What is JavaScript?

- A high-level, interpreted scripting language primarily used to add interactivity to webpages.
- It's a cornerstone of modern web development, working alongside HTML and CSS.
- Runs on the client-side, meaning directly within the user's web browser.

Why Use JavaScript?

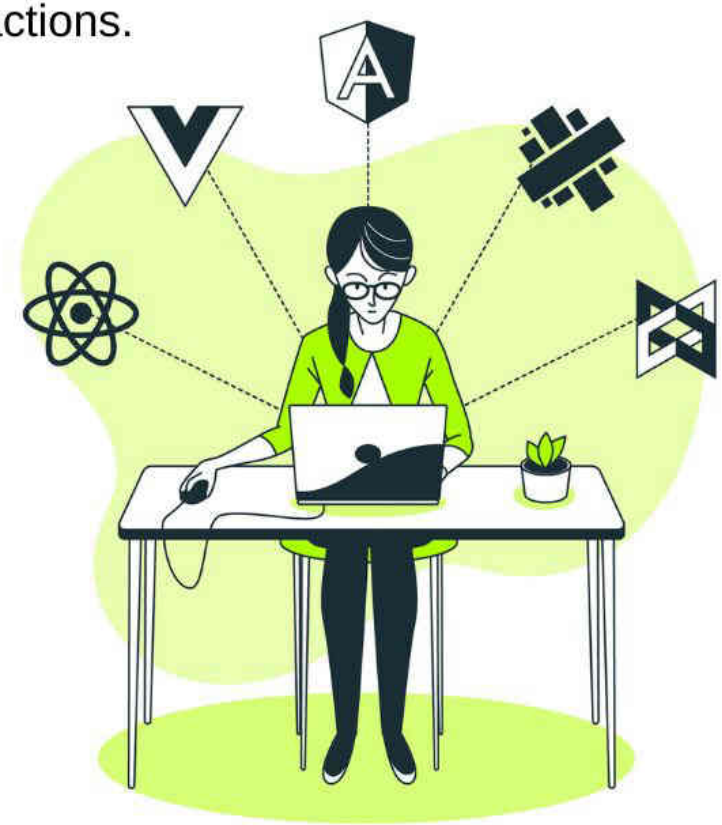
JavaScript allows you to create interactive web applications. With JavaScript, you can validate user inputs, respond to user actions (like clicks and keystrokes), fetch data from servers without reloading the page (using AJAX), create animations, and much more. It's a fundamental part of modern web development.

How to Use JavaScript?

You can include JavaScript code directly within HTML files using <script> tags, or you can write JavaScript code in separate .js files and link them to your HTML. JavaScript code is executed by the web browser, meaning it runs on the client-side (in the user's browser).

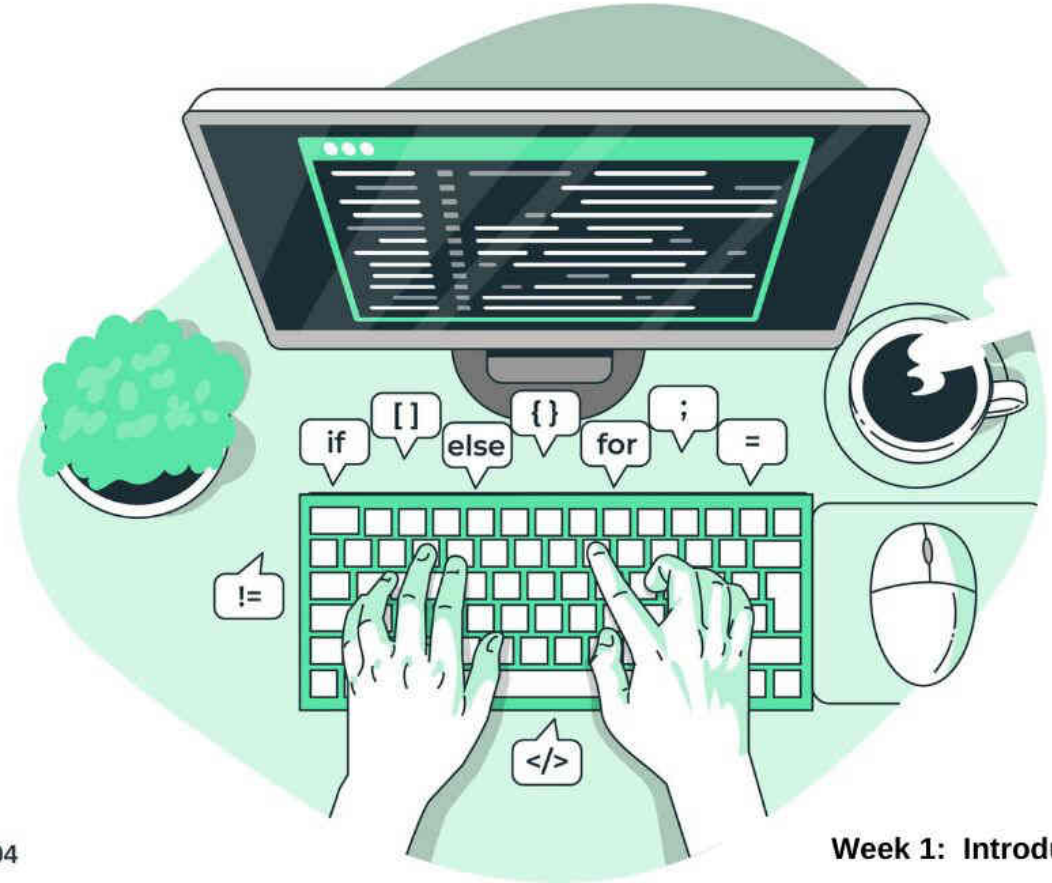
Key Features:

