1-9) declares variables 'name' and 'school', and uses an 'if' statement with the logical AND operator '&&' to check if 'name' is 'Laura' and 'school' is 'CCA'. The OR comparison section (lines 10-17) declares variables 'overSixteen' and 'parentsPresent', and uses an 'if' statement with the logical OR operator '||' to check if 'overSixteen' is true or 'parentsPresent' is false. The final line (line 18) is a comment asking what is logged to the console.

```
1 // AND Comparison
2
3 var name = "Laurel";
4 var school = "CCA";
5
6 if (name == "Laura" && school == "CCA"){
7     console.log("Come on in!");
8 }
9
10 // OR Comparison
11
12 var overSixteen = true;
13 var parentsPresent = false;
14 if ( overSixteen || parentsPresent ) {
15     console.log("You can go to an R-rated movie.");
16 }
17
18 // What is logged to the console?
```

Line 18, Column 34 Tab Size: 4 JavaScript

Logical operands — Example



The image shows a screenshot of a code editor window titled "script.js" with a status bar indicating "UNREGISTERED". The code is written in JavaScript and demonstrates the use of logical operands. It is divided into two sections: "AND Comparison" and "OR Comparison".

```
1 // AND Comparison
2
3 var name = "Laurel";
4 var school = "CCA";
5
6 if (name == "Laura" && school == "CCA"){
7     console.log("Come on in!");
8 }
9
10 // OR Comparison
11
12 var overSixteen = true;
13 var parentsPresent = false;
14 if ( overSixteen || parentsPresent ) {
15     console.log("You can go to an R-rated movie.");
16 }
17
18 // What is logged to the console?
19
20 // You can go to an R-rated movie.
```

The status bar at the bottom indicates "Line 20, Column 35", "Tab Size: 4", and "JavaScript".

Arrays

So far, we've only been able to store *one* number or *one* string. Arrays are used to store a set of related things.

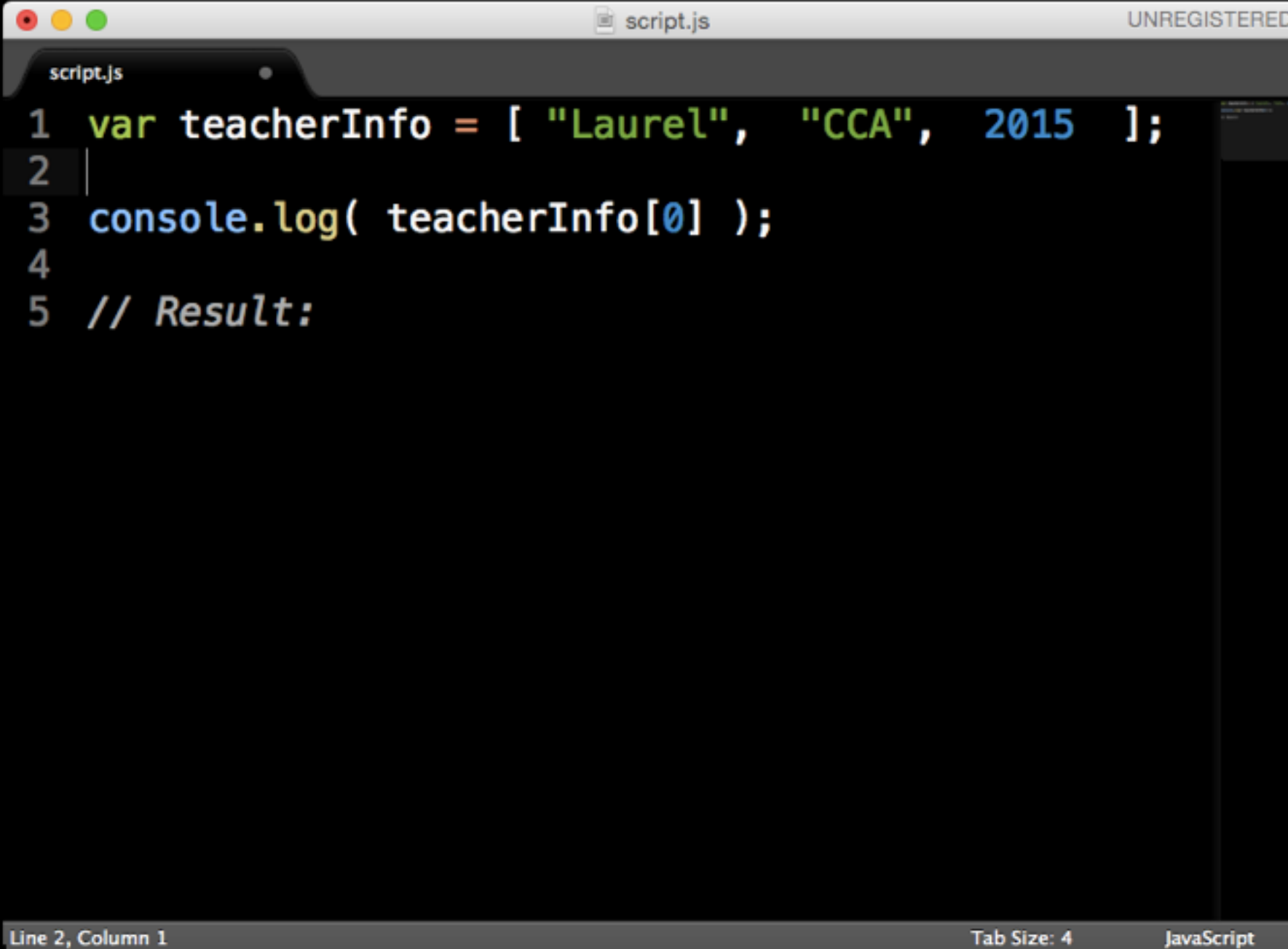
```
var arrayName = [data, data, data];  
var teacherInfo = ["Laurel", "CCA", 2015];
```

