Chapter 10

Introduction to JavaScript

Summary: Static websites that use only HTML and CSS are wonderful. But adding true interactivity to any site requires JavaScript. This will introduce you to the basic concepts behind JavaScript.

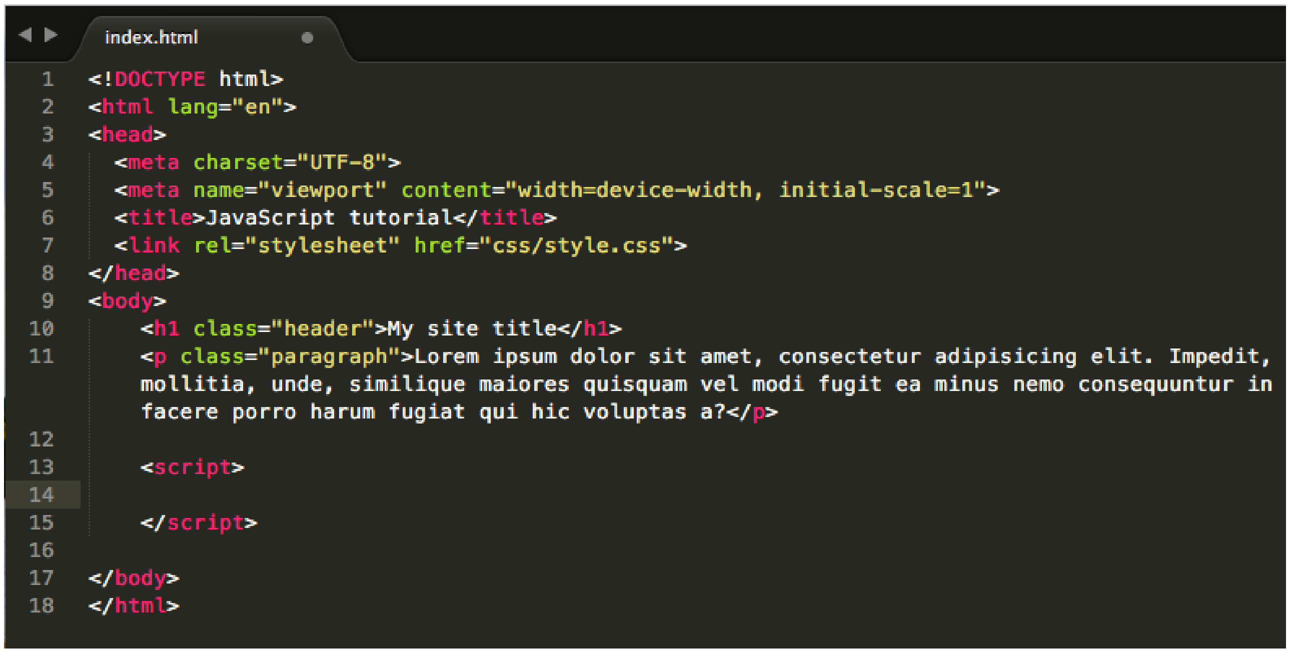
Skills you will learn:

1. The logical structures that underlie the JavaScript programming language
2. How to incorporate simple JavaScript functions into your website

The first thing you'll need is a text editor. See the tutorial **Choosing a Code Editor**.

It is strongly recommended that you complete the tutorial **A short guide to HTML and CSS** before continuing to learn JavaScript. JavaScript is largely meant to manipulate HTML and CSS, so understanding those structures is key to excelling with JavaScript.

JavaScript often lives at the bottom of your HTML document, in a new tag that we haven't explored before: The <script> tag. .



