MODULE 4

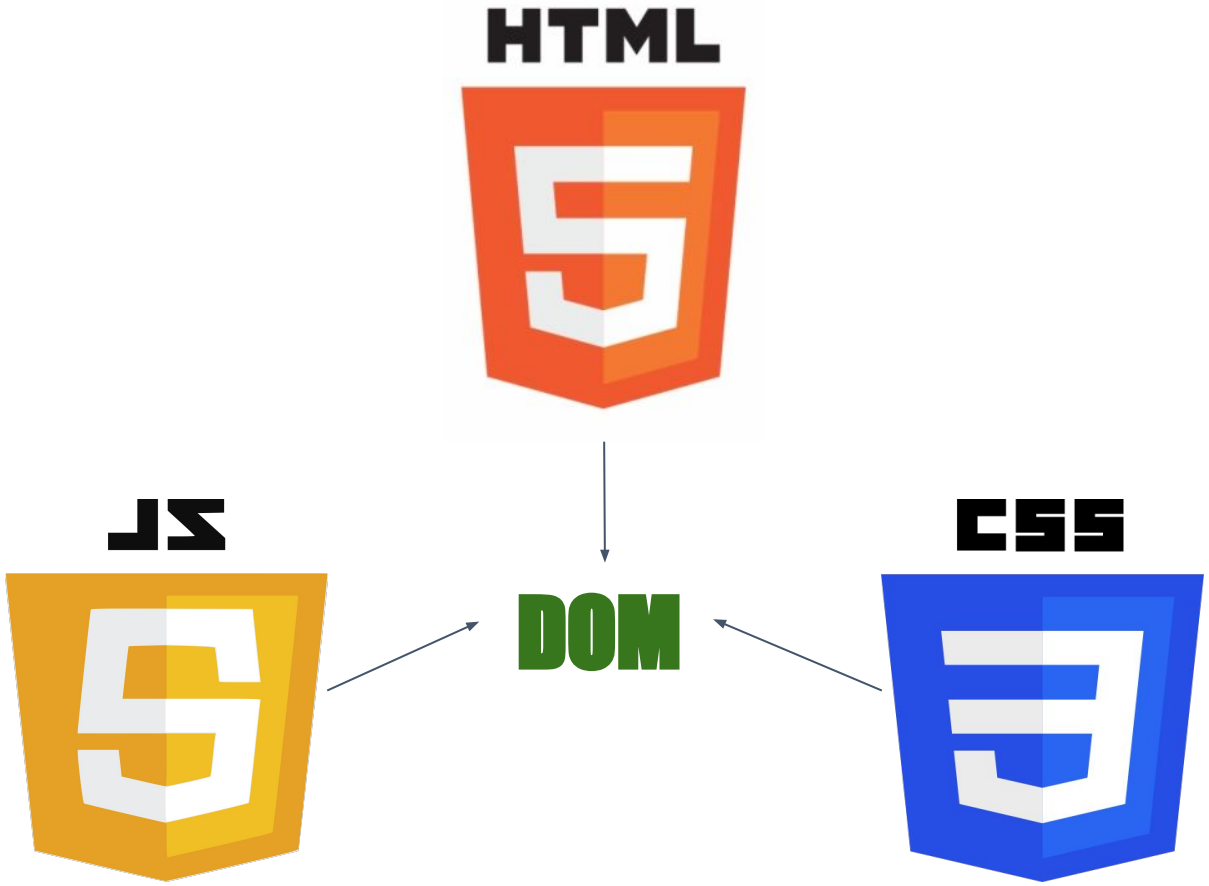
The DOM

Back to HTML

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="utf-8" />
   <title></title>
   <link rel="stylesheet" href="style.css" type="text/css" />
   <script src="jquery-3.1.0.js"></script>
</head>
<body>
   <section class="content">

      <div id="box1" class="red box">
         <p class="text">Text inside a red box</p>
      </div>

