Hello, World DOMination: Element Selection And Placement

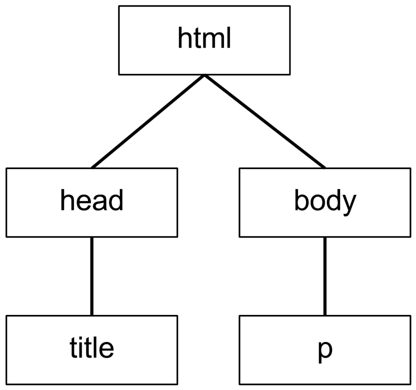
The objective of this lesson is to familiarize yourself with the usage and inner workings of the DOM API. When you finish this lesson, you should be able to:

- Reference and manipulate the DOM via Javascript
- Update and create new DOM elements via Javascript
- Change CSS based on a DOM event
- Familiarize yourself with the console

What is the DOM?

TheDocument Object Model, or DOM, is an object-oriented representation of an HTML document or Web page, meaning that the document is represented as objects, or nodes. It allows developers to access the document via a programming language, like Javascript.

The DOM is typically depicted as a tree with a specific hierarchy. (See the image below.) Higher branches represent parent nodes, while lower branches represent child nodes, or children. More on that later.



Referencing the DOM

The DOM API is one of the most powerful tools frontend developers have at their disposal. Learning how to reference, create, and update DOM elements is an integral part of working with Javascript. We'll start this lesson by learning how to reference a DOM element in Javascript.

Let’s assume we have an HTML file that includes the following `div`:

HTML

```
<div id="”catch-me-if-you-can”">HI!</div>
```

Because we've added the element to our HTML file, that element is available in the DOM for us to reference and manipulate. Using JavaScript, we can reference this element by scanning the document and finding the element by its id with the method document.getElementById(). We then assign the reference to a variable.

Javascript

```
const divOfInterest = document.getElementById(“catch-me-if-you-can”)
```

Now let’s say that our HTML file contains seven `span`elements that share a class name of `cloudy`, like below:

HTML

```
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
<span class="“cloudy”"></span>
```

In Javascript, we can reference all seven of these elements and store them in a single variable.

Javascript

```
const cloudySpans = document.querySelectorAll("span.cloudy");
```

While `getElementById`allows us to reference a single element,`querySelectorAll`references all elements with the class name “cloudy” as a static `NodeList`(*static*meaning that any changes in the DOM do not affect the content of the collection). Note that a NodeList is different from an array, but it is possible to iterate over a NodeList as with an array usingforEach(). Refer to the MDN doc onNodeListfor more information.

Using `forEach()`on a NodeList:

Javascript

```
const cloudySpans = document.querySelectorAll("span.cloudy");

cloudySpans.forEach(span => {
  console.log("Cloudy!");
});
```

Creating New DOM Elements

Now that we know how to reference DOM elements, let's try creating new elements. First we'll set up a basic HTML file with the appropriate structure and include a reference to a Javascript file that exists in the same directory in the`head`.

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script type="text/javascript" src="example.js"></script>
  </head>
  <body></body>
</html>
```

In our example.js file, we'll write a function to create a new `h1`element, assign it an id, give it content, and attach it to the body of our HTML document.

Javascript

```
const addElement = () => {
  // create a new div element
  const newElement = document.createElement("h1");
  // set the h1's id
  newElement.setAttribute("id", "sleeping-giant");

  // and give it some content
  const newContent = document.createTextNode("Jell-O, Burled!");

  // add the text node to the newly created div
  newElement.appendChild(newContent);

  // add the newly created element and its content into the DOM
  document.body.appendChild(newElement);
};
// run script when page is loaded
window.onload = addElement;
```

If we open up our HTML file in a browser, we should now see the words `Jell-O Burled!` on our page. If we use the browser tools to inspect the page (right-click on the page and select "inspect", or hotkeys fn + f12), we notice the new `h1` with the id we gave it.