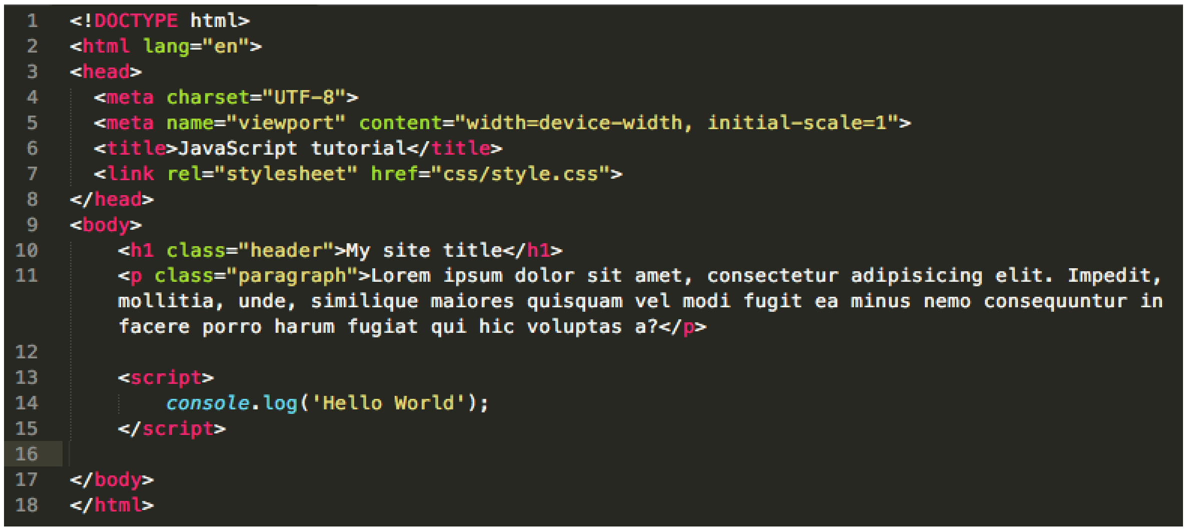
```html
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<title>JavaScript tutorial</title>
	<link rel="stylesheet" href="css/style.css">
</head>
<body>
	<h1 class="header">My site title</h1>
	<p class="paragraph">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Impedit,
	mollitia, unde, similique maiores quisquam vel modi fugit ea minus nemo consequuntur in
	facere porro harum fugiat qui hic voluptas a?</p>

	<script>

	</script>

</body>
</html>
```

Inside the <script> tag you can write custom code that will modify the page when certain conditions are met. These conditions might include a user clicking on something, when the page finishes loading, when a user has been on the page for a certain amount of time or even when the user scrolls to a certain point.

JavaScript is what allows the page to be interactive. This is its personality.

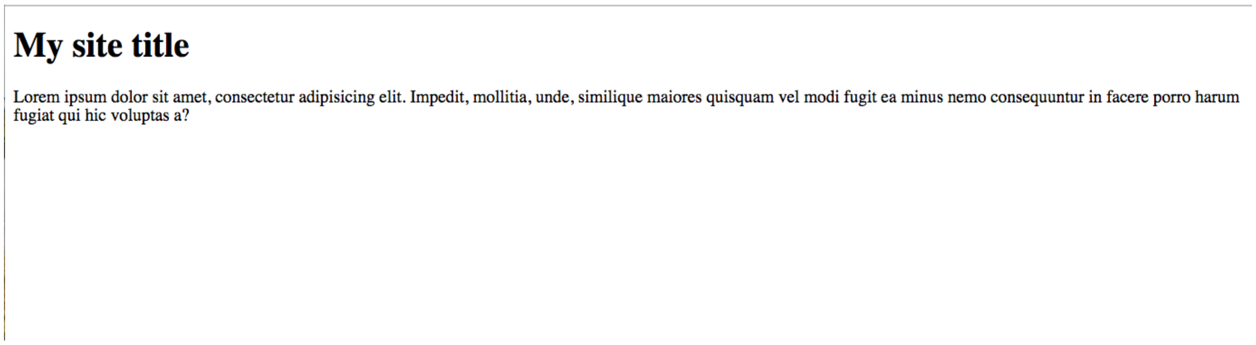
To get started, try using the console function built into JavaScript to print "Hello World" and make sure that everything is working as expected:

```
<script>
console.log('hello world');
</script>
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JavaScript tutorial</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1 class="header">My site title</h1>
    <p class="paragraph">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Impedit,
    mollitia, unde, similique maiores quisquam vel modi fugit ea minus nemo consequuntur in
    facere porro harum fugiat qui hic voluptas a?</p>

    <script>
        console.log('Hello World');
    </script>

</body>
</html>
```

Save your file and open it in a web browser. You'll only see the visible elements on the page at this point. JavaScript is working in the background and isn't, yet, modifying what the user sees.