Arrays — access elements

```
var teacherInfo = [ "Laurel", "CCA", 2015 ] ;
```

0 **1** **2**

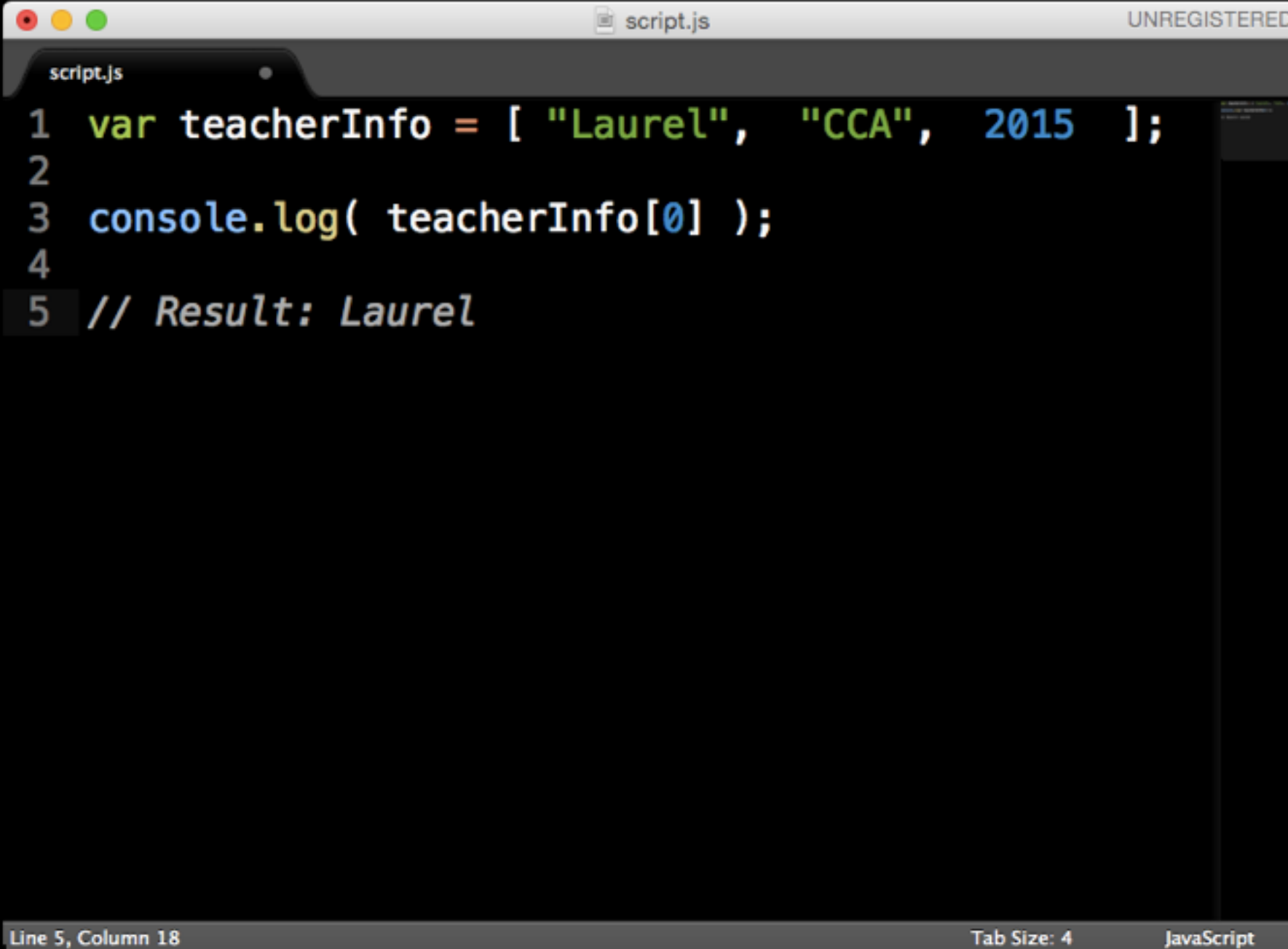
Arrays — Example



```
script.js UNREGISTERED
1 var teacherInfo = [ "Laurel", "CCA", 2015 ];
2
3 console.log( teacherInfo[0] );
4
5 // Result:
```

Line 2, Column 1 Tab Size: 4 JavaScript

Arrays — Example



```
script.js UNREGISTERED
1 var teacherInfo = [ "Laurel", "CCA", 2015 ];
2
3 console.log( teacherInfo[0] );
4
5 // Result: Laurel
```

Line 5, Column 18 Tab Size: 4 JavaScript

The image shows a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The code defines an array 'teacherInfo' with three elements: 'Laurel', 'CCA', and 2015. It then logs the first element, 'Laurel', to the console. A comment on the fifth line indicates the result of the log statement. The status bar at the bottom shows 'Line 5, Column 18', 'Tab Size: 4', and 'JavaScript'.

