How It Works: Adding Actions to the Web Page with JS

In this lesson, we'll understand the workings of the previous lesson's exercise.
Let's begin!

HOW IT WORKS

The key to this behavior is the `handleClick` **function** you added to **index.html** in step 3.

index.html

```
<!DOCTYPE html>
<html>
<head>
```

```
  <title>Table of Contents</title>
  <link href="style.css" rel="stylesheet" />
</head>

<body>
  <h1 onclick="handleClick(this)">
    Introduction
  </h1>
  <h2>Whom this book is for?</h2>
  <h2>Errata</h2>
  <h1 onclick="handleClick(this)">
    Chapter 1
  </h1>
  <h2>What you will learn in this chapter</h2>
  <h2>Summary</h2>
  <h1 onclick="handleClick(this)">
    Chapter 2
  </h1>
  <h2>Recap</h2>
  <h2>Conclusion</h2>

  <script>
    function handleClick(node) {
      var value = node.getAttribute('class') || '';
      value = value === '' ? 'clicked' : '';
      node.setAttribute('class', value);
    }
  </script>
</body>
</html>
```

When loading a page, the browser recognizes the `<script>` sections and immediately executes the code within. Here, the `<script>` contains a function definition and executing the code means that the definition is hoisted into the current page's JavaScript context. Let's have a look at what this function does.

```
1  function handleClick(node) {
2    var value = node.getAttribute('class') || '';
3    value = value === '' ? 'clicked' : '';
4    node.setAttribute('class', value);
5  }
```

Line 1 above defines the function that accepts a single parameter—which is assumed to be an HTML element in the following code, although nothing marks this fact.

Line 2 gets the value of that element's class attribute, or results in an empty string if there is no such attribute defined within the element.

Line 3 toggles this value between "clicked" and empty.

Line 4 assigns the toggled value to the class attribute of the element passed in the input argument.

The `handleClick` function is triggered by the `onclick` event handler `(onclick="handleClick(this)")` so that the element receiving the click event is passed as the input of `handleClick`. As a result, when you click an `<h1>` element, the class attribute of that element will be changed from nothing or an empty string to `“clicked”`, and from `“clicked”` to empty string.

The `h1.clicked` style selector you added its definition to **style.css** in step 1, represents the style of all `<h1>` elements that have a class attribute with the value clicked. Changing the class attribute to clicked triggers the rendering engine of the browser to apply the `h1.clicked` style, which results in an inversion of the original text and background colors.

style.css

```css
body {
  font-family: Verdana, Arial, sans-serif;
}
h1 {
  color: white;
  background-color: navy;
}
h1.clicked {
  color: navy;
  background-color: lightgray;
}
h2 {
  color: green;
  margin-left: 40px;
  border-bottom: 4px dotted black;
}
```

Complete live demo at your service!

The complete implementation of the above exercise is given below for you to play around and experiment with.

Learn and enjoy! :)

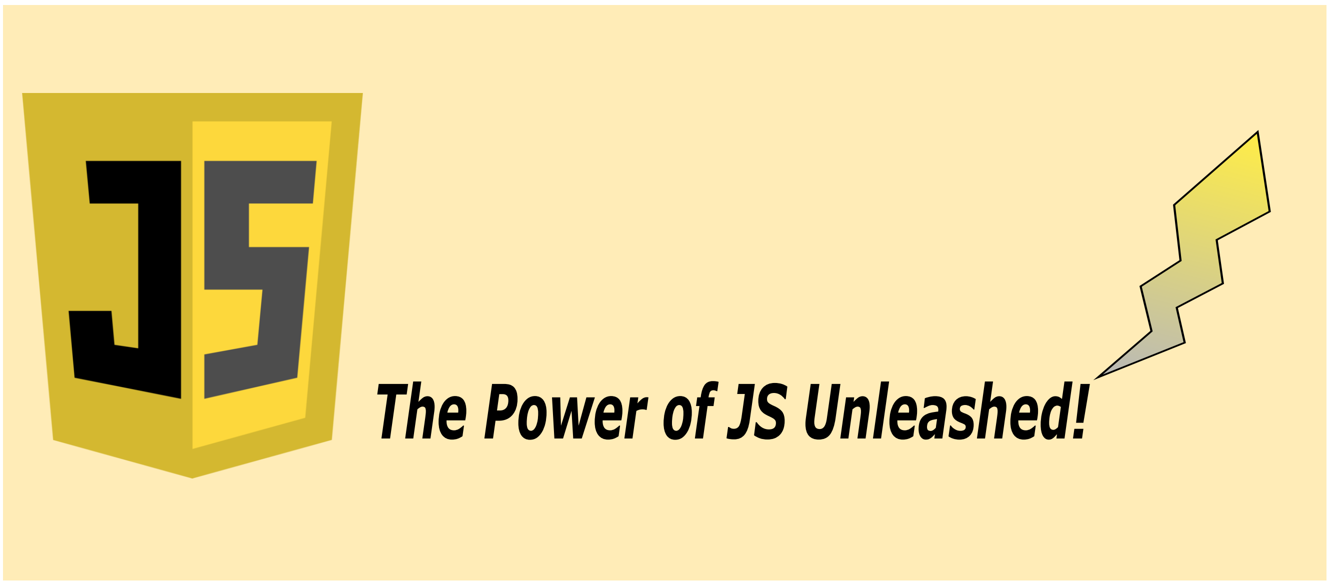
```html
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Table of Contents</title>
  <link href="style.css" rel="stylesheet" />

</head>
<body >
  <h1 onclick="handleClick(this)">Introduction</h1>
  <h2>Whom this book is for?</h2>
  <h2>Errata</h2>
  <h1 onclick="handleClick(this)">Chapter 1</h1>
  <h2>What you will learn in this chapter</h2>
  <h2>Summary</h2>
  <h1 onclick="handleClick(this)">Chapter 2</h1>
  <h2>Recap</h2>
  <h2>Conclusion</h2>

  <script>
    function handleClick(node) {
      var value = node.getAttribute('class') || '';
      value = value === '' ? 'clicked' : '';
      node.setAttribute('class', value);
    }
  </script>

</body>
</html>
```



This very short JavaScript code carried out a simple action. Due to the power of JavaScript, you can do so much with only a few lines of code.

There are many typical tasks that can easily be done with JavaScript, so the community created libraries like **jQuery**, **Bootstrap**, **AngularJS**, **Knockout.js** and many, many more.

As this exercise suggested, with JavaScript you can change the structure of the page, or, more technically, the **DOM (Document Object Model)** behind the page. When you toggled the class attribute of `<h1>` elements in the exercise, you actually changed the DOM.

Show Fun Fact

In the *next lesson*, we’ll discover the workings of the **DOM** and get a high-level overview of what it really is.

Stay tuned! :)