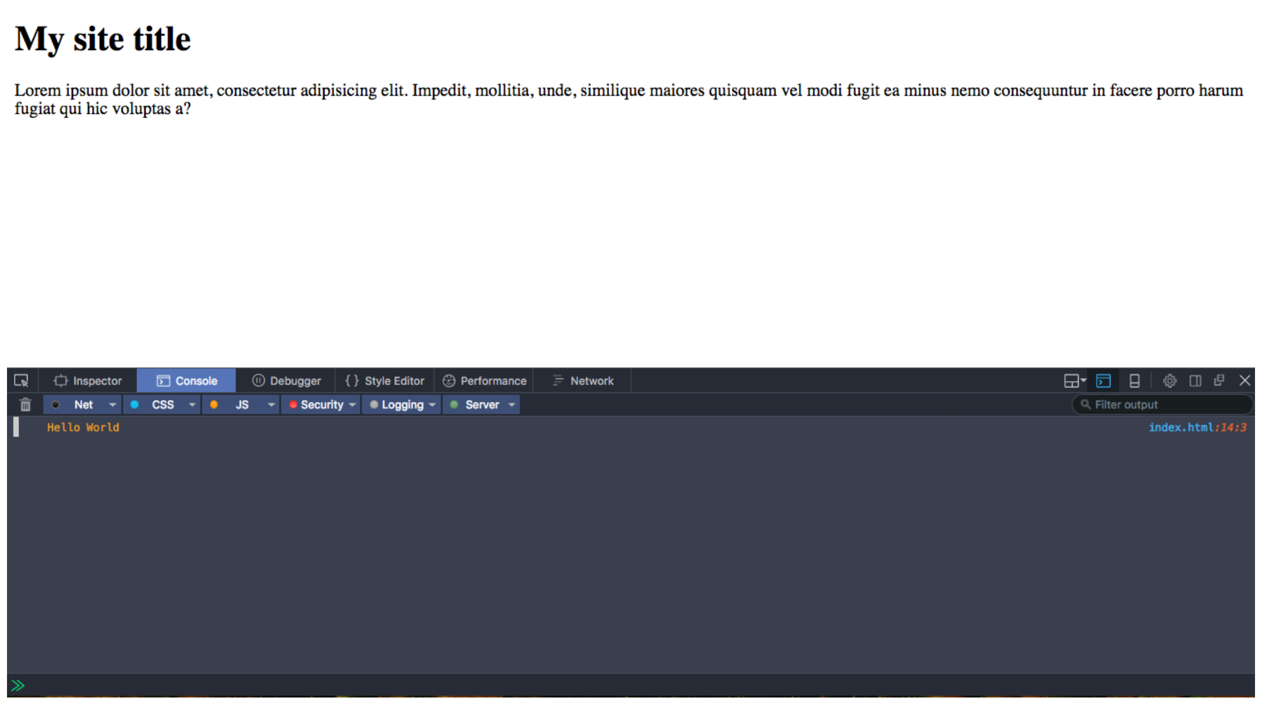


My site title

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Impedit, mollitia, unde, similique maiores quisquam vel modi fugit ea minus nemo consequuntur in facere porro harum fugiat qui hic voluptas a?

To see the effects, you have to open the console. In Firefox, you can go to the Tools menu, click "Web Developer" and then "Web Console." You can also use the keyboard shortcut command-option-k. In Chrome, you can click the View menu, then "Developer" and then "JavaScript Console." You can also use the keyboard shortcut command-option-j.

Once you have the console open, you should see the phrase Hello World printed there. This is what console.log() does: It logs text to the console in your browser.



My site title

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Impedit, mollitia, unde, similique maiores quisquam vel modi fugit ea minus nemo consequuntur in facere porro harum fugiat qui hic voluptas a?

```
Hello World                                                        index.html:14:3
```

JavaScript is a programming language built into all web browsers and your <script> tags are interpreted as soon as the page loads. That puts some very powerful tools at your disposal.

The core concept you need to understand first is the variable. Variables contain bits of information that we can pass around the program for later use. This avoids us from having to type everything that we want to do every time. It also allows us to manipulate information.

A variable can contain either numbers or letters or both. It can contain material submitted by users or material that you define. You can change a variable and perform operations such as math and text manipulation with them. They're extremely flexible and the core concept behind all programming languages.