For loop

Repeat a block of code a set number of times. Most often used to access elements of an array one by one.

```
for (counter; condition; increment) {  
    // code to execute  
}
```

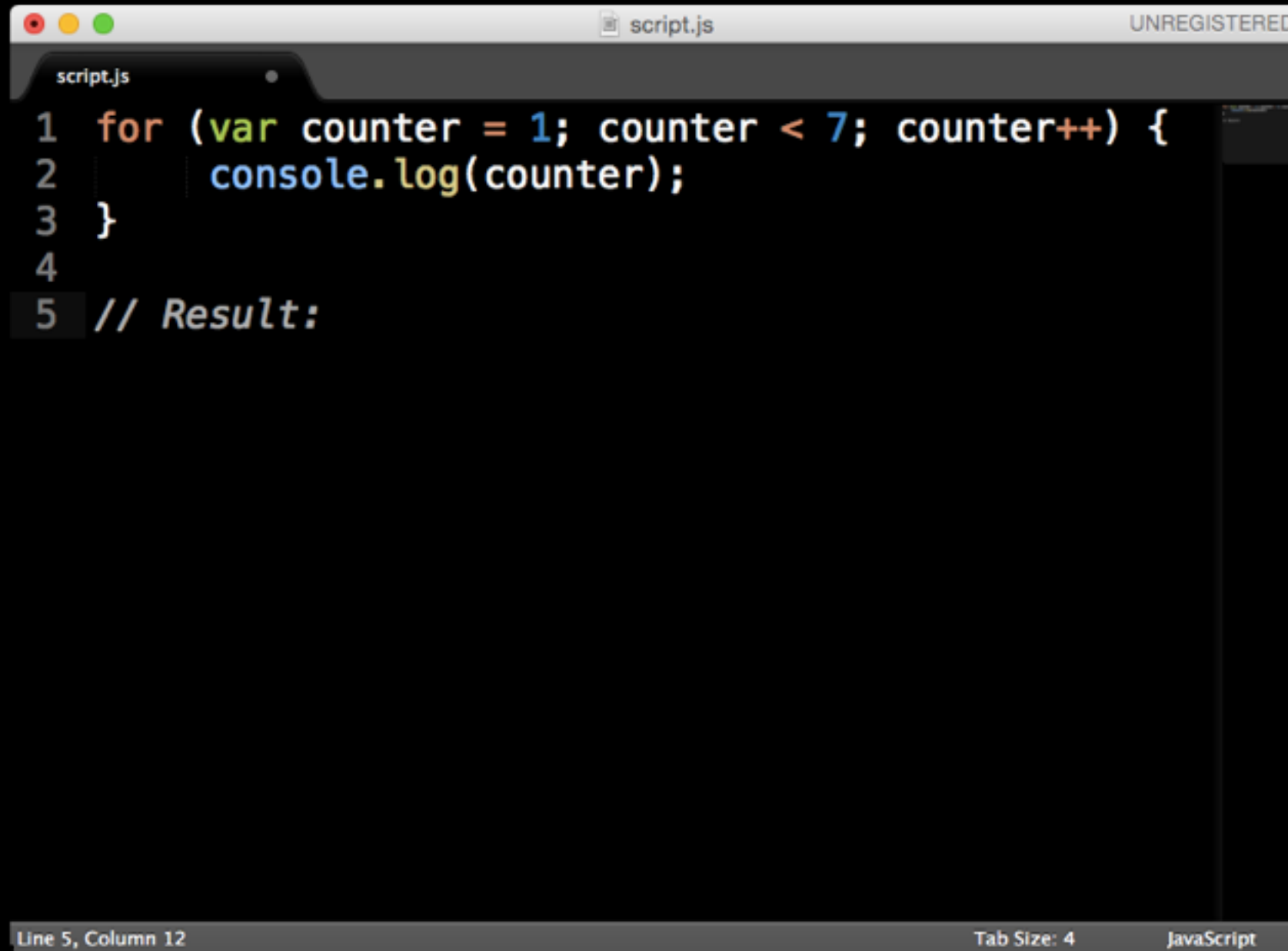
For loop

Repeat a block of code a set number of times. Most often used to access elements of an array one by one.

```
for (counter; condition; increment) {  
    // code to execute  
}
```

