JavaScript:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Variables:

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables>

Math Operator:

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Math>

Functions:

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Functions>

Arrays

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays>

Object Basics

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics>

VSCode

<https://code.visualstudio.com/docs>

<https://code.visualstudio.com/docs/getstarted/keybindings>

<https://code.visualstudio.com/docs/getstarted/settings>

<https://code.visualstudio.com/docs/editor/extension-gallery>

Chrome Developer Tools

<https://developers.google.com/web/tools/chrome-devtools/>

Precedence

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence>

Statement Vs Expression

<https://stackoverflow.com/questions/12703214/javascript-difference-between-a-statement-and-an-expression>

Flow Control and Error Handling

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling>

Loops and Iteration

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration>

Strict Mode

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode#Changes_in_strict_mode>

Jet Engine Explained

<https://hackernoon.com/javascript-v8-engine-explained-3f940148d4>

Spider Monkey

<https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey/Internals>

Memory Management

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management>

Garbage Collection

<https://v8.dev/blog/free-garbage-collection>

Functions

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>

Bind

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_objects/Function/bind>

QuerySelectorAll

<https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll>

Element.getElementsByTagName()

<https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByTagName>

DOM

<https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>

ARRAYS

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

Summaries:

Summary: Insert, Replace, Remove

There are many ways of creating, inserting, replacing and removing DOM elements - here's a summary of the options you have.

For browser support, check the provided links and also the "Browser Support" module you find later in the course.

#### Create & Insert

You got two main options: Provide an HTML snippet (e.g. via innerHTML) to a valid HTML snippet and let the browser render it OR create a DOM object in JS code and append/ insert it manually. The latter approach has the advantage of giving you direct access to the DOM object (useful for setting its properties or adding event listeners). The downside is that you have to write more code.

**Adding HTML Code:**

1. const root = document.getElementById('root-el'); // selects something like <div id="root-el">
2. root.innerHTML = `
3. <div>
4. <h2>Welcome!</h2>
5. <p>This is all create & rendered automatically!</p>
6. </div>
7. `;

**Important**: Any existing content in root is  completely replaced when using innerHTML. If you want to append/ insert HTML code, you can use insertAdjacentHTML instead: <https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentHTML>

1. const root = document.getElementById('root-el'); // selects something like <div id="root-el">
2. root.insertAdjacentHTML('afterbegin', `
3. <div>
4. <h2>Welcome!</h2>
5. <p>This is all create & rendered automatically!</p>
6. </div>
7. `);

**Creating & Inserting DOM Objects Manually:**

1. const someParagraph = document.createElement('p'); // creates a "p" element (i.e. a <p> element)
2. const root = document.getElementById('root-el'); // selects something like <div id="root-el">
3. root.append(someParagraph);

In this example, we create a paragraph and append it to root - append means that it's inserted at the end of root (i.e. inside of it but AFTER all other child nodes it holds).

**Insertion Methods:**

**append()** => <https://developer.mozilla.org/en-US/docs/Web/API/ParentNode/append>

Browser support is decent but for IE, **appendChild()** could be preferred => <https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild>

**prepend()** => <https://developer.mozilla.org/en-US/docs/Web/API/ParentNode/prepend>

Browser support is decent but for IE, **insertBefore()** could be preferred => <https://developer.mozilla.org/en-US/docs/Web/API/Node/insertBefore>

**before()**, **after()** => <https://developer.mozilla.org/en-US/docs/Web/API/ChildNode/before> & <https://developer.mozilla.org/en-US/docs/Web/API/ChildNode/after>

Browser support is okay but IE and Safari don't support it. Consider **insertBefore()** (<https://developer.mozilla.org/en-US/docs/Web/API/Node/insertBefore>) or **insertAdjacentElement()** (<https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentElement>) as substitutes.

Important (no matter how you insert elements): Whenever you insert elements, you **MOVE the element** to that new place if you already inserted it before. It's **NOT copied** (you can copy an element via someElement.cloneNode(true) though).

#### Replace

You can replace elements in the DOM with two main methods:

**replaceWith()** => <https://developer.mozilla.org/en-US/docs/Web/API/ChildNode/replaceWith>

**replaceChild()** => <https://developer.mozilla.org/en-US/docs/Web/API/Node/replaceChild>

replaceWith() is a bit easier to use and has decent browser support - with IE being the exception. To support that as well, consider using replaceChild().

#### Remove

You can remove elements with three main methods:

**someElement.innerHTML = ''** => Clears all HTML content of someElement and hence removes any objects rendered in there.

**someElement.remove()** => Removes a single element (someElement) from the DOM (<https://developer.mozilla.org/en-US/docs/Web/API/ChildNode/remove>). Browser support is good, IE again doesn't like it though. Use removeChild (see below) instead.

**someElement.parentNode.removeChild(someElement)** =>  Removes the provided child element (NOT the element on which you call it). Provides broad browser support but of course requires a bit more code (<https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild>).

#### What about Text Nodes?

You can easily create & insert text nodes in one go:

1. someElement.textContent = 'Hi there!';

This creates and inserts the text node with a content of 'Hi there!'.

**Want to append to existing text?**

Just use:

1. someElement.textContent = someElement.textContent + 'More text!';