- Lightweight and Interpreted: Code is relatively small and doesn't require compilation, making it easier to learn and experiment.
- Dynamically Typed: Data types are determined at runtime, offering flexibility but requiring careful handling.
- Object-Oriented: Supports object-oriented programming concepts like objects, classes, and inheritance.
- Event-Driven: Reacts to user actions and browser events, enabling dynamic interactions.



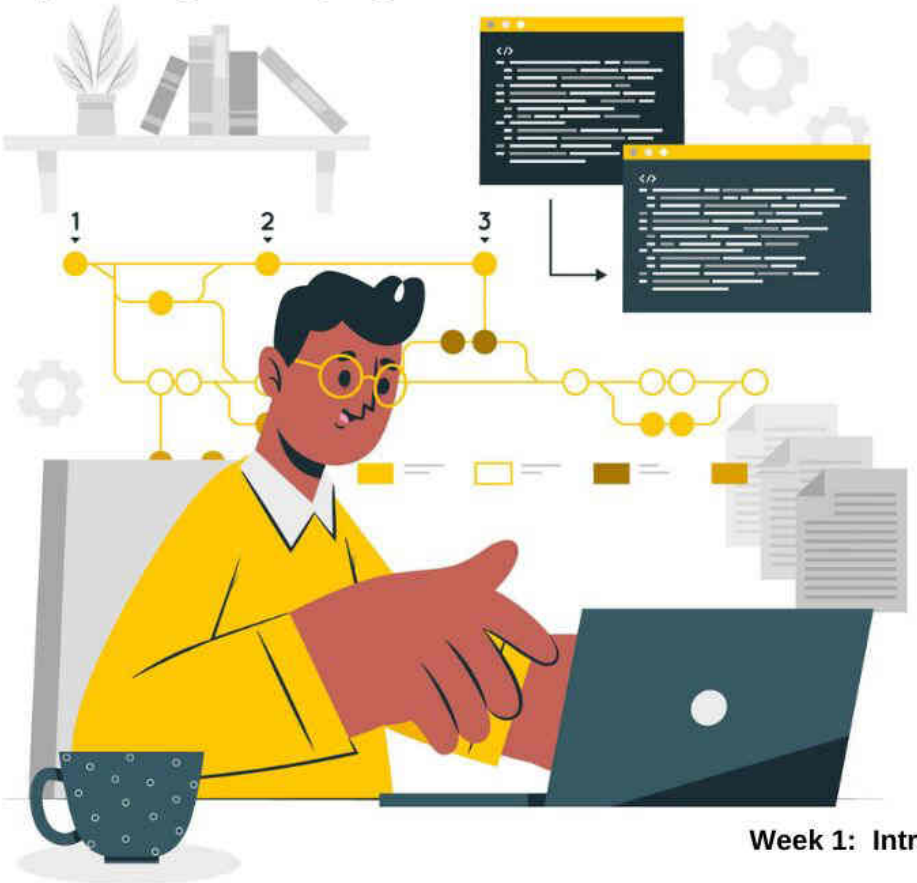
What Does JavaScript Do?