**Decides how many times
the loop will run**

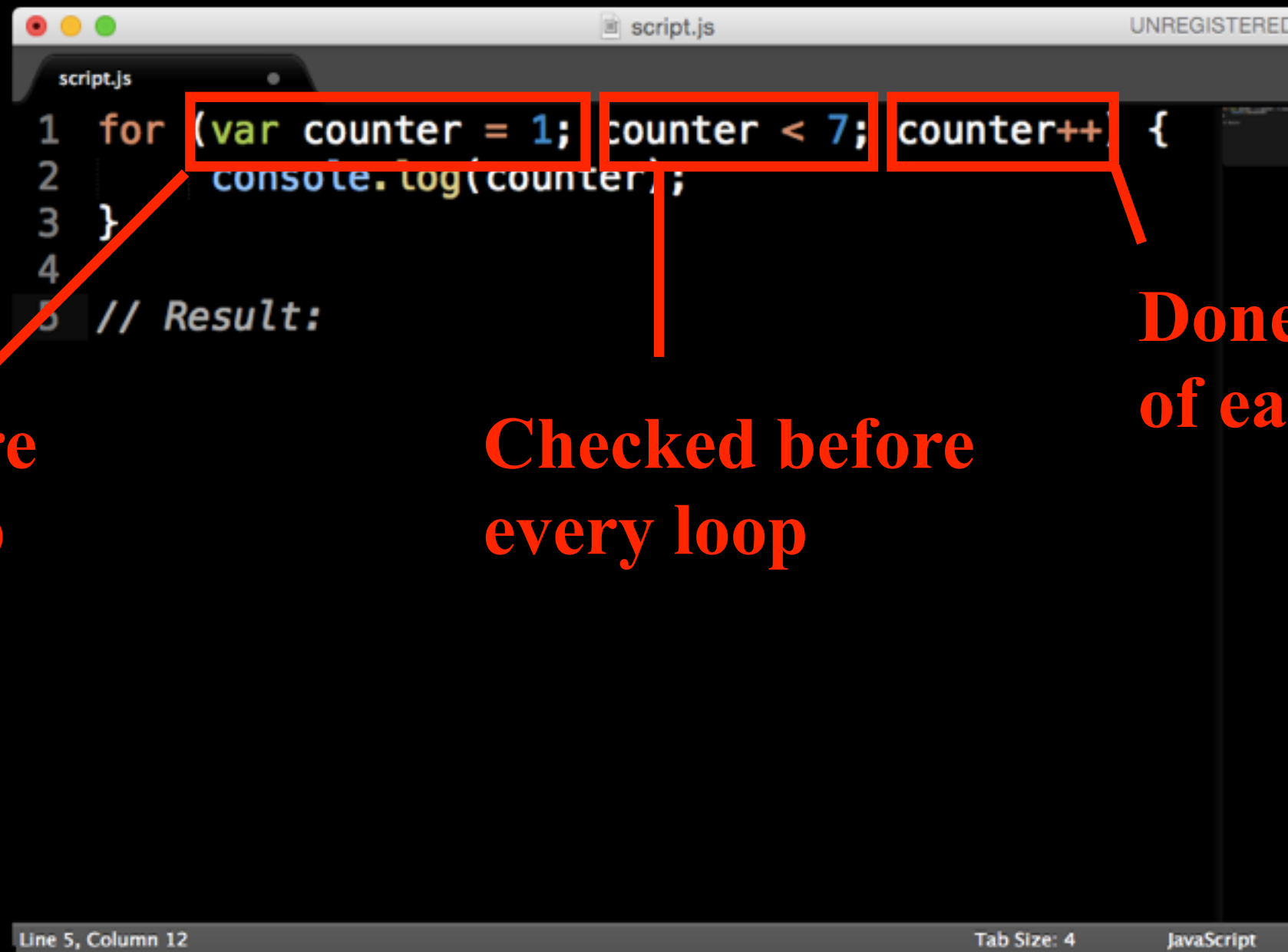
For loop — Example

A screenshot of a code editor window titled 'script.js' with 'UNREGISTERED' in the top right corner. The editor contains five lines of JavaScript code. Line 1: '1 for (var counter = 1; counter < 7; counter++) {'; Line 2: '2 console.log(counter);'; Line 3: '3 }'; Line 4: '4'; Line 5: '5 // Result:'. The code is syntax-highlighted. The status bar at the bottom shows 'Line 5, Column 12', 'Tab Size: 4', and 'JavaScript'.

```
1 for (var counter = 1; counter < 7; counter++) {  
2     console.log(counter);  
3 }  
4  
5 // Result:
```

Line 5, Column 12 Tab Size: 4 JavaScript

For loop — Example



The screenshot shows a code editor window titled 'script.js' with a 'UNREGISTERED' label. The code is as follows:

```
1 for (var counter = 1; counter < 7; counter++) {  
2   console.log(counter);  
3 }  
4  
5 // Result:
```

Three red boxes highlight the components of the for loop: 'var counter = 1;', 'counter < 7;', and 'counter++'. Red lines connect these boxes to explanatory text:

- A line from 'var counter = 1;' points to the text 'Set before first loop'.
- A line from 'counter < 7;' points to the text 'Checked before every loop'.
- A line from 'counter++' points to the text 'Done at the end of each loop'.

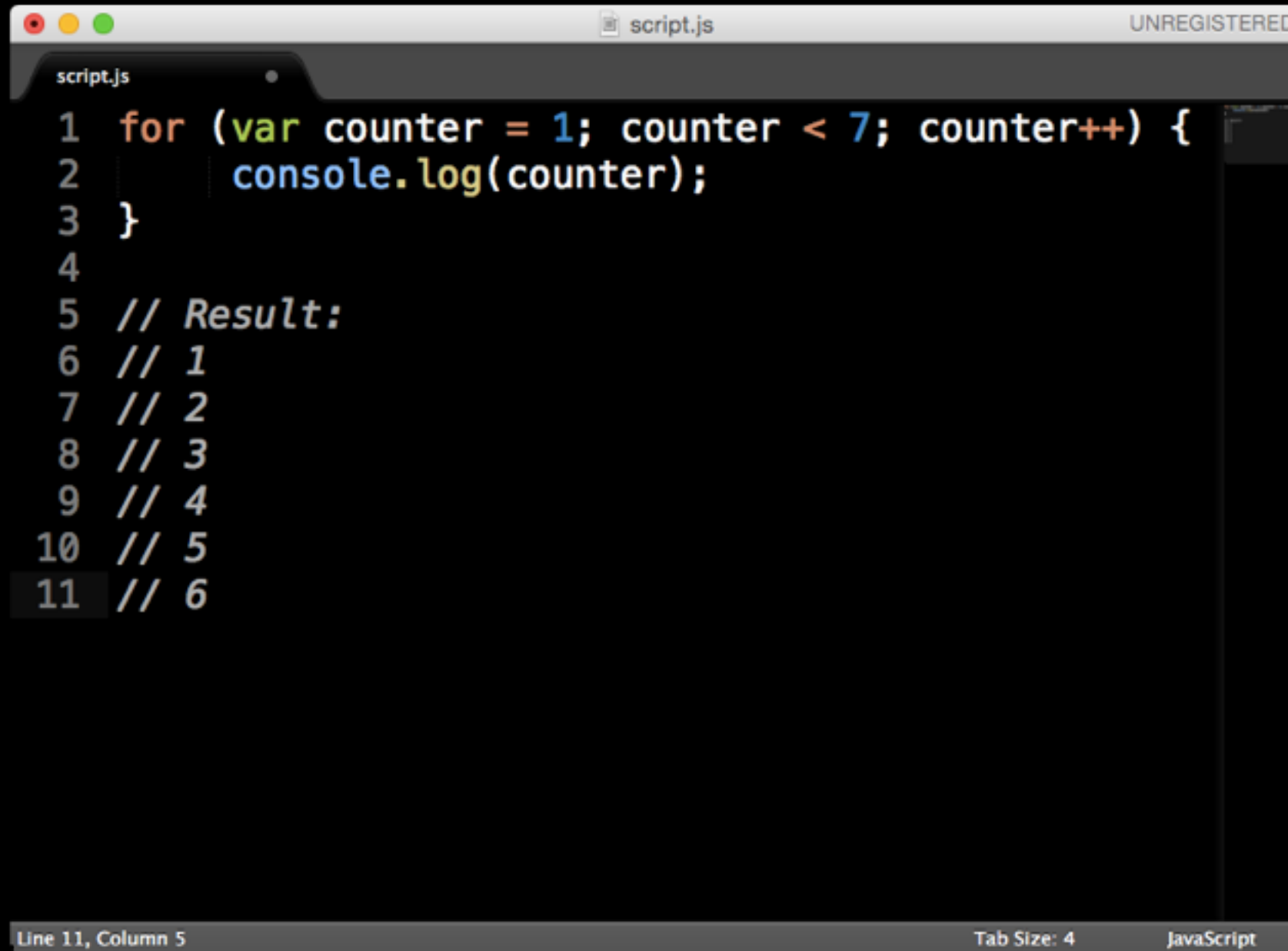
The status bar at the bottom indicates 'Line 5, Column 12', 'Tab Size: 4', and 'JavaScript'.

