The DOM
(Document Object Model)

Today's Objectives

- **What is the DOM?**
- **How is the DOM different from HTML?**
- **Selecting DOM elements in JavaScript**
  - `getElementById()`
  - `querySelector()`
  - `querySelectorAll()`
- **DOM Structure**
- **Modify HTML using `innerText` (but NOT `innerHTML`)**
- **Creating DOM elements in JavaScript**
  - `createElement()`
  - `insertAdjacentElement()`
- **Traversing the DOM**
- **Viewing the living DOM**

What is the Document Object Model (DOM)?

The Document Object Model (**DOM**) is the programming interface for the graphical representation of documents that are loaded into the browser.

It provides a way to access and manipulate the structure, style, and content once loaded.

The DOM is not HTML

- The DOM is not the same as the HTML you see in the html document.
- It is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript. The representation is organized in a tree structure.
- **`document`** represents root object of the HTML document.

Selecting DOM Elements

- We can select DOM elements in JavaScript using one of several methods:
  - **`document.getElementById('element-id')`**
    - Most efficient
    - Returns first (should be only) element matching **`id`**
    - Called only through **`document`**

Selecting DOM Elements

- We can select DOM elements in JavaScript using one of several methods:
  - **`querySelector('css-selector')`**
    - Returns first element matching the selector
    - Called through any element to narrow the search

Selecting DOM Elements

- We can select DOM elements in JavaScript using one of several methods:
  - **`querySelectorAll`**('**`css-selector`**')
    - Returns NodeList containing all matching elements
    - Called through any element to narrow the search

Other Get Methods

- `getElementsByClassName('class-name')`
- `getElementsByName('element-name')`
- `getElementsByTagName('tag-name')`
- These return 'live' lists

Let's Modify the DOM!

Selecting the Text of an Element

- We can set the text within an HTML element using **`innerText`** property of the element.

```
function setPageTitle() {
  let pageTitle = document.getElementById('page-title');
  pageTitle.querySelector('.name').innerText = name;
}
```

Avoid Using innerHTML!!!

Anything passed to **innerHTML** will be read and rendered into the living DOM of the browser. That **could be really dangerous!** If you ever take input from a user and then use innerHTML to put that into an element, you're setting yourself up for what is called a **Cross Site Scripting Attack (XSS).**

If a user is able to add HTML to your page, that means that they can embed JavaScript into your page using a <script> element and that means they can use all these methods to completely rewrite your page, including making it look like a login page that sends usernames and passwords to their own site instead of yours.

Never, ever send user inputted data to an innerHTML call. When taking user input, always use **innerText** to add their content to a DOM element.

Modifying Elements - Properties

Property	Description
innerText / innerHTML	Gets or sets the text inside the node. innerText is safe; innerHTML is susceptible to injection attack.
value	Gets or sets the value of most input elements
checked	Gets or sets the Boolean state of a checkbox
classList	Gets a collection of the classes applied to the element. Use .add() or .remove() to change the classes on an element.
children / childNodes	Gets a collection of this element's child elements, or child nodes, respectively. children is *usually* what you want; childNodes include text, comments and other nodes that you are usually not interested in.
parentNode	Gets the element to which this element belongs (is in the parent's children collection)
nextElementSibling / previousElementSibling	Gets to the next/previous element with the same parent

Let's Create Some Elements!

Create and Add an Element

- Can create an HTML element using

  `document.createElement('tag-name')`

Create and Add an Element

- Can create an HTML element using

  `document.createElement('tag-name')`
- Can insert the element using

  `insertAdjacentElement('insert-location',element)`
  - `beforebegin`
  - `afternbegin`
  - `beforeend`
  - `afterend`

Append a Child to an Element

- Can append a child to an element using **`appendChild(element)`**

```
function addReviewer(parent,name) {
    const reviewer = document.createElement('h4');
    reviewer.innerText = name;
    parent.appendChild(reviewer);
}
```

Removing an Element

```
// Find the element
let ele = document.querySelector('css-selector');

// Remove the element from its parent
ele.parentNode.removeChild(ele);
```

An Easier Way...