1 Debugging JavaScript 3 sections, 1 - 2 hours

- JavaScript Errors (/courses/intermediate-javascript/javascript-debugging-errors)
- Debugging with the Sources Tab (/courses/intermediate-javascript/javascript-debugging-with-the-sources-tab)
- Debugging Exercises (/courses/intermediate-javascript/javascript-debugging-exercises)

2 Nested Data Structures 4 sections, 2 - 3 hours

- Nested Objects (/courses/intermediate-javascript/javascript-nested-data-structures-objects)
- Nested Objects Exercises (/courses/intermediate-javascript/javascript-nested-data-structures-objects-exercises)
- Nested Arrays (/courses/intermediate-javascript/javascript-nested-data-structures-arrays)
- Nested Arrays Exercises (/courses/intermediate-javascript/javascript-nested-data-structures-arrays-exercises)

3 Higher Order Functions, Timers, and Closures 4 sections, 3 - 4 hours

- Higher Order Functions (/courses/intermediate-javascript/javascript-higher-order-functions)
- Timers (/courses/intermediate-javascript/javascript-timers)
- Closures (/courses/intermediate-javascript/javascript-closures)
- Higher Order Functions, Timers, and Closures Exercises (/courses/intermediate-javascript/javascript-hof-timers-closures-exercises)

4 Document Object Model 3 sections, 2 - 3 hours

- Introduction to the DOM (/courses/intermediate-javascript/javascript-dom-introduction)
- DOM Manipulation (/courses/intermediate-javascript/javascript-dom-manipulation)
- DOM Exercises (/courses/intermediate-javascript/javascript-dom-exercises)

5 Events and LocalStorage 4 sections, 4 - 5 hours

- Introduction to Events (/courses/intermediate-javascript/javascript-dom-events-introduction)
- Events Continued (/courses/intermediate-javascript/javascript-dom-events-continued)
- Local Storage (/courses/intermediate-javascript/javascript-dom-local-storage)
- Events Exercises (/courses/intermediate-javascript/javascript-dom-events-exercises)

6 JavaScript Iterators 4 sections, 2 - 3 hours

- forEach, Map, and Filter (/courses/intermediate-javascript/javascript-iterators-foreach-map-filter)
- Reduce (/courses/intermediate-javascript/javascript-iterators-reduce)
- Additional Array Methods (/courses/intermediate-javascript/javascript-iterators-additional-array-methods)
- Iterators Exercises (/courses/intermediate-javascript/javascript-iterators-exercises)

Previous (/courses/intermediate-javascript/javascript-dom-introduction) | Table of Contents (/courses/intermediate-javascript) | Next (/courses/intermediate-javascript/javascript-dom-exercises)

{ DOM Manipulation. }

»

Objectives

By the end of this chapter, you should be able to:

- Change attributes and values on DOM elements
- Traverse the DOM
- Add, create and remove DOM elements

In this section we'll continue to use the same HTML setup as before. Here we'll use it to examine some functions we can use to traverse the DOM. In case you need it:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body id= "container">
    <div class="hello">
        Hello World
    </div>
    <div class="hello">
        Hello Everyone!
    </div>
    <a href="#">This link goes nowhere!</a>
    <button>Click me!</button>
    <!-- make sure for now that your script tag is placed here -->
</body>
</html>
```

Modifying properties and attributes on elements in the DOM

We can change the text of an element through the `innerHTML` property.

```
var firstDiv = document.getElementsByTagName("div")[0];

firstDiv.innerHTML = "Just changed!";
```

This can also be done using the `innerText` property.

```
var secondDiv = document.getElementsByTagName("div")[1];

secondDiv.innerText = "Just changed Again!";
```

What's the difference between `innerText` and `innerHTML` ? You can find it here (http://stackoverflow.com/questions/19030742/difference-between-innertext-and-innerhtml-in-javascript).

We can also directly manipulate the `CSS` properties for elements (through inline styling) with the `style` property.