**Set before
first loop**

**Checked before
every loop**

**Done at the end
of each loop**

For loop — Example



A screenshot of a code editor window titled "script.js" with a "UNREGISTERED" label in the top right corner. The editor contains the following JavaScript code:

```
1 for (var counter = 1; counter < 7; counter++) {  
2     console.log(counter);  
3 }  
4  
5 // Result:  
6 // 1  
7 // 2  
8 // 3  
9 // 4  
10 // 5  
11 // 6
```

The status bar at the bottom indicates "Line 11, Column 5", "Tab Size: 4", and "JavaScript".

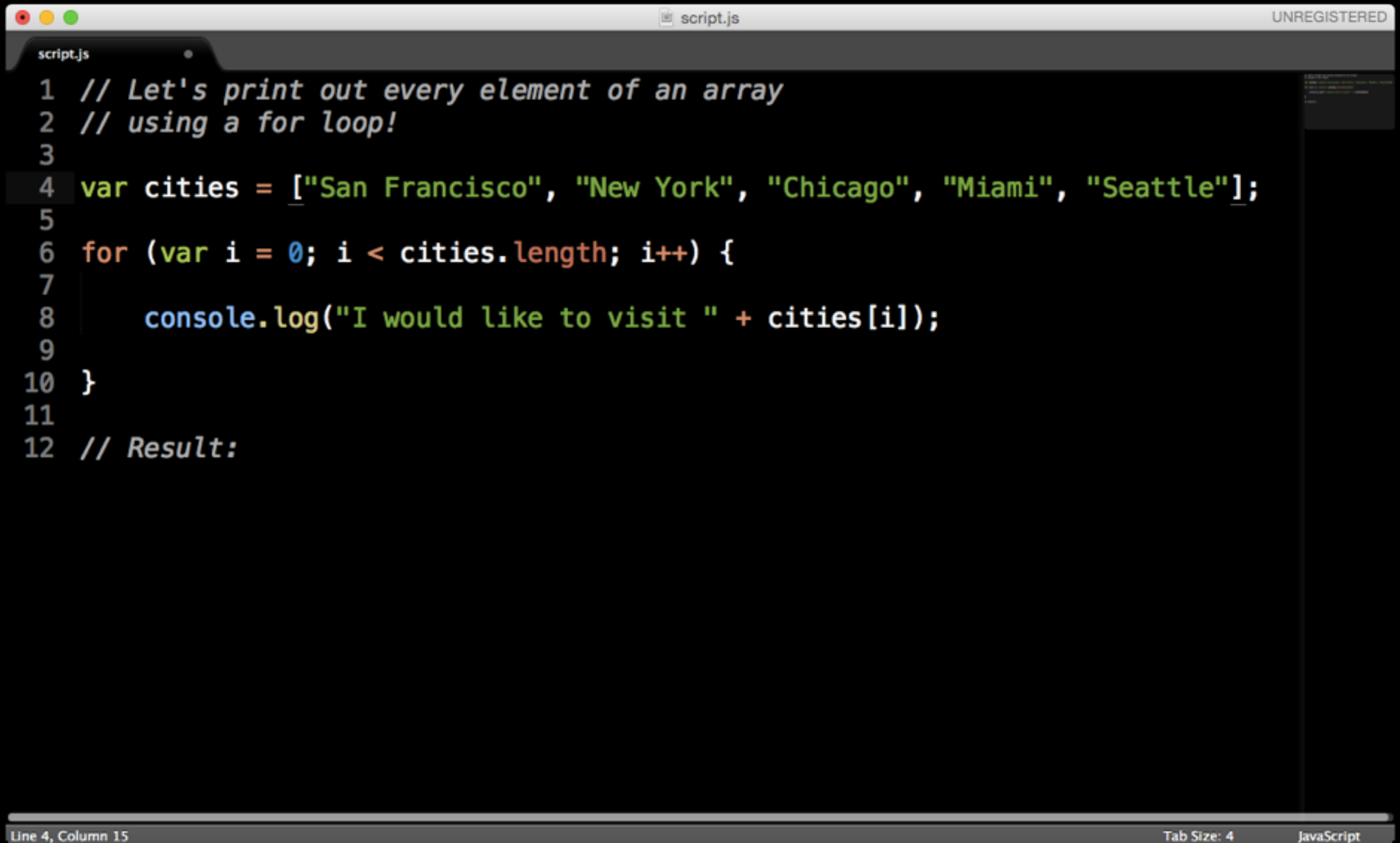
For loop

Repeat a block of code a set number of times. Most often used to access elements of an array one by one.

```
for (counter; condition; increment) {  
    // code to execute  
}
```

**Decides how many times
the loop will run**

For loop w/ array — Example

A screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The editor contains 12 lines of JavaScript code. Line 4 is highlighted. The code defines an array of city names and uses a for loop to iterate through them, logging each city name to the console. The status bar at the bottom shows 'Line 4, Column 15', 'Tab Size: 4', and 'JavaScript'.

