Task - JavaScript and the DOM

Objectives

* To be able to manipulate the DOM by creating and adding content.

Activity

1. In VSCode, create a HTML page with script tag.
2. Within the HTML, create a div with id as "placeholder".
3. In JavaScript, create an arrays called "colours" with the following elements:

'red', 'green', 'blue', 'orange', 'lime', 'olive', 'black', 'yellow'.

1. Under the array declaration, create a function called buildP that takes 2 parameters:
   1. One called placeholder;
   2. One called num.

The parameter placeholder will hold a reference to the DOM element we want to append new content to. The DOM programming mechanism allows us to create new elements by type using the createElement function. Any new elements have to be added to the DOM or it will not display on the page and it will not have any content, unless we give it some! When DOM programming, it is important to work in the correct order:

* + 1. Create the new element;
    2. Create any text nodes that need to be attached to the new element;
    3. Attach the text nodes to the new element;
    4. Attach the new element to the appropriate existing DOM element;

1. Inside the function declare a variable called p and set it to a new element of type p using the createElement function:

document.createElement('p')

1. Create a text node to attach to the new paragraph called text with any string you want in it using the line of code below (the text given is an example):

document.createTextNode(`Have you tried turning it off and back on again?`)

1. Append the text node to the paragraph p using the appendChild function.
2. Append the paragraph p to the placeholder that was passed into the function.

The function is now complete and ready to go. All we need to do is call it and pass in the correct parameters. The new <p> element we are going to create needs to be appended to a container in the page. This could be another element on the page or even the document root itself. We will add it to the <div id=”placeholder”> found in index.html.

1. Under the function definition, call the function setting the placeholder argument with the element found by using:

document.querySelector('#placeholder')

When we created the buildP function, we added a second parameter to the function that we have not used, or even set in the function so far. This is one of JavaScript’s nice features. You can either add none, the first, or all parameters.

1. Save your code and refer to your browser.

You should see that the text and paragraph you created with JavaScript are now part of the page. Verify this by inspecting the elements in the Developer Tools.

The second parameter of the buildP function is going to be used to create and append multiple paragraph elements. We will always want at least 1 paragraph but maybe more - time to use the do…while loop.

1. At the top of the buildP function, declare a variable i initialised to 0.
2. Surround the rest of the code in the buildP function with a do…while loop that exits when i reaches the number supplied to the function (remember to increment i).
3. Amend the call to the buildP function to add a second parameter of any number you choose.
4. Save and refer to your browser, checking the output is as expected.
5. Try a few different values and observe the page.

The final piece in this part of the exercise is to randomise the colours used to display our text in. To do this, we will use the array of colours already defined and a randomly generated value in the range of the indexes of the array.

Math.random gives us a value somewhere between 0 and 1. Indexes of arrays are integer values as are their length. The parseInt function will be used to return the nearest integer to a value we are going to calculate using the randomly generated number and the length of the colours array.

1. Before the line that appends the text node to the paragraph add the following line:

p.style.color = colours[parseInt(Math.random() \* colours.length)]

1. Save and refer to the output of the browser.

This is the end of Task