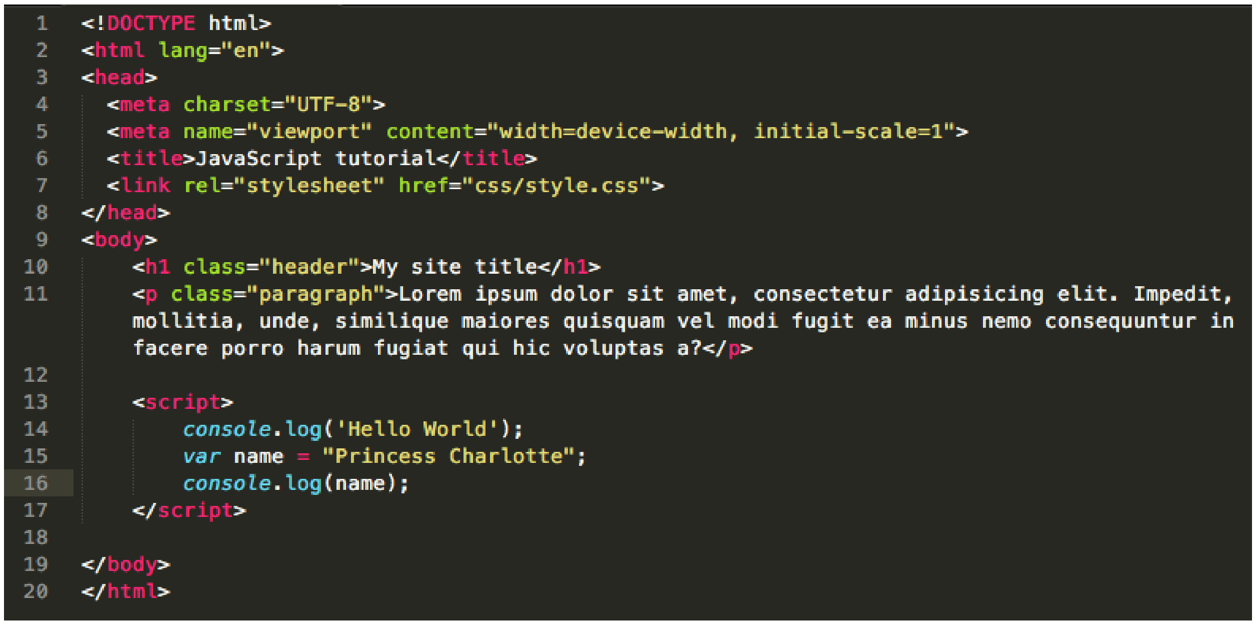
Here is how you create one:

```
var name = "Princess Charlotte"
```

That keyword "var" in front of it means you're creating a variable. Then the little equals sign is all it takes to assign it a value. We have a variable called name. Then we assigned it the value “Princess Charlotte”.

So try this:

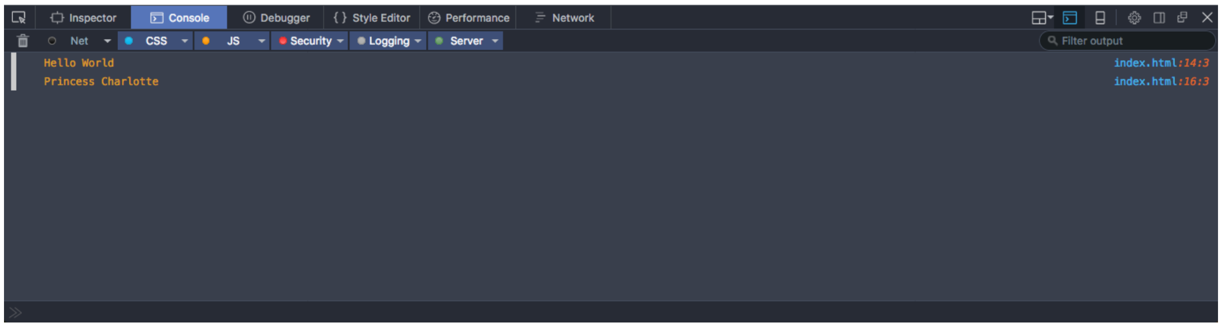
```
var name = “Princess Charlotte”;
console.log(name);
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JavaScript tutorial</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1 class="header">My site title</h1>
    <p class="paragraph">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Impedit,
    mollitia, unde, similique maiores quisquam vel modi fugit ea minus nemo consequuntur in
    facere porro harum fugiat qui hic voluptas a?</p>

    <script>
        console.log('Hello World');
        var name = "Princess Charlotte";
        console.log(name);
    </script>

</body>
</html>
```

Now save your file and refresh the page in your web browser.