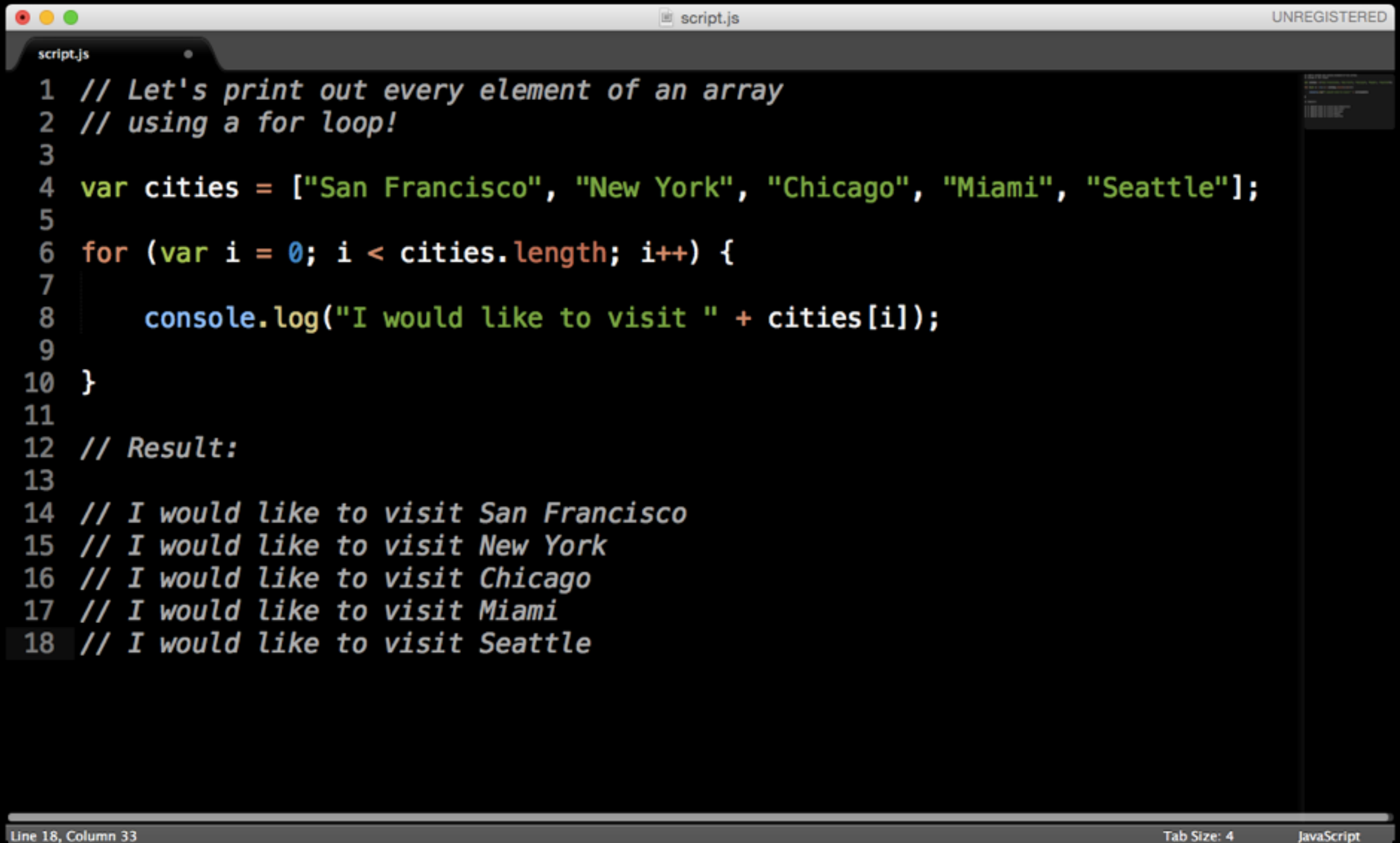
```
1 // Let's print out every element of an array
2 // using a for loop!
3
4 var cities = ["San Francisco", "New York", "Chicago", "Miami", "Seattle"];
5
6 for (var i = 0; i < cities.length; i++) {
7
8     console.log("I would like to visit " + cities[i]);
9
10 }
11
12 // Result:
```

Line 4, Column 15

Tab Size: 4

JavaScript

For loop w/ array — Example

A screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The editor contains 18 lines of JavaScript code. Lines 1-3 are comments. Line 4 declares an array 'cities' with five city names. Lines 6-10 are a for loop that iterates over the array and logs each city name with a message. Lines 12-18 are comments showing the expected output of the loop. The code is syntax-highlighted: keywords like 'var', 'for', and 'console.log' are in blue, strings are in green, and numbers are in orange. The status bar at the bottom shows 'Line 18, Column 33', 'Tab Size: 4', and 'JavaScript'.

```
1 // Let's print out every element of an array
2 // using a for loop!
3
4 var cities = ["San Francisco", "New York", "Chicago", "Miami", "Seattle"];
5
6 for (var i = 0; i < cities.length; i++) {
7
8     console.log("I would like to visit " + cities[i]);
9
10 }
11
12 // Result:
13
14 // I would like to visit San Francisco
15 // I would like to visit New York
16 // I would like to visit Chicago
17 // I would like to visit Miami
18 // I would like to visit Seattle
```

Line 18, Column 33 Tab Size: 4 JavaScript

Functions

Don't Repeat Yourself (D.R.Y.)

