DOM, jQuery and Web Components

Hunter Jorgensen

What is the DOM?

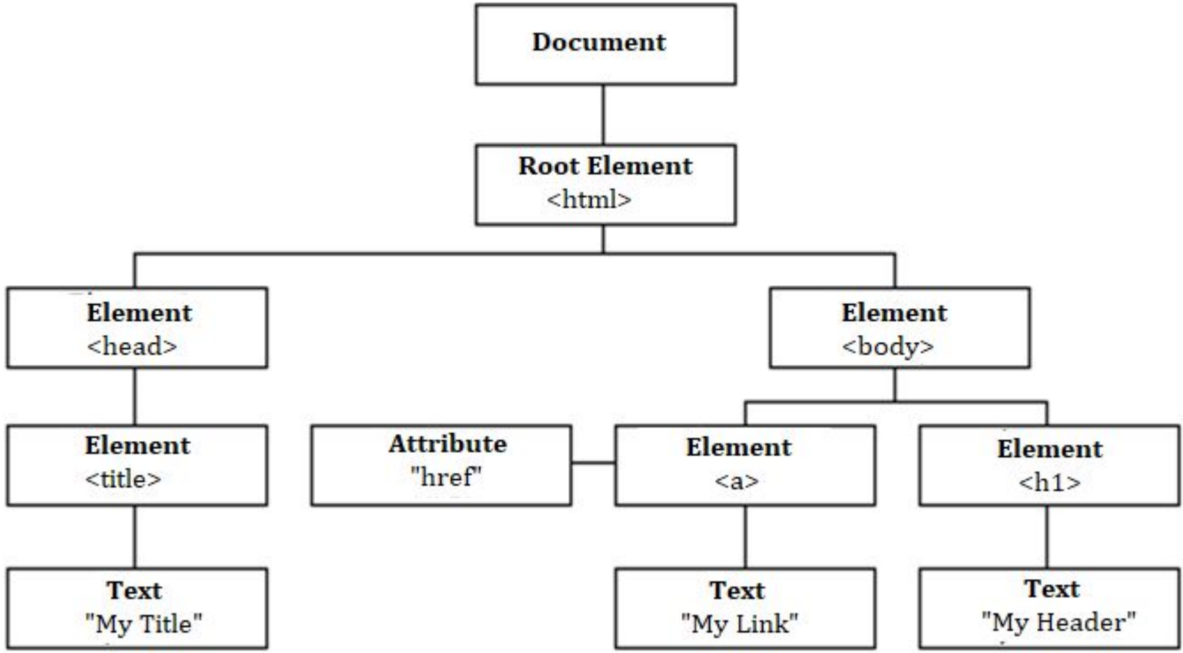
The DOM, or Document Object Model, is an API provided by the browser to allow us to programmatically interact with the content on the page. To access it, you can use the `document` object, as it comes with its own set of APIs.

We won't often access the DOM directly, but other libraries will provide that functionality for us.

Demo

DOM Tree

When accessing the DOM via API, it changes the representation of the document to a tree structure. Nested elements, become ‘children’ to their parents. This is an idea that is picked up in React and other Frameworks.



DOM Document

When you type in `document` in JavaScript, what does this refer to?

This refers to root of the document on the browser. We are given an API to search and modify it (there are more!):

- document.getElementById(id)
- document.getElementsByTagName(name)
- document.createElement(name)
- document.querySelector(name)

Demo

DOM Elements

When we make one of those queries to the browser, the content returned is *usually* an element or list of elements. This refers to the HTML element and includes a reference to the tags, attributes, content and child tags. Some common APIs for the element object are:

- element.innerHTML
- element.style.left
- element.setAttribute()
- element.getAttribute()
- element.addEventListener()

Demo

DOM Window

Finally, there is the DOM `window`, which is commonly seen in modern code. This is a reference to the actual browser itself so that web developers can do things like open popups or close windows:

- window.content
- window.onload
- window.scrollTo()

Demo

DOM Usage

Aside from `window`, accessing the DOM programmatically isn’t particularly common. Instead, we will often use libraries that take care of this work for us!

But it is important to understand what we can programmatically do with JavaScript and how these libraries implement their functionality.

Web Components

All of this comes together with Web Components, which are an API that allow developers to create custom HTML elements or tags.

There are 3 parts of Web Components

Parts of Web Components: Custom Elements

Custom Elements:

- We can create or (if you’re familiar with OO) extend HTML tags
    - (we’ll typically name them with a hypen!)
- We’ll create a custom class
- This class with provide us with lifecycle methods:
    - constructor: called when element is created
    - connectedCallback: called when element is inserted in DOM/page
    - disconnectedCallback: called when element is removed from DOM/page
    - attributedChangedCallback: called when attributes are changes/removed/etc.

Part of Web Components: Shadow DOM

The Shadow DOM is essentially a disconnected, sub-DOM that gives up some cool features

- Can have self contained components
- Encapsulates styles, markup, etc. Can even disconnect from JavaScript of parent
- Called with **element.attachShadow({mode: “open”})**
- This creates a shadowRoot that we can reference and build upon (we can see this with our code inspector!)

Parts of Web Components: HTML Template

- When the page loads, we’ll define the our web component
- We can store things like style and tag organization in something called a “template”
- This is made up of HTML and CSS
- We can use something called “slots” to allow for custom text or children (i.e., nested components)

Web Component Demo

Demo

jQuery

Speaking of libraries, that introduces to jQuery, one of the most important JavaScript libraries! Some of you may even have used it when you were working with Bootstrap.

jQuery was created in 2006 and was initially developed as a way to make it easier to read JavaScript vs HTML. It also gained traction since it worked similarly across multiple browsers, allowing developers to only have to write code a single time.

jQuery is invoked with the `$`

Demo

DOM Manipulation Today

Today, it is increasingly uncommon for developers to manipulate the DOM directly or through libraries such as jQuery. Instead, now libraries like React, AngularIO and Vue.js allow for a more sophisticated mix of interacting with the DOM and creating a website.

Extra Credit: Web Component Dice Roller

Create a Web Component (using the class HTMLElement, template and shadow dom like we do in class.) In this component there should be a button. When you click the button, a number from one to six is randomly shown. The number should also be randomly styled red, green or blue.

Create an image or a gif.

+3 points if you submit the code and image/gif by tomorrow midnight.