      <div id="box2" class="blue box">
         <p class="text">Text inside a blue box</p>
      </div>

   </section>

   <script src="script.js"></script>
</body>
</html>
```

HTML is Static

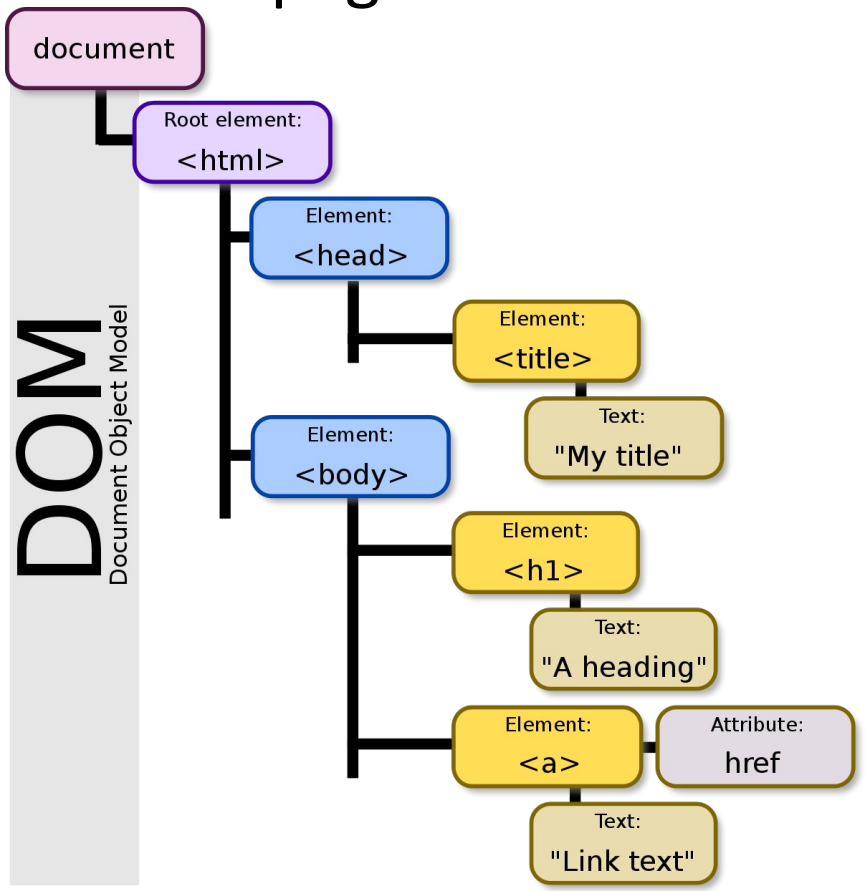
- HTML is read once and turned into the DOM
- CSS and Javascript run against the DOM, not the HTML



Document Object Model (the DOM)

The Document Object Model (DOM) is an **internal, in-memory representation** of a web page's structure, typically stored in RAM as a **nested tree of objects** that represent the elements of the page.

- It is *not* the page source
- It allows developers to:
  - look for an element with JavaScript
  - find an element's parents, siblings, children
  - add/remove css classes via JavaScript
  - add/remove elements from the page
  - manipulate pretty much anything on the page



Vanilla JavaScript

What makes it “vanilla”?

It is JavaScript that does not rely on any outside utility libraries to do things that can be done with functions and object defined in the ECMAScript specification.

Vanilla JS is what we’ll use to interact with the DOM.

DOM Selection Functions

https://book.techelevator.com/content/dom-api-javascript.html#dom-selection-functions

- getElementByID()
  - This function will get a single HTMLElement from the DOM and return a reference to it.
  - https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById
- querySelector()
  - Takes a standard CSS selector and returns the first element it finds that matches that selector
  - https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector
- querySelectorAll()
  - This will return a NodeList of all the elements, which you can use as an array
  - https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll

Selector Review… Here's Just a Few

https://www.w3schools.com/cssref/css_selectors.asp

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements
element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects all <p> elements that are placed immediately after elements
element1~element2	p ~ ul	Selects every element that are preceded by a <p> element

Changing Elements

- innerText
  - Updates any text information on the page
  - All text (including html tags) is replaced!
  - Insert text treated as literals: no interpreting of HTML
- innerHTML
  - Updates any text information on the page
  - All text (including html tags) is replaced!
  - Interprets HTML for display
  - Do not use with user input! (Why? --see demo)

Manipulating Classes

- classList accesses the classes applied to an element

