Document Object Model (DOM)

allows you to access elements of a web page & enable interactions w/ the page by adding & removing elements, changing the order, content & attributes of elements & even altering how they are styled

What is the DOM?

- represents an HTML document as a network of connected nodes that form a tree-like structure
- treats everything on a webpage as a node
  - HTML tag is the root node
  - every other element is a child of this
- not actually a part of JS bc. it is language agnostic
  - it can be used in any programming language, not just JS

History of the DOM

- web browsers developed their own ways of accessing & altering parts of a webpage

"In the beginning, they tended to focus on common page elements such as images, links and forms – this was known as Dynamic HTML (DHTML). These methods became known as DOM level 0, or legacy DOM"

- W3C started to standardize the process & created the DOM level 1 in 1998
- DOM level 2 specification was published in 2000
  - brought w/ it the `getElementById()` method
- DOM level 3 came in 2004 & since then W3C has stopped using levels

Getting Elements

DOM provides methods that allow access to any element on a page & these methods will return a node object or a node list, which is an array-like object

Legacy DOM Shortcut Methods

methods from DOM level 0 that can still be employed to access common elements

- `Document.body`
- `Document.images`
- `Document.links`
- `Document.anchors`
- `Document.forms`

Getting an Element By Its ID

- `getElementById()`
- does exactly what it says on the tin
- returns a reference to the element w/ a unique `id` attribute that is given as an argument

Every ID attribute should be unique to just one element

- odd things can happen in your code if you have more than one element with the same ID

- its a quick way of finding elements in a document

Get Elements By Their Tag Name

- getElementsByTagName()

-return a live node of all elements with the tag name that is provided as an argument

Get Elements By Their Class Name

- getElementsByClassName()

- return a live node list of all elements that have the class name that is supplied as an argument

Query Selectors

- document.querySelector()

-allows you to use CSS notation to find the first element in the doc that matches a CSS selector

- document.querySelectorAll()

- returns node list of all elements in doc that match the CSS query selector

Navigating the DOM Tree

- Node objects have a number of properties & methods for navigating around the doc tree
- childNodes property is a list of all the nodes that are children of the node concerned
- childNodes property returns all the nodes that are children of an element
- children property only returns any element nodes that are children of that node
- firstChild property returns the 1st child of a node
- lastChild property returns the last child of a node
- parentNode property returns the parent node of an element
- nextSibling property returns the next adjacent node of the same parent
- previousSibling property returns the previous adjacent node

Finding the Value of a Node

- we can find the text contained inside it using the nodeValue method
- can also find this value using the textContent property

Getting & Setting Attributes

Getting An Element's Attributes

- getAttribute() returns the value of the attribute provided as an argument

Setting An Element's Attributes

- setAttribute() can change the value of an element's attributes
- takes two arguments:
  - the attribute you wish to change
  - the new value of that attribute
- if an element does not have an attribute, the setAttribute method can be used to add it to the element

Classes of An Element

The ClassName Property

- className property that allows the class of an element to be set directly
- can be used to find out the value of the class attribute

The ClassList Property

- classList property is a list of all the classes an element has
- add method can be used to add a class to an element w/out overwriting any classes that already exist
- remove method will remove a specific class from an element
- toggle method is a particularly useful method that will add a class if an element doesn't have it already, & remove the class if it does have it →
  → it returns true if the class was added & false if it was removed
- contains method will check to see if an element has a particular class

Creating Dynamic Markup

Creating An Element

- doc. object has a createElement() method that takes a tag name as a parameter & returns that element

Creating a Text Node

"text node can be created using the document.createTextNode() method. It takes a parameter, which is a string containing the text that goes in the node"

Appending Nodes

- appendChild() method that will add another node (given as an argument) as a child node

"the process to follow each time you want to create a new element with text content is this:

1. Create the element node
2. Create the text node
3. 'Append the text node to the element node

- made simpler by using the textContent property that every element object has

A Function To Create Elements

```
function createElement (tag,text) {
const el = document.createElement(tag);
el.textContent = text;
return el
}
```

Adding Elements to the Page

- appendChild() method can be called on a node to add a new child node
- insertBefore()

insertBefore() method will place a new element before another element in the markup. It's important to note that this method is called on the parent node. It takes two parameters: the first is the new node to

be added, and the second is the node that you want it to go before

```
heroes.insertBefore(aquaman, wonderWoman);
```

there is no insertAfter() method, so you need to ensure you have access to the correct elements to place an element exactly where you want it.

Removing Elements From A Page

- element can be removed from a page using the removeChild()

Replacing Elements on a Page

- replaceChild() method can be used to replace one node w/ another
  - two parameters
    1) the new node
    2) the node to be replaced

innerHTML

- innerHTML element property returns all the child elements of an element as a string of HTML
- also writable & can be used to place a chunk of HTML inside an element
- saves you having to create a new text node as its done automatically and inserted into the DOM
- when inserting a large amount of HTML into a doc. you can enter the raw HTML as a string. The relevant nodes will then be added to the DOM tree automatically

Live Collections

"document.getElementsByClassName() and document.getElementsByTagName() methods are live collections that will update to reflect any changes on the"
"the node list updates automatically without having to make another call to the method. Therefore, its use is discouraged for performance reasons, but it can be useful"

Updating CSS

"Every element node has a style property. This can be used to dynamically modify the presentation of any element on a web page"

Camel Case Properties

"Any CSS property names that are separated by dashes must be written in camelCase notation, so the dash is removed and the next letter is capitalized because dashes are not legal characters in property names"
"CSS property background-color becomes backgroundColor"
"the bracket notation that we saw in chapter 5 can also be used,"

Disappearing Act

"display property. This can be used to make elements disappear and reappear on the page as needed"
"The element can be made to 'reappear' by changing the display property back to block:"

Checking Style Properties

“getComputedStyle() that will retrieve all the style information of an element that is given as a parameter. This is a read-only property, so is only used for finding out information about the style of an element.”

“returns an object (more specifically, it is a CSSStyleDeclaration object) that contains a list of property-value pairs of all the CSS styles that have been applied to the element in question”

Use with caution

“it is much better practice to dynamically change the class of an element and keep the relevant styles for each class in a separate stylesheet”

Chapter Summary

“The Document Object Model is a way of representing a page of HTML as a tree of nodes.

The document.getElementById(), document.getElementsByClassName(), document.getElementsByTagNames() and document.querySelector() can be used to access elements on a page.

The parentNode(), previousSibling(), nextSibling(), childNodes() and children() methods can be used to navigate around the DOM tree.

An element’s attributes can be accessed using the getAttribute() method, and updated using the setAttribute() method.

The createElement() and createTextNode() methods can be used to create dynamic markup on the fly.

Markup can be added to the page using the appendChild() and insertBefore() methods.

Elements can be replaced using the replaceChild() method, and removed using the removeChild() method.

The innerHTML property can be used to insert raw HTML directly into the DOM.

The CSS properties of an element can be changed by accessing the style property”