The D.R.Y. principle is really important in programming.
No repeating!

Any time you find yourself typing the same thing, but modifying only one small part, you can probably use a function.

Functions — Syntax

First define the function:

```
var functionName = function( variable ) {  
    // code code code  
    // code code code  
    // (more lines of code)  
};
```

Functions — Syntax

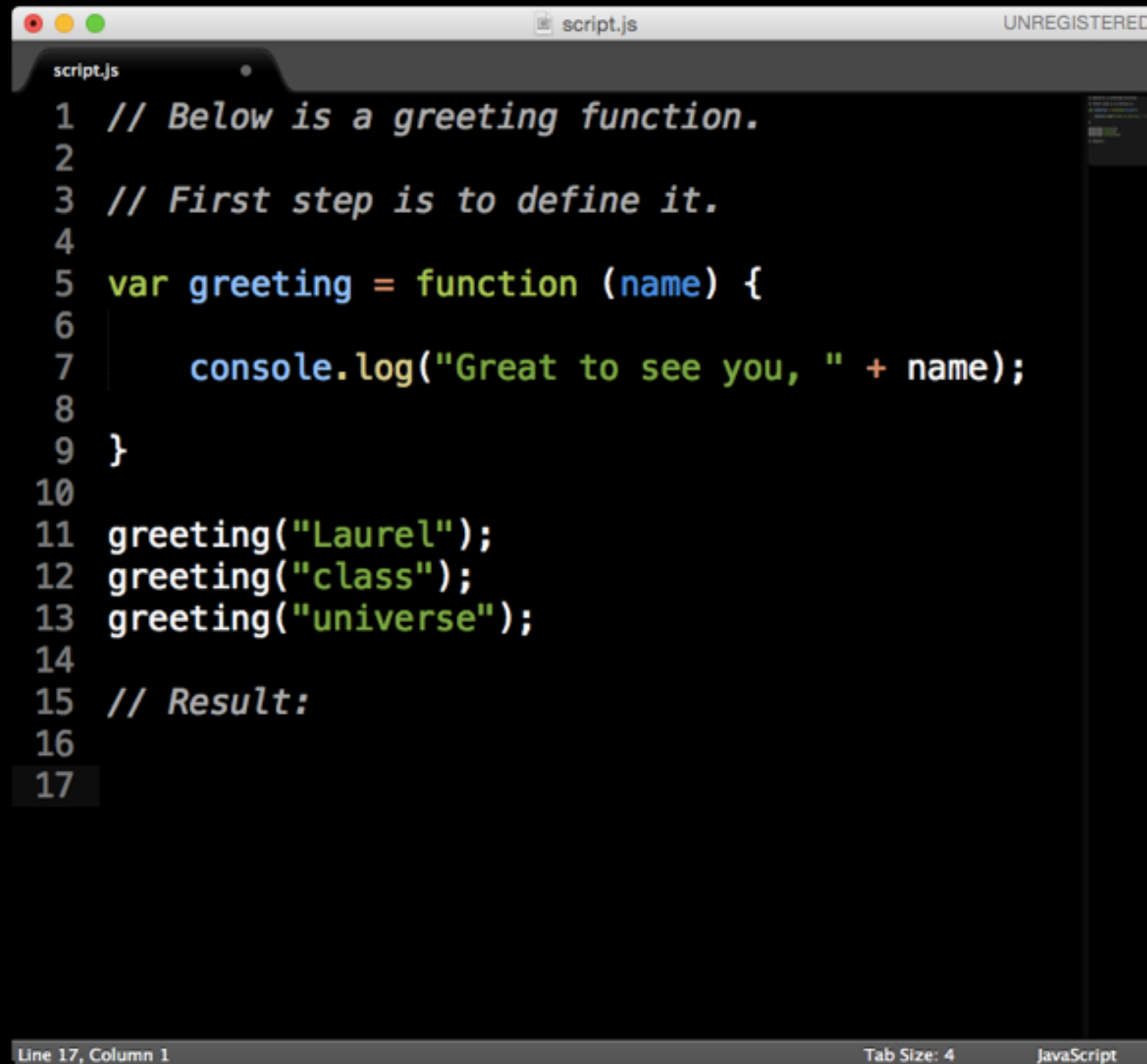
First define the function:

```
var functionName = function( variable ) {  
    // code code code  
    // code code code  
    // (more lines of code)  
};
```

Then you can call it anytime:

```
functionName( value1 );  
functionName( value2 );
```

