Unit 3: Client Side Programming with JavaScript

How JavaScript fits into a web page?

JavaScript is a high-level programming language that is widely used for client-side web development. It can be used to create dynamic and interactive user interfaces, as well as to add interactivity and functionality to web pages. JavaScript is supported by all major web browsers and is often used in combination with HTML and CSS.

JavaScript Basics

Variable

Variables are used to store data in JavaScript. There are three ways to declare variables in JavaScript:

***var keyword**

The var keyword is used to declare a variable in JavaScript. Here's an example:

```
var x = 10;
```

In this example, we've declared a variable x and assigned the value 10 to it using the = operator. The var keyword can be used to declare a variable anywhere in the code, and the variable can be reassigned a new value at any time.

***let keyword**

The let keyword is used to declare a block-scoped variable in JavaScript. Here's an example:

```
let x = 10;
```

In this example, we've declared a variable x and assigned the value 10 to it using the = operator. The let keyword is block-scoped, which means that the variable is only accessible within the block it was declared in. The variable can be reassigned a new value at any time.

Operators

JavaScript has a number of operators that are used to perform different operations on values. Here are some common operators in JavaScript:

i. Arithmetic operators

Arithmetic operators are used to perform mathematical operations on values. The most common arithmetic operators in JavaScript are + (addition), - (subtraction), *(multiplication), / (division), and % (modulus).

Example:

```
let x = 10;

let y = 5;

let sum = x + y; // sum will be 15

let difference = x - y; // difference will be 5

let product = x * y; // product will be 50

let quotient = x / y; // quotient will be 2

let remainder = x % y; // remainder will be 0
```

ii. Assignment operators

Assignment operators are used to assign values to variables. The most common assignment operator in JavaScript is = (assignment). Example:

let x = 10;

x += 5; // equivalent to x = x + 5, x will be 15.

iii. Comparison operators

Comparison operators are used to compare values and return a Boolean value (true or false). The most common comparison operators in JavaScript are == (equality), != (inequality), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to). Example:
let x = 10;

let y = 5;

let isEqual = x == y; // isEqual will be false

let isGreater = x > y; // isGreater will be true

iv. Logical operators

Logical operators are used to perform logical operations on Boolean values. The most common logical operators in JavaScript are && (logical and), || (logical or), and ! (logical not). Example:

let x = 10;

let y = 5;

let isBothTrue = (x > 5) && (y > 5); // isBothTrue will be false

let isEitherTrue = (x > 5) || (y > 5); // isEitherTrue will be true.

Understanding the Document Object Model (DOM)

The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects, allowing a programming language like JavaScript to manipulate the content and structure of a web page. In simpler terms, the DOM is a way for web developers to interact with the structure and content of a web page using code. It's a programming interface that lets you dynamically change the content of a web page without having to reload the entire page. The DOM is organized as a tree-like structure, where each element on a web page is represented as a node in the tree. Each node in the tree can have child nodes (representing elements nested within the parent element) and/or sibling nodes (representing elements that are on the same level in the tree). Here's an example of the DOM structure for a simple HTML document.

```
<html>
 <head>
 <title>My Web Page</title>
 </head>
 <body>
<h1>Hello, World!</h1>
 <p>This is a paragraph.</p>
 <ul>
 <li>List item 1</li>
 <li>List item 2</li>
 <li>List item 3</li>
 </ul>
 </body>
</html>
```

By using JavaScript to manipulate the DOM, you can dynamically change the content and structure of a web page. For example, you could use JavaScript to add a new element to the page, change the text of an existing element, or remove an element from the page. The possibilities are endless, and the DOM gives you the power to create dynamic and interactive web pages that respond to user input and behavior.

****{Accessing HTML Elements with getElementById(),getElementsByClassName(), getElementsByName(), getElementsByTagName()}****

There are several methods in JavaScript that allow you to access HTML elements within a web page. Here are some of the most common methods for accessing elements in the DOM:

****getElementById()**

The getElementById() method is used to get a reference to an HTML element using its ID. Here's an example:

```
// Get a reference to an element with ID
"myElement" const myElement= document.getElementById("myElement");
// Change the text of the element
myElement.textContent = "This is my element.";
```

In this example, we've used the getElementById() method to get a reference to an HTML element with an ID of "myElement". We can then use the returned element reference to manipulate the element's properties, such as changing the text content.

****getElementsByClassName()**

The getElementsByClassName() method is used to get a collection of HTML elements using their class name. Here's an example:

```
// Get a collection of elements with class
"myClass" const myElements= document.getElementsByClassName("myClass");
// Loop through the elements and change their text
```

for (let i = 0; i < myElements.length; i++)

{ myElements[i].textContent = "This is my class element."; }
In this example, we've used the getElementsByClassName() method to get a collection of HTML elements with a class name of "myClass". We can then loop through the returned collection and manipulate each element's properties, such as changing the text content.

