Lesson **Weekend**

Workshops (/workshops) / Intro to Programming Workshop (/workshops/intro-to-programming-workshop) / Introduction to JavaScript

Text

So far we've successfully created our project's directory and files through the command line, filled our site with content using HTML via the Atom text editor, and even added styling and formatting using CSS. Great work! Next, we'll explore something called **JavaScript** that will allow us to add interactivity into our site. JavaScript is a little more complex than the topics we've covered so far, so we'll spend two lessons discussing it.

We'll explore and experiment with basic JavaScript in this lesson. Then, we'll implement an interactive form into our site using a specific type of JavaScript called **jQuery** in the next lesson.

Using JavaScript

We've done a good job of building a simple page, and adding some styling to it. However, nothing is interactive, and it remains static - there is no way to make the page *do* anything.

Let's change that.

First let's talk about the tools that let us make our page interactive. We've used HTML and CSS to build the look and content. We're going to introduce a new tool, **JavaScript**. JavaScript (often abbreviated to JS) is the first tool we'll deal with that works behind the scenes, and changes things in the browser.

There is a lot to learn about JavaScript. In fact, once we account for all of the different tools and frameworks built in JavaScript, there is probably more to learn than any single person could commit to memory. At Epicodus, we concentrate on

learning the fundamentals, and learning how to learn new tools, as well as find answers to our specific questions within all that knowledge. That's what the professionals do, so that's how we approach it as well.

Let's write some JavaScript and have it do something.

Let's use our terminal to create a new folder in our project called `js`. Within that folder, create a file called `scripts.js`. Can you remember the commands we used earlier to create folders and files?

Once they exist, let's go into our *index.html* file and link *scripts.js*. It will be similar to how we linked our CSS file, but we'll need to use a slightly different format. Arrange your HTML in *index.html* so that the `<head>` element looks like this.

index.html

```
...
<head>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link rel="stylesheet" href="css/styles.css">
  <script src="js/scripts.js"></script>
  <title>Epicodus Workshop Project</title>
</head>
...
```

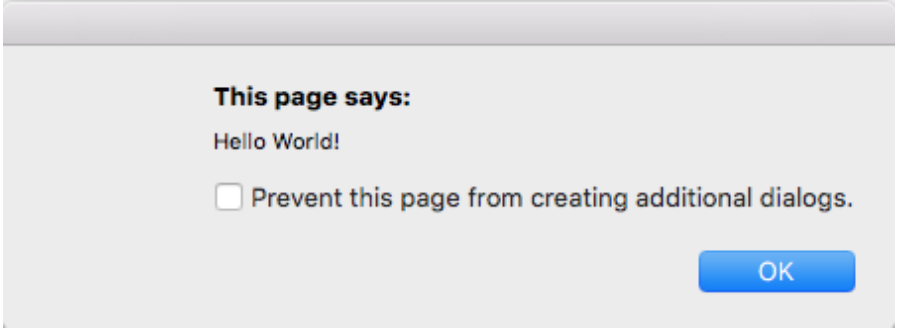
We use the `<script>` element and a `src` attribute to link JavaScript to HTML. Note that scripts are *not* self-closing, so we need to close them.

Now let's open our *scripts.js* file and add some code.

js/scripts.js

```
alert("Hello World!");
```

After you save your files, reload your HTML, and we should get a nice interaction with the webpage.



Isn't that nice? Our page said hi. The box that popped up is called an **alert**.

JavaScript files run immediately when they are loaded. In this case we're calling `alert("Hello World!");`. Let's break this down and talk about what's happening here. This is something called a function. Simply put, a **function** is a piece of code that carries out an operation. We can use all kinds of pre-existing functions, and we can even define our own functions.

Here, we're calling a pre-existing function called `alert()`. Notice the parenthesis? In programming, we call those **parens**, and they're always a good clue that piece of code we're looking at is a function. The parens have a special use, which is to collect data that the function itself will then use. In the case of our function here, we're giving it the phrase `"Hello World!"`.

Our phrase is contained within quotes. This is because in JavaScript, words aren't simply words. the word `"alert"` and the function `alert()` would use the same set of letters, so we need a way to differentiate them. For a function, we used parens, for a piece of text we use quotes. A piece of text contained within quotes is called a **string**. When we write code, everything that needs to be treated as a piece of text must be formatted as a string.

String is a **data type**. There are others. Numbers, for example, are called **integers**. This is because there is a difference between the number `2` and the string `"2"`. Let's see how this works.

Let's change our *scripts.js* to see data types interact. For now, we'll keep using our `alert()` function. Clear the contents of *scripts.js*, and replace it with the following lines of code. Pause before you reload your page and try to predict what we might see.

js/scripts.js

```
alert(2 + 2);
alert(4 * "2");
alert(8 + "2");
```

We were alerted with

- 4
- 8
- 82

How strange. What happened?

- In the first case, JavaScript took our two **integers** and added them, then used alert to show us the sum.
- Next, JavaScript tried to multiply 4 times the **string** `"2"`. That would be like saying "What is the product of 4 times trout". It's a word. We can't do math with a word. So JavaScript changed the `"2"` from a **string** to an **integer** and gave us the result. It *inferred* the data type.
- Finally, JavaScript took the **integer** `8` and tried to add the **string** `"2"`. No problem. But instead of doing a mathematical operation, it seems to have attached the two symbols to one another. This is an operation in programming called **concatenation**. It's literally sticking two things together to make them one thing. This happened because the symbol we use in JavaScript to tell it to concatenate is the symbol `+`. Makes sense, right?

Keep using alert, and experiment with different combinations of strings and integers. Try to guess what will happen before you refresh your page and see if you can start to predict the behavior of the JavaScript function.

Data types and operators are complicated business. There are many different types of data in computer programming, and we take the time to learn a lot of them in our curriculum. We can even define our own data types and set up rules for how they behave. If you'd like to learn more about JavaScript, there are many great resources online. You can check out our lessons at learnhowtoprogram.com (https://www.learnhowtoprogram.com/intro-to-programming/javascript-and-jquery-c950c9ce-679c-4678-ab1f-11881b766e22) or even sign up for our full-time or part-time Intro to Programming (https://www.epicodus.com/) classes.

In our next lesson, let's take this functionality and apply it to do something useful on our page.