JavaScript

HTML DOM: -

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects.

With the object model, JavaScript gets all the power it needs to create dynamic HTML.

JavaScript can Modify HTML elements and attributes, Change CSS styles, Remove or add HTML elements and attributes, Handle existing HTML events.

HTML DOM Method: -

HTML DOM methods are actions you can perform (on HTML Elements).

The getElementById Method: -

The most common way to access an HTML element is to use the id of the element.

```
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

The innerHTML Property: -

The innerHTML property is useful for getting or replacing the content of HTML elements.

The HTML DOM Document Object: -

The document object represents your web page. If you want to access any element in an HTML page, you always start with accessing the document object.

Finding HTML Elements: -

Method	**Description**
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

Changing HTML Elements: -

Property	**Description**
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

Adding and Deleting Elements: -

Method	**Description**
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

Adding Events Handlers: -

Method	**Description**
document.getElementById(id).onclick = function(){code}	Adding event handler code to an onclick event

Finding HTML Elements: -

Finding HTML Element by Id: -

const element = document.getElementById("intro");

If the element is found, the method will return the element as an object (in element). If the element is not found, element will contain null.

Finding HTML Elements by Tag Name: -

This example finds all <p> elements

const element = document.getElementsByTagName("p");

If there are multiple paragraphs, then element variable will become an array and can be accessed through index.

This example finds the element with id="main", and then finds all <p> elements inside "main"

```
const x = document.getElementById("main");
const y = x.getElementsByTagName("p");
```

Finding HTML Elements by Class Name: -

```
const x = document.getElementsByClassName("intro");
```

This example returns a list of all elements with class="intro"

Finding HTML Elements by CSS Selectors: -

If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

```
const x = document.querySelectorAll("p.intro");
```

This example returns a list of all <p> elements with class="intro".

Finding HTML Elements by HTML Object Collections: -

```
const x = document.forms["frm1"];
let text = "";
for (let i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

This example finds the form element with id="frm1", in the forms collection, and displays all element values.

HTML DOM - Changing HTML: -

The easiest way to modify the content of an HTML element is by using the innerHTML property.

```
document.getElementById(id).innerHTML = new HTML
```

```
<!DOCTYPE html>

<html>

<body>

<p id="p1">Hello World!</p>

<script>

document.getElementById("p1").innerHTML = "New text!";

</script>

</body>

</html>
```

Changing the Value of an Attribute: -

```
document.getElementById(id).attribute = new value.
```

```
<!DOCTYPE html>
<html>
<body>
<img id="myImage" src="smiley.gif">
<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
</body>
</html>
```

Dynamic HTML content: -

```
<script>
document.getElementById("demo").innerHTML = "Date : " + Date();
 </script>
```

document.write(): -

In JavaScript, document.write() can be used to write directly to the HTML output stream.

Never use document.write() after the document is loaded. It will overwrite the document.

JavaScript Forms: -

JavaScript Form Validation: -

HTML form validation can be done by JavaScript. If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted.

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

The function can be called when the form is submitted.

JavaScript Can also validate Numeric Input.

Automatic HTML Form Validation: -

If a form field (fname) is empty, the required attribute prevents this form from being submitted.

Data Validation: -

Data validation is the process of ensuring that user input is clean, correct, and useful. Check if user has filled in all required fields? has the user entered a valid date? has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct user input.

Validation can be defined by many different methods and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.

Client side validation is performed by a web browser, before input is sent to a web server.

HTML Constraint Validation: -

Constraint Validation HTML Input Attributes: -

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

Constraint Validation CSS Pseudo Selectors: -

Selector	Description
:disabled	Selects input elements with the "disabled" attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no "required" attribute specified
:required	Selects input elements with the "required" attribute specified
:valid	Selects input elements with valid values

HTML DOM - Changing CSS: -

Changing HTML Style: -

document.getElementById(id).style.property = new style

document.getElementById("p2").style.color = "blue";

// changes text colour of paragraph with id ‘p2’ to blue.

Using Events: -

The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements:

An element is clicked on, The page has loaded, Input fields are changed.

For example, by clicking a button, you can change the text colour.

Pattern Problem – 2: -

Generate a pattern for any n . below is example for n =6.

					1
				2	4
			3	5	7
		6	8	10	12
	9	11	13	15	17
14	16	18	20	22	24

Method 1: -

```
function generatePattern(n) {
    let num1 = 1,num2 = 2;
    let result = [];
    for (let i = 1; i <= n; i++) {
        let row = [];
        for (let j = 1; j <= n - i; j++) {
            row.push(' ');
        }
        for (let j = 1; j <= i; j++) {
            if(i % 2 == 0){
                row.push(num2);
                num2+=2;
            }
            else{
                row.push(num1);
                num1+=2;
            }
        }
        result.push(row.join(' '));
    }
    result.forEach(row => console.log(row));
```

```
}
generatePattern(6);
```

Time Complexity is $O(n^2)$

Space Complexity is $O(n^2)$

Method 2: -

```
function generatePattern(n) {
    let num1 = 1, num2 = 2;
    for (let i = 1; i <= n; i++) {
        for (let j = 1; j <= n - i; j++) {
            if(j === n)
                process.stdout.write(' ');
            else
                process.stdout.write(' ' + ' ') ;
        }
        for (let j = 1; j <= i; j++) {
            if(i % 2 == 0){
                process.stdout.write(num2 + ' ');
                num2+=2;
            }
            else{
                process.stdout.write(num1 + ' ');
                num1+=2;
            }
        }
        console.log();
    }
}
generatePattern(6);
```

Time Complexity: $O(n^2)$

Space Complexity: O(1)