See what happened? The first time we did this, we put quotation marks around the name “Princess Charlotte” to indicate it was just text. But the second time we did it, we just used the variable name with no quotation marks. Quotation marks indicate literal text. Removing the quotation marks means we have a variable.



```
Hello World                                         index.html:14:3
Princess Charlotte                                  index.html:16:3
```

Note that variable names must not contain spaces and cannot start with numbers. Otherwise, the limit is your imagination: You can call variables whatever you like.

You can also do math with variables. Try:

```
var number1 = 5;
var number2 = 15;
console.log(number1 * number2);
```

This also works. .The * is a multiplication operator and the two numbers are multiplied so only one number is logged to the console at the end. You can also use different operators like + - and / to expand your repertoire.

But clearly what we really want to do is change the way our webpage behaves. We don't always want to be pushing personal notes to ourselves or other developers. We actually want to take control of the page.

In your HTML file, give your paragraph tag (<p>) an ID attribute:

```
<p id="myTargetText">
```

The ID is similar to a class, but an ID is only used once in your entire website. That means that an ID can be used to identify a singular element on a page instead of a class which can be used to identify a group of elements on a page that must behave similarly.

ID names follow the same rules as class names in HTML, and give you all the flexibility you need to name them creatively but also descriptively.

The JavaScript that we would write to control that ID is:

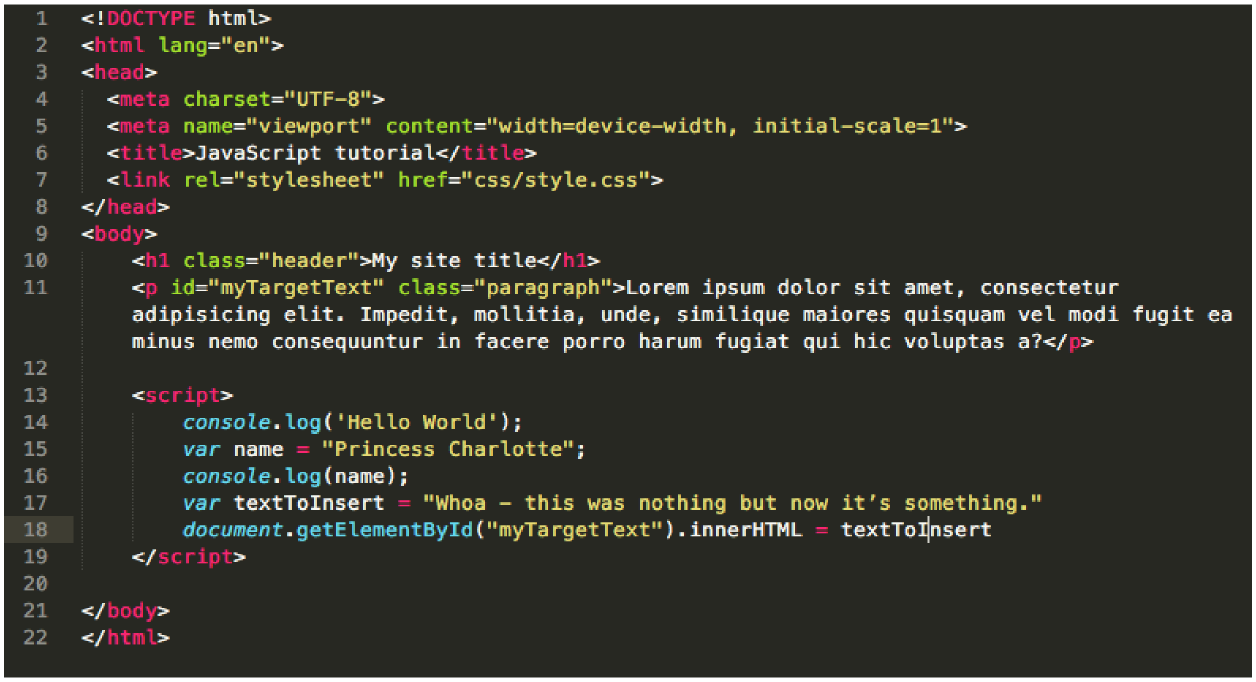
```
document.getElementById("myTargetText").innerHTML = "Whoa -
this was nothing but now it's something.";
```

Now save the file and refresh the page in your web browser. The text in that <p> tag should have changed to the text you inserted in quotation marks in your script.

