Lesson Tuesday

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Accessing the DOM

Text Cheat sheet

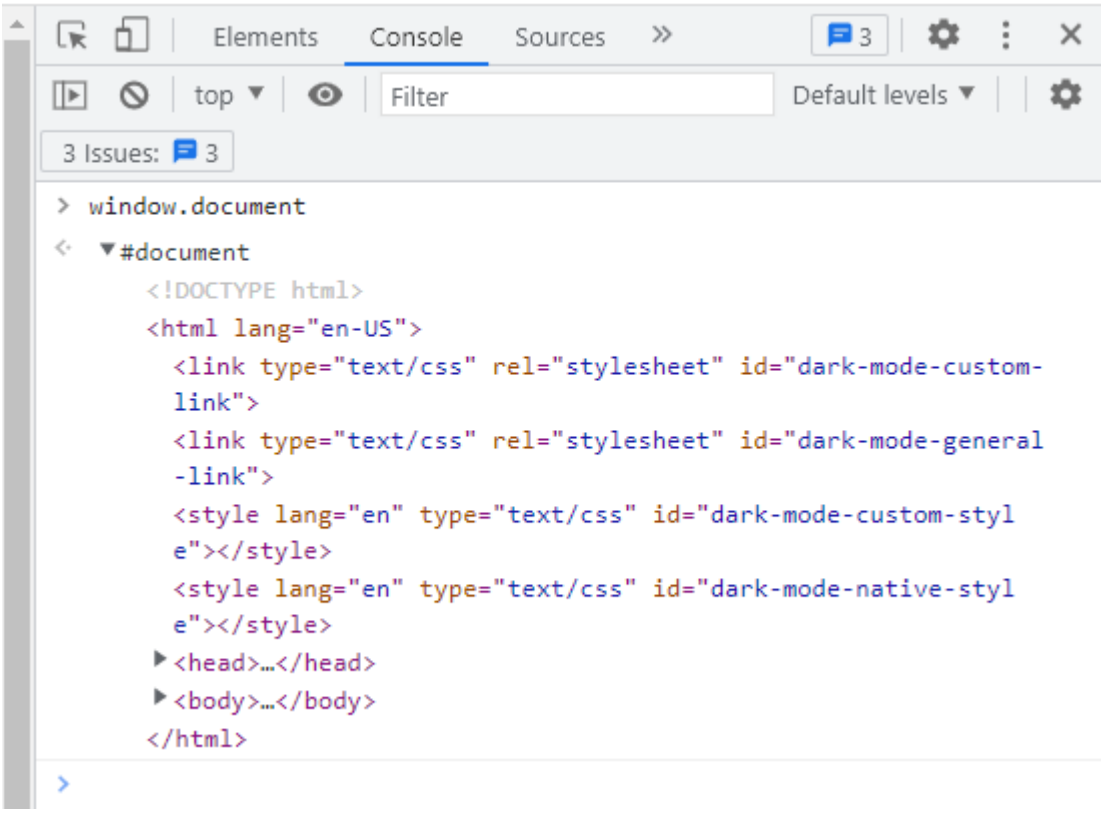
Now we're ready to try accessing the `document` object. Just like with JavaScript objects, the `document` object is made up of properties that we can access to get the DOM's data (like HTML elements), and to modify the DOM (like adding a new HTML element). Remember that we use dot notation to access object properties. As we go along, type out the commands in your DevTools console.

Accessing the DOM

Because `document` is a property of the `window` object, we access it like this:

```
> window.document;
```

You should see returned to you the HTML for the webpage you are on. The next few examples will be from the webpage `https://www.learnhowtoprogram.com/tracks`. When we enter `window.document`, this is what we get returned in the console:



In the image above, we've expanded the returned `#document` object by clicking the triangle to its left. As we can see, the `document` object includes all of the HTML that makes up the webpage we are on, and even though it looks like HTML, we're in fact dealing with an object.