- Adds Interactivity: Creates dynamic elements like animations, image carousels, and form validation.
- DOM Manipulation: Modifies the webpage structure (Document Object Model) by adding, removing, or changing elements.
- Ajax Communication: Enables communication with web servers in the background without refreshing the page, allowing for features like live data updates and chat applications.



Core Concepts:

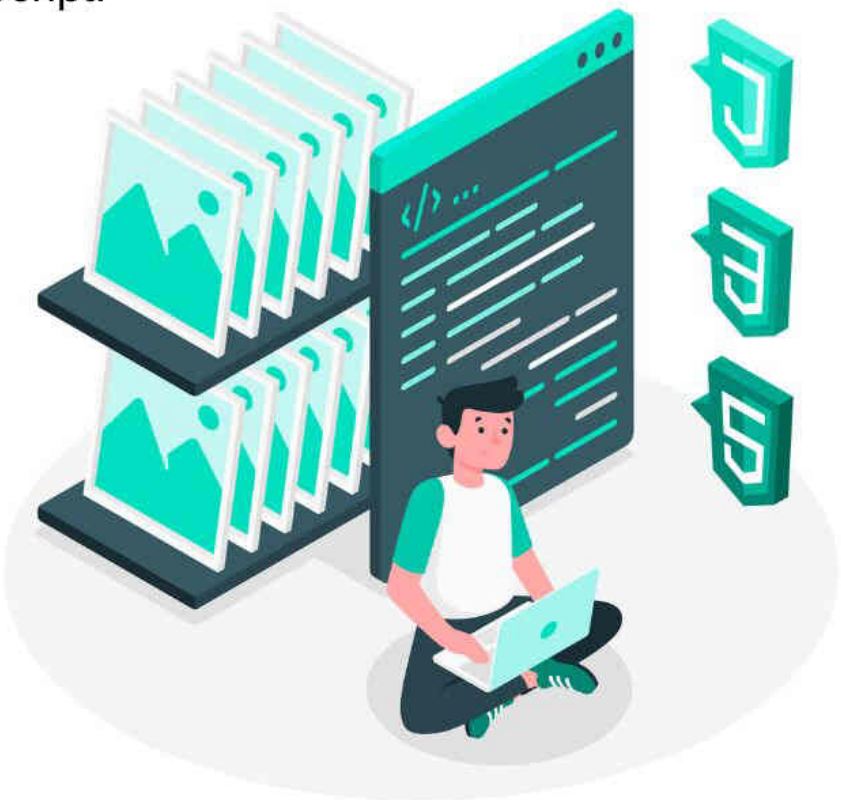
- Variables: Used for storing data.
- Data Types: Such as numbers, strings, booleans, arrays, and objects.
- Functions: Blocks of reusable code.
- Control Flow: Conditional statements (if-else) and loops (for, while).
- DOM Manipulation: Interacting with the Document Object Model to dynamically change webpage content, structure, and styling.



Frameworks and Libraries:

- In addition to vanilla JavaScript, there are many frameworks and libraries available, such as React, Angular, and Vue.js, which provide additional tools and abstractions to streamline web development and build more complex web applications.

- JavaScript (or "JS") is a programming language used most often for dynamic client-side scripts on webpages, but it is also often used on the server side, using a runtime such as Node.js.
- Node. js is not a framework. Instead, Node. js is a JavaScript runtime environment.
- Node.js is a cross-platform JavaScript runtime environment that allows developers to build server-side and network applications with JavaScript.



JavaScript (JS) is a versatile scripting language that brings webpages to life. It's a core technology behind the dynamic and engaging experiences we encounter online.