```
// Get the first line item
let firstListItem = document.querySelector('#todos li');

// Add the class `done`
firstListItem.classList.add('done');

// Remove the class `priority`
firstListItem.classList.remove('priority');
```

Traversing the DOM

- .children will select the immediate child elements
- Convert to an array to manipulate
  - map, filter, reduce, etc.
- .childNodes will get all nodes inside an element

- children vs childNodes
  - children are elements, only contain HTML not text
  - childNodes contain HTML and text/values

Traversing the DOM

- .parentNode
  - Gets the immediate parent node of the element

- nextElementSibling
- previousElementSibling

- removeChild
  - Removes the child element from the DOM

LET'S CODE!

ELEVATE YOURSELF

A Quick Aside

Not to be a bore, but there's just a little more,
a final technique for us to critique . . .

JQuery

- Part of the JS Foundation (https://js.foundation)
- Can be installed via CDN (Content Delivery Network)

JQuery Syntax

- Start with bling!!
  - $()
    - Accepts any argument that is a CSS selector and returns a jQuery DOM element(s) that represents that selector

    - Javascript:
    - const element = document.querySelector("#main-content");

    - JQuery
    - const element = $("#main-content");

Manipulating the DOM with JQuery

- The element can have any HTML or CSS property set on it
  - .text()
  - .html()
  - .val()
  - .addClass('name')
  - .removeClass('name')
  - .hasClass('name')

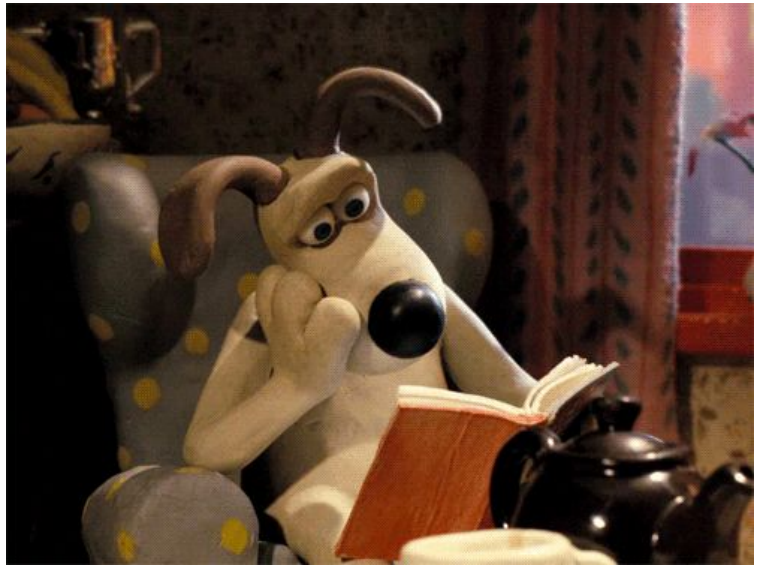
Manipulating the DOM with JQuery

- Creating a DOM element can be done with $(" ")

- Elements can be added/removed from the DOM
  - .append() adds the element as the last child in the jQuery collection (inside)
  - .prepend() adds as the first child in the jQuery collection. (inside)
  - .before() adds before each element in the set of matched elements. (outside)
  - .after() adds after each element in the set of matched elements (outside)
  - .remove() removes matched elements from the DOM
  - .empty() removes all child nodes from the DOM
  - .detach() removes from the DOM to be attached later

WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:



Event Handling