Let's try accessing a few properties of the `document` object. Enter these one at a time into your DevTools console:

```
> window.document.body;
```

```
> window.document.head;
```

What do we get? We get the HTML of our document's `<body>` and `<head>` elements. That's pretty cool! Both of these are objects as well, objects that contain HTML. Make sure that you expand each object in your DevTools console to look inside of them.

Documentation On MDN

Note, while we can access `window.document.body` to get the HTML elements in our `<body>` tags, this doesn't mean we can access `window.document.body.p` to get to a `<p>` tag in our HTML. Instead, we need to use certain `document` methods. Before we move on to that topic, note that you can find all of the properties and methods of the `document` object listed on this reference page in MDN:

- **`document` (https://developer.mozilla.org/en-US/docs/Web/API/Document)**

`document` Methods to Access HTML Elements in the DOM

Let's learn about methods that will help us access specific HTML elements, like a `<p>` tag or an `<h1>`. To help us explore `document` methods, we'll use the cookie recipe from last week. Don't worry — we've already corrected the HTML indentation and spacing for you! For now, just read along with this lesson. In the next lesson, we'll have you clone down the cookie recipe HTML and try it all out yourself.

Here's our cookie recipe HTML (with a few small changes):

cookie-recipe.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> Best Chocolate Chip Cookies </title>
  </head>
  <body>
    <h1 id="specialHeader">Best Chocolate Chip Cookies</h1>
    <img src="https://static01.nyt.com/images/2022/02/12/di
ning/JT-Chocolate-Chip-Cookies/JT-Chocolate-Chip-Cookies-ar
ticleLarge.jpg" alt="An image of a cookie"/>

    <p>This recipe is from my dad and they are a favorite a
mong friends and family. The secret ingredient is the cocon
ut! <em>Be warned</em>, these will fly off of the plate!</p
>

    <h2 class="h2styles">Ingredients</h2>
    <ul class="ul-style">
      <li>Butter</li>
      <li>White sugar</li>
      <li>Brown sugar</li>
      <li>Eggs</li>
      <li>Vanilla</li>
      <li>Flour</li>
      <li>Baking soda</li>
      <li>Salt</li>
      <li>Chocolate chips</li>
      <li>Oatmeal</li>
      <li>Coconut</li>
    </ul>

    <h2 class="h2styles">Directions</h2>
    <ol class="ol-style">
      <li>Preheat the oven to 325.</li>
      <li>Beat the butter, sugar, eggs and vanilla together
until creamy.</li>
      <li>Mix together the flour, baking soda and salt in a
separate bowl.</li>
      <li>Add flour mixture to butter mixture slowly.</li>
      <li>Stir in chocolate chips, oatmeal and coconut.</li
```

```
>
      <li>Bake for <strong>10 minutes</strong> or until gol
den brown.</li>
    </ol>

    <p><a href="http://allrecipes.com">Click here</a> to ch
eck out my other great recipes.</p>
  </body>
</html>
```

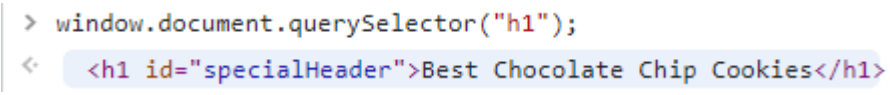
Now, let's run some `document` methods on it!

`window.document.querySelector()`

The first method we'll learn about is `window.document.querySelector()`. This method takes a string as an argument, and the string is the HTML tag that we want to target. With `window.document.querySelector()`, we can use any valid CSS selector as the argument to specify which HTML element we want to access. Here are some examples using the DevTools console:

```
> window.document.querySelector("h1");
```

When we input this into the console, we get this returned:



Since the argument we input to `window.document.querySelector()` works like a CSS selector, we can target the same `<h1>` element by doing this as well:

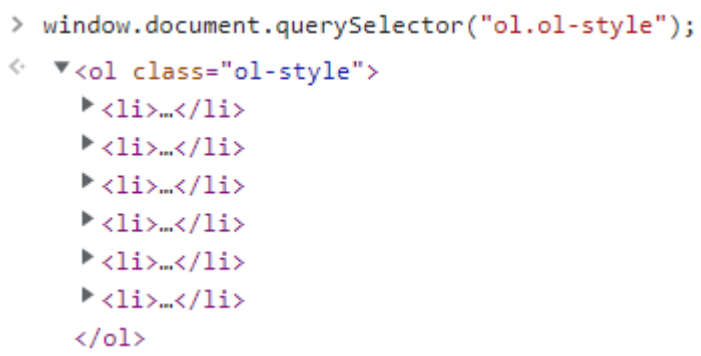
```
> window.document.querySelector("h1#specialHeader");
```

As we can see, the same `<h1>` element is returned to us. And while this `<h1>` element looks like HTML, it's actually an object that is representing the H1 element in our code. We'll circle back to this soon. For now just keep in mind that the DOM is entirely made up of objects, objects that represent HTML elements, the attributes of elements, text, and other data in our webpages.

If we want to target an element using its class name, we'd need to use a `.` to denote that we're targeting a class name, like this:

```
> window.document.querySelector("ol.ol-style");
<ol>...</ol>
```

When we input this into the console, we get this returned:

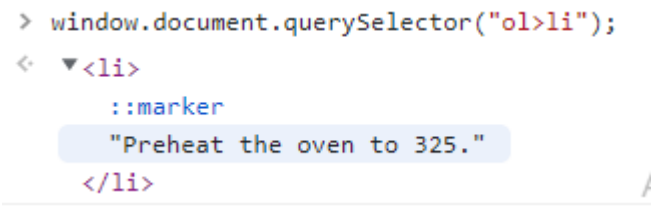


Note that in the image above we've expanded the `<ol>` tags in the DevTools console. When this element is first returned to us it has an ellipses `<ol>...</ol>` indicating that there are nested elements inside of these tags. Anytime we access an HTML element that has elements nested in it, we will be able to expand the returned element in the DevTools console to see what's inside.

Let's try using the angle bracket `>` CSS selector to get children elements of an HTML element, like this:

```
> window.document.querySelector("ol>li");
```

When we input this into the console, we get this returned:



Notice that only the first list element is returned. Any time there are multiple instances of the tag we are targeting, `window.document.querySelector()` will return the first one it finds.

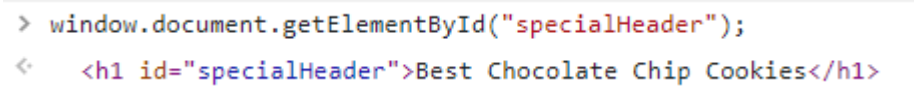
That's the basics of the `window.document.querySelector()` method. Let's look at another method.

`window.document.getElementById()`

The next method that we'll practice with lets us target an HTML element by its `id` attribute, well, the *value* of its `id` attribute. In our sample HTML, we only have one element that has an id, the H1 element. So, let's use that as an example.

```
> window.document.getElementById("specialHeader");
```

When we input this into the console, we get this returned:



Here we can see that the entire H1 element has been returned, which we've successfully targeted using the value of its `id` attribute as a string.

Other `document` Methods to Access HTML Elements in the DOM

There are many other `document` methods that we won't cover in this section, and this is because they return collections of elements, similar to an array. We won't learn about arrays until the next section, "Arrays and Looping". Briefly, an array is a special kind of object that holds a collection of items in it. You might see some of these methods on documentation or in researching online. To give you an idea of what you'll encounter, these are `document` methods that return an array:

- `window.document.querySelectorAll()`
- `window.document.getElementsByClassName()`
- `window.document.getElementsByTagName()`
- `window.document.getElementsByName()`
- `window.document.getElementsByTagName()`

Previous (/introduction-to-programming/javascript-and-web-browsers/understanding-web-apis-the-dom)
Next (/introduction-to-programming/javascript-and-web-browsers/practice-accessing-the-dom)

Lesson 44 of 75
Last updated more than 3 months ago.