Now replace it with this:

```
var textToInsert = “Whoa - this was nothing but now it’s
something.”
document.getElementById("myTargetText").innerHTML =
textToInsert
```

Now you’ve combined variables with manipulation of your web page.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JavaScript tutorial</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1 class="header">My site title</h1>
    <p id="myTargetText" class="paragraph">Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Impedit, mollitia, unde, similique maiores quisquam vel modi fugit ea
    minus nemo consequuntur in facere porro harum fugiat qui hic voluptas a?</p>

    <script>
        console.log('Hello World');
        var name = "Princess Charlotte";
        console.log(name);
        var textToInsert = "Whoa - this was nothing but now it’s something."
        document.getElementById("myTargetText").innerHTML = textToInsert
    </script>

</body>
</html>
```

This is what JavaScript is good at: Manipulating information then presenting it in the way that you see fit.

JavaScript can also contain lists. This is useful when you don’t want a whole pile of different variables containing people’s names, for example, but instead just want one variable with a list of everyone’s names.

In JavaScript, this is called an array. Array’s look like this:

```
Var myList = [“Name”, “Second Name”, “Third Name”]
```

The square brackets surround them, commas separate each item in the list. You can reference them also with square brackets. So if you want the second name in that list, you and say myList[1] If this looks just like a list in Python, you are right. It's exactly the same concept.

So console.log(myList[1]) in this example would log to the console "Second Name".

That's right, the second one in the list is referenced with the number one. Like Python, JavaScript starts counting at 0 instead of one. So 0 is the first in the list, 1 is second in the list and 2 is third in the list, and so on.

You can also write little mini programs that go through your list and act on each item in it. For example, make a list that looks like the one above. Then try a few lines of JavaScript that look like this:

```
var myList = ["Name", "Second Name", "Third Name"];