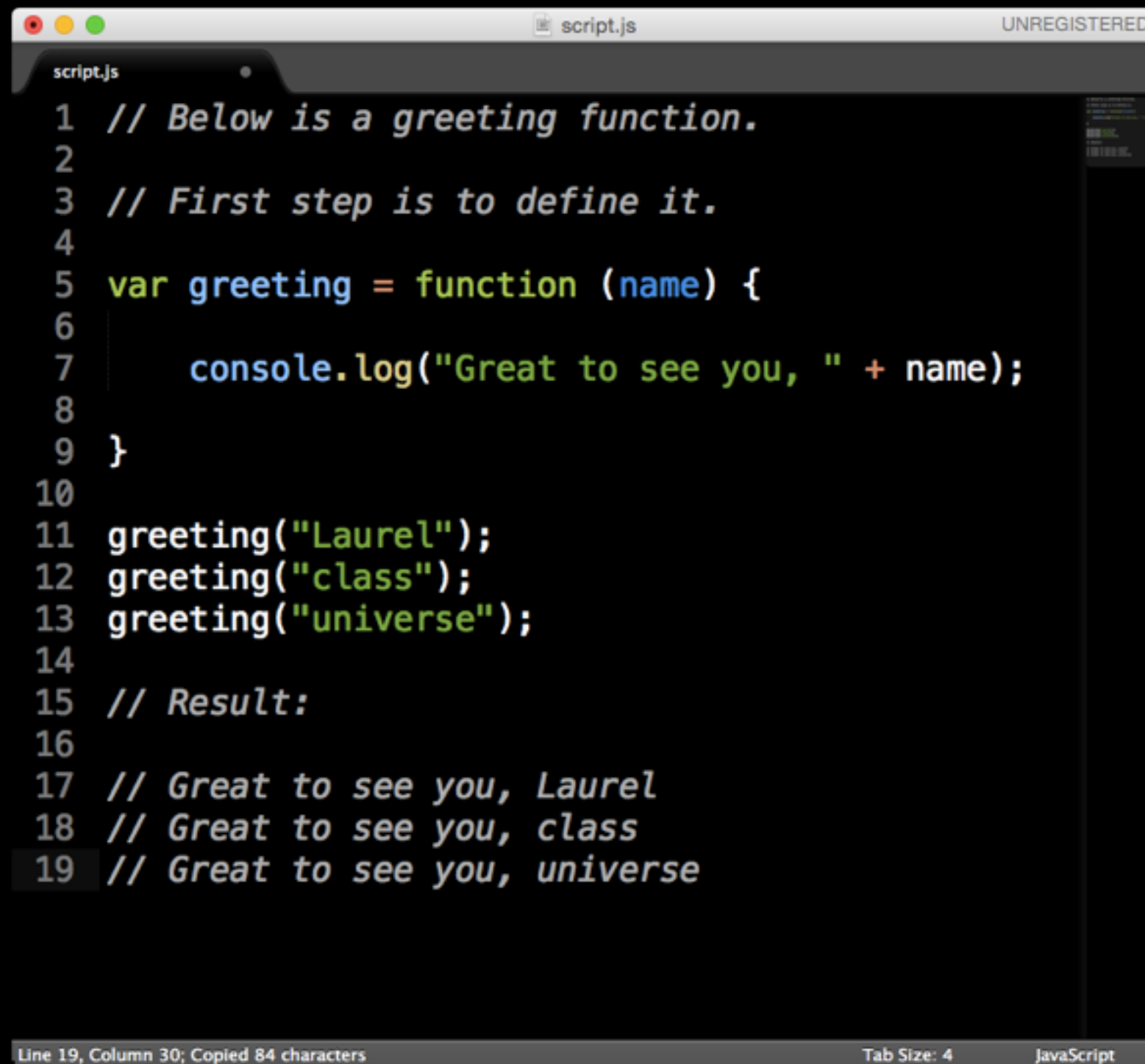
Functions — Example

A screenshot of a code editor window titled 'script.js' with a 'UNREGISTERED' label in the top right corner. The editor contains 17 lines of JavaScript code. Lines 1-3 are comments: '1 // Below is a greeting function.', '2', and '3 // First step is to define it.'. Line 4 is empty. Line 5 starts a function definition: '5 var greeting = function (name) {'. Line 6 is empty. Line 7 has an indented log statement: '7 console.log("Great to see you, " + name);'. Line 8 is empty. Line 9 closes the function: '9 }'. Line 10 is empty. Lines 11-13 call the function: '11 greeting("Laurel");', '12 greeting("class");', and '13 greeting("universe");'. Line 14 is empty. Line 15 is a comment: '15 // Result:'. Line 16 is empty. Line 17 is the current cursor position. The status bar at the bottom shows 'Line 17, Column 1', 'Tab Size: 4', and 'JavaScript'.

```
1 // Below is a greeting function.
2
3 // First step is to define it.
4
5 var greeting = function (name) {
6
7     console.log("Great to see you, " + name);
8
9 }
10
11 greeting("Laurel");
12 greeting("class");
13 greeting("universe");
14
15 // Result:
16
17
```

Line 17, Column 1 Tab Size: 4 JavaScript

Functions — Example

A screenshot of a code editor window titled 'script.js' with a status bar indicating 'UNREGISTERED'. The editor contains 19 lines of JavaScript code. Lines 1-4 are comments. Line 5 defines a function 'greeting' that takes a 'name' parameter and logs a message. Lines 11-13 call the 'greeting' function with 'Laurel', 'class', and 'universe' respectively. Lines 15-19 are comments showing the expected output. The status bar at the bottom shows 'Line 19, Column 30; Copied 84 characters', 'Tab Size: 4', and 'JavaScript'.

```
1 // Below is a greeting function.
2
3 // First step is to define it.
4
5 var greeting = function (name) {
6
7     console.log("Great to see you, " + name);
8
9 }
10
11 greeting("Laurel");
12 greeting("class");
13 greeting("universe");
14
15 // Result:
16
17 // Great to see you, Laurel
18 // Great to see you, class
19 // Great to see you, universe
```

Line 19, Column 30; Copied 84 characters

Tab Size: 4

JavaScript

Variable scope — global vs. local

When you define a variable inside brackets, it only exists inside the brackets!

Local Variable:

```
var bar = function() {  
    var localVar = "haha";  
}  
  
console.log(localVar);    // error
```

Variable scope — global vs. local

When you define a variable inside brackets, it only exists inside the brackets!

Local Variable:

```
var bar = function() {  
    var localVar = "haha";  
}  
  
console.log(localVar);    // error
```


Final notes

Google is your best friend!

So are...

stackoverflow.com

codecademy.com

eloquentjavascript.net

jsfiddle.net

nodeschool.io

:)