```
var firstDiv = document.getElementsByTagName("div")[0];

firstDiv.style.color = "red";
firstDiv.style.backgroundColor = "teal";
```

Notice that if you're accessing CSS properties using dot notation, you need to camelCase those property names, since `firstDiv.style.background-color` is invalid JavaScript. (Bonus question: what do you think will happen if you try typing this in the console?) However, if you use brackets, you can write the properties the same way as you would in a stylesheet:

```
firstDiv.style["background-color"] = "purple"; // this works too
```

If we want to access/modify attributes on elements, we can do that with `getAttribute` and `setAttribute`:

```
var body = document.getElementById("container");

body.getAttribute("id"); // "container"
body.setAttribute("id", "new_container");
body.getAttribute("id"); // "new_container"
```

We can also add and remove classes to elements using `classList`

```
var secondDiv = document.getElementsByTagName("div")[1];

secondDiv.classList; // ["hello"]
secondDiv.classList.add("another_class");
secondDiv.classList; // ["hello", "another_class"]
secondDiv.classList.remove("hello");
secondDiv.classList; // [another_class"]
```

Notice here that we use methods called `add` and `remove`, not the `push` and `pop` methods we've seen when dealing with arrays. This is because the `classList` isn't actually an array and doesn't have `push` or `pop` methods on it.

Nodes vs Elements

When you read documentation about the DOM, you will see the terms `node` and `element`. In JavaScript these two are different and you can read more about the difference here (http://stackoverflow.com/questions/9979172/difference-between-node-object-and-element-object). In short, there are many different types of nodes, but the ones we will most often be working with are ELEMENT_NODE and TEXT_NODE (you can see them all here (https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType)). Element nodes are HTML elements (`div`, `span` etc.). Text nodes are the actual text of an element node.

Traversing the DOM

Another very common operation when working with the DOM is trying to find elements inside of other elements. When we travel through the DOM in search of something, we are doing what is called DOM `traversal`. Here are a few common methods we can use for finding elements and/or text nodes in relation to an element that we have already found.

```
var container = document.getElementById("container");
container.childNodes; // // this contains all of the nodes (including text nodes) that are
container.childNodes.length; // 11
container.children; // this contains all of the elements that are children of the element
container.children.length; // 5

var link = document.querySelector("a");
link.parentElement; // <body id="container">...</body>
link.previousElementSibling; // <div class="hello">Hello Everyone!</div>
link.previousSibling; // text node
link.nextSibling; // text node
link.nextElementSibling; // <button>Click me!</button>
```

Creating elements

To create elements we use the `.createElement` function on the `document` object and pass in a string with the name of the element that we would like to create. This will just return a new HTML element without any text/attributes or placement on the page!

```
var newDiv = document.createElement("div");
```

So now that we created this element, how do we place it on the page?

Appending elements

```
var button = document.createElement("button");
button.innerText = "I am a button created with JavaScript!";

var container = document.getElementById("container");
container.appendChild(button);
```

And now that we created an element, how do we remove one?

Removing elements

```
var linkToBeRemoved = document.querySelector("a");

var container = document.getElementById("container");
container.removeChild(linkToBeRemoved);
```

Changing multiple elements

So what would happen if we wanted to change all of our divs to have a background color of red? Unfortunately, we can not do that without a loop:

```
var divs = document.querySelectorAll("div");
divs.style.backgroundColor = "red"; // this will not work, try to understand the error you

// we have to use a loop for each one instead.
for (var i = 0; i < divs.length; i++) {
  divs[i].style.backgroundColor = "red"; // this will work!
}
```

When you're ready, move on to DOM Exercises (/courses/intermediate-javascript/javascript-dom-exercises)

Continue (/courses/intermediate-javascript/javascript-dom-exercises)

{R} Rithm School (/) info@rithmschool.com (mailto:info@rithmschool.com)

(https://www.twitter.com/rithmschool) (https://www.facebook.com/rithmschool)
(https://www.linkedin.com/company/rithm-school)