for (var i =  0; i <= myList.length - 1; i++) {
    var newParagraph = document.createElement('p');
    newParagraph.innerHTML = myList[i]
    document.body.appendChild(newParagraph);
}
```

There are a ton of things happening here, but there are also things that we've already covered and aren't going to be new. The general concept here is called a loop: We're looping over the same bit of code several times.

The "for" sets it all up. In real english, what this says is:

> Start my counter (the i variable) at 0. And while the i variable is less than the length of my list, keep looping over this bit of code. Increase the i variable by 1 at the end of each loop.

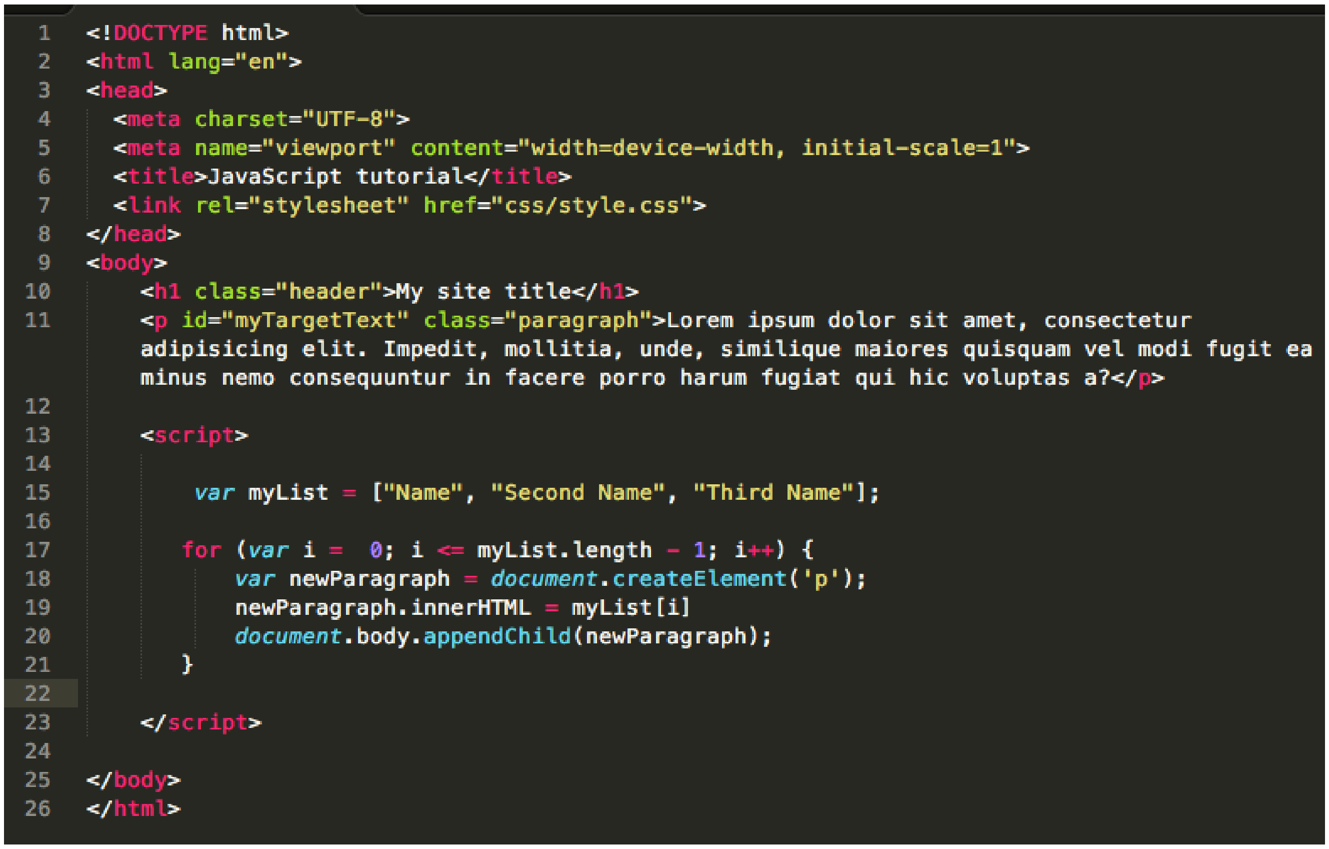
Inside the curly braces is the part of the code that gets looped over several times. Here's what that does:

newParagraph is a variable we made that is going to build a new p tag for us. We use the document and then the createElement command to make the new p tag, but right now it's not on the page, it's just stored in a variable, waiting for us to do something with it.

Then we changed the newParagraph variable, giving it innerHTML like we did before. But this time we're giving it the innerHTML of what's stored in our list.

Once we have a new p tag and text inside it, we append it to our body. That's when it appears on our page. We do that for every name in our list, and it's like magic.

Modifying our page.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JavaScript tutorial</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1 class="header">My site title</h1>
    <p id="myTargetText" class="paragraph">Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Impedit, mollitia, unde, similique maiores quisquam vel modi fugit ea
    minus nemo consequuntur in facere porro harum fugiat qui hic voluptas a?</p>

    <script>

        var myList = ["Name", "Second Name", "Third Name"];

       for (var i =  0; i <= myList.length - 1; i++) {
           var newParagraph = document.createElement('p');
           newParagraph.innerHTML = myList[i]
           document.body.appendChild(newParagraph);
       }

    </script>

</body>
</html>
```

Of course you don't want everything to happen on page load, triggering as soon as someone arrives on your website. Sometimes you want to wait until they've engaged with the site in some way, scrolled down a page, or clicked on a button.

In JavaScript language, these are called "events." Events are triggered when a user clicks, scrolls, hovers, loads or otherwise interacts with a page in any way. Your web browser is constantly firing off hundreds of events, and we can tell JavaScript to listen for some of them.

