DAY-15

JAVASCRIPT

JavaScript is the scripting language of the Web.

JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JavaScript is the most popular scripting language on the Internet, and works in all major browsers, such as Internet Explorer, Mozilla Firefox, and Opera.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do ?

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.

JavaScript Variables

Variables are "containers" for storing information.

JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

```
<html>
<body>
<script type="text/javascript">
var firstname;
firstname="Welcome";
document.write(firstname);
document.write("<br />");
firstname="XYZ";
document.write(firstname);
</script>

<p>The script above declares a variable,
assigns a value to it, displays the value, change the value,
and displays the value again.</p>

</body>
</html>
```

Output :
Welcome
XYZ

The script above declares a variable, assigns a value to it, displays the value, change the value, and displays the value again.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Scorpio";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Scorpio**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
carname="Scorpio";
```

have the same effect as:

```
var x=5;
var carname="Scorpio";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
var x;
```

DAY-16

1. JavaScript Functions:

- **Defining Functions:** Functions are blocks of code designed to perform specific tasks and can be reused throughout the code.

```
js
Copy code
// Function declaration function greet(name) { return "Hello, " + name; } // Function expression const greet = function(name) { return "Hello, " + name; }; // Arrow function (ES6) const greet = (name) => "Hello, " + name;
```

- **Calling Functions:**

```
js
Copy code
console.log(greet("Alice")); // Output: Hello, Alice
```

- **Parameters and Arguments:** Functions can accept parameters and use them within the function body.

```
js
Copy code
function add(a, b) { return a + b; } console.log(add(5, 3)); // Output: 8
```

- **Default Parameters:** Parameters can have default values if no arguments are provided.

```
js
Copy code
function greet(name = "Guest") { return "Hello, " + name; } console.log(greet()); // Output: Hello, Guest
```

- **Returning Values:** Functions can return values using the `return` statement.

```
js
Copy code
function multiply(a, b) { return a * b; } let result = multiply(4, 5); console.log(result); // Output: 20
```

DAY -17

1. Introduction to the DOM:

- The DOM is a programming interface for HTML and XML documents.

- It represents the document as a tree structure where each node is an object representing a part of the document.

2. Accessing DOM Elements:

- **Selecting Elements:**
-
- `getElementById()`

```
let element = document.getElementById('elementId');
```

- `getElementsByClassName()`

```
let elements = document.getElementsByClassName('className');
```

- `getElementsByTagName()`

```
let elements = document.getElementsByTagName('tagName');
```

- `querySelector()`

```
let element = document.querySelector('.className');
```

- `querySelectorAll()`
-

```
let elements = document.querySelectorAll('.className');
```

3. Modifying DOM Elements:

- **Changing Content:**

- `innerHTML`

```
element.innerHTML = 'New Content';
```

- `textContent`

```
element.textContent = 'New Text';
```

- **Changing Attributes:**

- `setAttribute()`

```
element.setAttribute('attributeName', 'attributeValue');
```

- `getAttribute()`

```
let value = element.getAttribute('attributeName');
```

- `removeAttribute()`

```
element.removeAttribute('attributeName');
```

- **Changing Styles:**
- `style`

```
element.style.color = 'blue';
```

4. Creating and Removing Elements:

- **Creating Elements:**

```
let newElement = document.createElement('div'); newElement.textContent = 'Hello, World!';
document.body.appendChild(newElement);
```

- **Removing Elements:**

```
element.remove();
```

DAY-18

1. Conditional Statements:

- **if Statement:** Executes a block of code if a specified condition is true.

```
let age = 18; if (age >= 18) { console.log("You are an adult."); }
```

- **if...else Statement:** Provides an alternative block of code if the condition is false.

```
let age = 17; if (age >= 18) { console.log("You are an adult."); } else { console.log("You are a minor."); }
```

- **if...else if...else Statement:** Tests multiple conditions.

```
let score = 85; if (score >= 90) { console.log("Grade: A"); } else if (score >= 80) { console.log("Grade: B"); } else { console.log("Grade: C"); }
```

- **Switch Statement:** Provides a way to execute different blocks of code based on different values of an expression.

```
let day = 3; switch (day) { case 1: console.log("Monday"); break; case 2: console.log("Tuesday"); break; case 3: console.log("Wednesday"); break; default: console.log("Another day"); }
```

2. Loops:

- **for Loop:** Used to repeat a block of code a specific number of times.

```
for (let i = 0; i < 5; i++) { console.log("Iteration:", i); }
```

- **while Loop:** Repeats a block of code as long as a specified condition is true.

```
let i = 0; while (i < 5) { console.log("Iteration:", i); i++; }
```

- **do...while Loop:** Similar to the `while` loop, but it will execute the block of code at least once before checking the condition.

```
let i = 0; do { console.log("Iteration:", i); i++; } while (i < 5);
```