****getElementsByName()**

The getElementsByName() method is used to get a collection of HTML elements using their name attribute. Here's an example:
// Get a collection of elements with name "myName"

const myElements = document.getElementsByName("myName");

// Loop through the elements and change their text

for (let i = 0; i < myElements.length; i++)

{ myElements[i].textContent = "This is my name element."; }

In this example, we've used the getElementsByName() method to get a collection of HTML elements with a name attribute of "myName". We can then loop through the returned collection and manipulate each element's properties, such as changing the text content.

****getElementsByTagName()**

The getElementsByTagName() method is used to get a collection of HTML elements using their tag name. Here's an example:
// Get a collection of elements with tag name "p"

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

// Loop through the elements and change their text

for (let i = 0; i < myElements.length; i++)

{ myElements[i].textContent = "This is my paragraph element."; }
In this example, we've used the getElementsByTagName() method to get a collection of HTML elements with a tag name of "p" (paragraph). We can then loop through the returned collection and manipulate each element's properties, such as changing the text content.

JavaScript objects: window, document, array, string, math,date

JavaScript objects are a way to store and manipulate data in a structured way. An object is a collection of properties, where each property consists of a key and a value. The key is a string that identifies the property, and the value can be any JavaScript data type, such as a number, string, boolean, array, or even another object. JavaScript has several built-in objects that you can use in your code. Here are some examples of built-in objects in JavaScript:

Window Object

The window object is the top-level object in the browser's JavaScript hierarchy, and it represents the browser window or tab that the JavaScript code is running in. The window object provides access to many browser-specific properties and methods, such as the alert() method, which displays an alert dialog box. Here's an example:

```
window.alert("Hello, World!");
```

In this example, we're using the alert() method to display a message in an alert dialog box.

Document Object

The document object represents the web page that the JavaScript code is running in, and it provides access to the HTML elements and other content on the page. The document object has many properties and methods for manipulating the content and structure of the web page, such as the getElementById() method, which returns a reference to an HTML element with a specified ID. Here's an example:

```
const myElement = document.getElementById("myElement");

myElement.textContent = "This is my element.";
```

In this example, we're using the getElementById() method to get a reference to an HTML element with an ID of "myElement", and then we're using the textContent property to change the text content of the element.

Array Object

The Array object represents an array of values in JavaScript. Arrays are a way to store multiple values in a single variable, and they can be manipulated using a variety of built-in methods, such as the push() method, which adds a new element to the end of an array. Here's an example:

```
const myArray = ["apple", "banana", "orange"];

myArray.push("pear");

console.log(myArray);
```

In this example, we're using the push() method to add a new element ("pear") to the end of an array (myArray), and then we're using the console.log() method to display the updated array.

String Object

The String object represents a string of text in JavaScript. Strings can be manipulated using a variety of built-in methods, such as the toUpperCase() method, which converts all characters in a string to uppercase. Here's an example:

```
const myString = "hello, world!";

const upperString = myString.toUpperCase();

console.log(upperString);
```

In this example, we're using the toUpperCase() method to convert all characters in a string (myString) to uppercase, and then we're using the console.log() method to display the updated string.

Math Object

The Math object provides a set of mathematical functions and constants in JavaScript. The Math object has many properties and methods for performing common mathematical operations, such as the round() method, which rounds a number to the nearest integer. Here's an example:

```
const myNumber = 3.14159;

const roundedNumber = Math.round(myNumber);

console.log(roundedNumber);
```

In this example, we're using the round() method to round a number (myNumber) to the nearest integer, and then we're using the console.log() method to display the rounded number.

Date Object

The Date object represents a date and time value in JavaScript. The Date object has many properties and methods for working with dates and times, such as the getHours() method, which returns the hours value of a date object. Here's an example:

```
const myDate = new Date();

const hours = myDate.getHours();

console.log(hours)
```

Writing scripts to handle events

In JavaScript, you can write scripts that handle events, such as mouse clicks, keyboard input, and page loading. When an event occurs, the browser executes the script code that's associated with the event, allowing you to add interactivity and dynamic behavior to your web pages. Here's an example of how to write a script that handles a mouse click event:

```
// Get a reference to the button element

const myButton = document.getElementById("myButton");

// Add a click event listener to the button

myButton.addEventListener("click", function() {

 // Code to execute when the button is clicked

 console.log("Button clicked!");

});
```

In this example, we're using the getElementById() method to get a reference to an HTML button element with an ID of "myButton". We then use the addEventListener() method to add a click event listener to the button. The second argument to addEventListener() is a function that contains the code to execute when the button is clicked. In this case, we're just logging a message to the console.