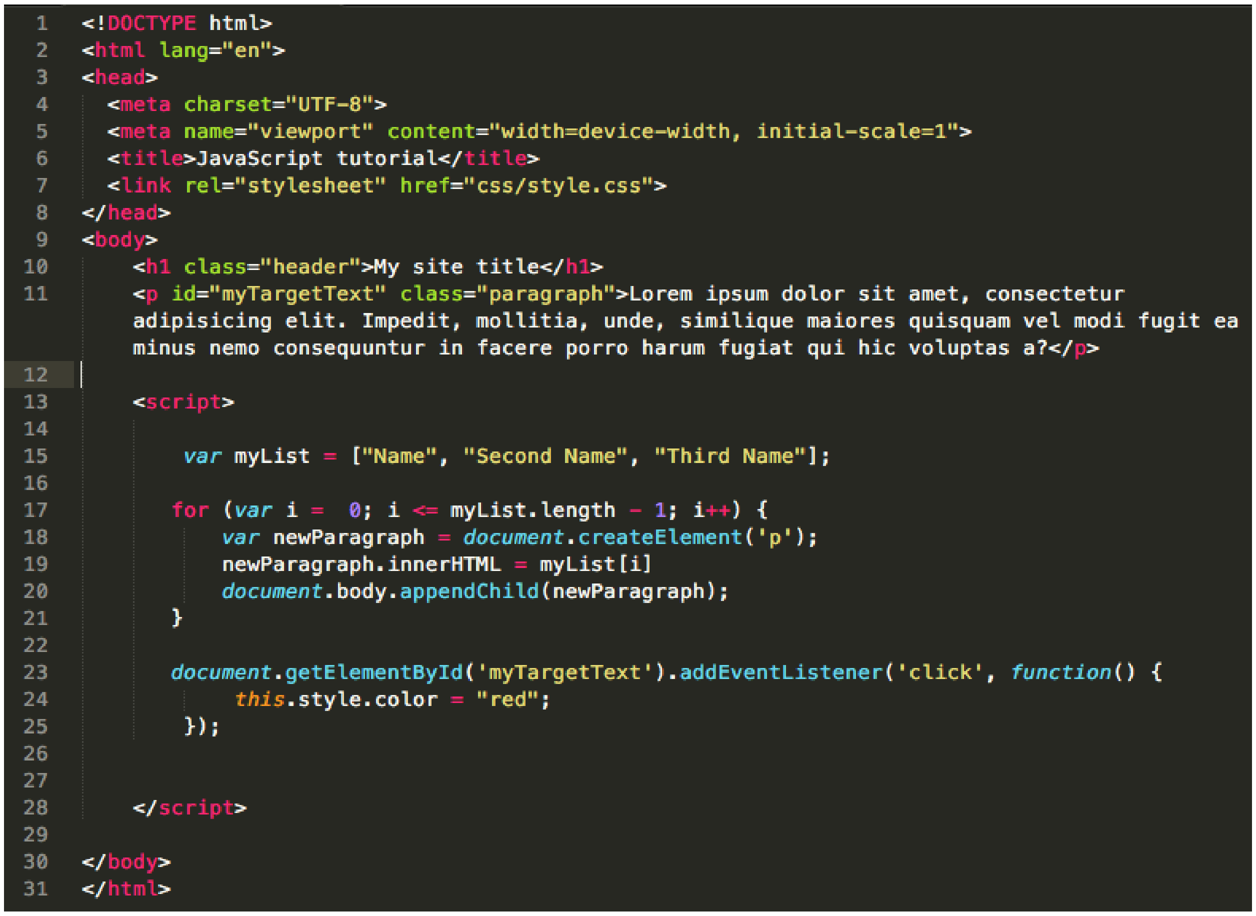
For example, click events fire every time you click on anything on a web page. We can use a little block of code like this to tell JavaScript to listen for it.

```
docu-
ment.getElementById('myTargetText').addEventListener('click
', function() {
     this.style.color = "red";
});
```

This is how click events work. The first part should be familiar from above: document.getElementbyId is how you select which element on the page you want to pay attention to. Then the .addEventListener tells that element to listen to one of these many events that are firing in your browser.

Then it gets complicated. The ('click', function() { }); portion of the code is a bit messy, and that mess is one of the things that JavaScript is known for. But the "click" is the type of event that we're listening for. The function() { } is what happens after that click.

Functions are blocks of code that only run when they're called. They're very useful in situations like these to run little snippets of code when someone clicks in the right spot. In functions, the code you want to run goes in between those curly braces.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JavaScript tutorial</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1 class="header">My site title</h1>
    <p id="myTargetText" class="paragraph">Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Impedit, mollitia, unde, similique maiores quisquam vel modi fugit ea
    minus nemo consequuntur in facere porro harum fugiat qui hic voluptas a?</p>

    <script>

        var myList = ["Name", "Second Name", "Third Name"];

       for (var i =  0; i <= myList.length - 1; i++) {
           var newParagraph = document.createElement('p');
           newParagraph.innerHTML = myList[i]
           document.body.appendChild(newParagraph);
       }

       document.getElementById('myTargetText').addEventListener('click', function() {
            this.style.color = "red";
        });


    </script>

</body>
</html>
```

What this does is set up a click listener on the paragraph with the id “'myTargetText'” and when the user clicks on it, it modifies the paragraph style to turn it red.

JavaScript is one of those tools that can be as complicated as you want it to be. There are always new ways to implement it, and new tricks to learn. But these are the basics that will get you started. Once you know how variables work, how functions work, how loops work and the basics of how JavaScript works in the browser the rest is relatively straightforward to pick up.