Basics of JavaScript:

- Client-side vs. Server-side: JavaScript primarily runs on the client-side, meaning directly within the user's web browser. This allows for immediate responses and interactions without needing to refresh the entire page.
- Lightweight and Interpreted: JavaScript code is relatively small and doesn't require compilation before execution. The browser interprets the code line by line, making it easier to learn and experiment.
- Dynamically Typed: Unlike some languages, JavaScript doesn't strictly define data types until runtime. This offers flexibility but requires careful handling to avoid errors.
- Core Building Blocks: Variables, operators, control flow statements (if/else, loops), and functions are fundamental elements for creating JavaScript programs.

Diving into the basics of JavaScript

Here are some key concepts to grasp:

1. Scripting and Execution:

- JavaScript code is written in scripts embedded within HTML pages or linked as external files.
- The browser interprets and executes these scripts, adding interactivity to the webpage.

2. Variables and Data Types:

- Variables store data used within your code. You declare them using keywords like let, const, or var.
  - JavaScript has several data types, including Numbers (integers and decimals)
  - Strings (text)
  - Booleans (true or false)
  - Arrays (ordered lists of values)
  - Objects (collections of key-value pairs)

Variables: Variables are containers for storing data values. In JavaScript, you can declare variables using the var, let, or const keywords. Variables declared with var have function scope, while those declared with let and const have block scope.

variables

```
var age = 30; // var (older way)
let name = "John"; // let (preferred for variables that will change)
const PI = 3.14; // const (for constants)
```

Data Types: JavaScript supports various data types, including numbers, strings, booleans, arrays, objects, and more.

datatypes

```
let num = 10;
let greeting = "Hello";
let isTrue = true;
let fruits = ["apple", "banana", "orange"];
let person = { name: "John", age: 30 };
```

3. Operators:

Operators perform calculations or comparisons on data. Examples include:

- Arithmetic operators (+, -, *, /)
- Comparison operators (==, !=, <, >)
- Logical operators (&&, ||, !)

Operators: JavaScript supports different types of operators, such as arithmetic, assignment, comparison, and logical operators.

operators

```
let x = 10;
let y = 5;
let sum = x + y; // arithmetic operator
let isEqual = x === y; // comparison operator
let isTrue = (x > 5) && (y < 10); // logical operator
```

4. Control Flow:

Control flow statements dictate how your code executes. Common ones are

- Conditional statements (if/else) to make decisions based on conditions.
- Loops (for, while) to repeat code blocks multiple times.

Conditional Statements: JavaScript supports conditional statements like **‘ if ’**, **‘ else if ’**, and **‘else’** for executing different blocks of code based on certain conditions.

conditional statements

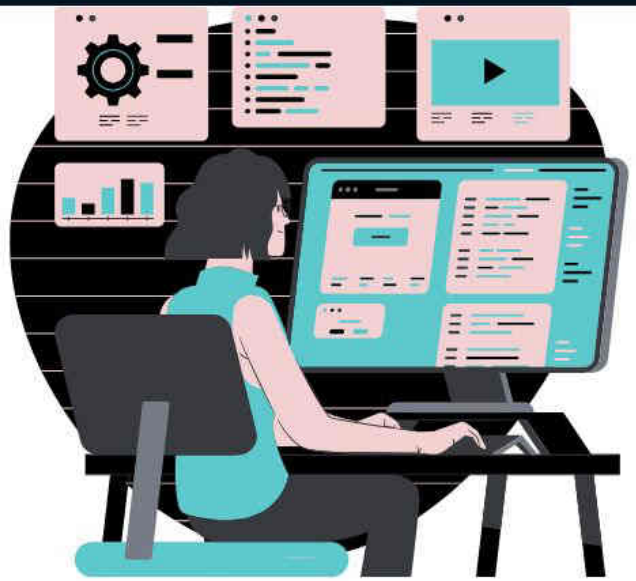
```
let num = 10;
if (num > 0) {
    console.log("Number is positive");
} else if (num < 0) {
    console.log("Number is negative");
} else {
    console.log("Number is zero");
}
```

Loops:

Loops allow you to execute a block of code multiple times. JavaScript supports **‘for’**, **‘while’**, and **‘do-while’** loops.

loops

```
for (let i = 0; i < 5; i++) {
    console.log(i); // Outputs: 0, 1, 2, 3, 4
}
```

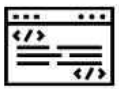


Functions:

- Functions are blocks of reusable code. They can take parameters and return values.
- Functions are reusable blocks of code that perform specific tasks. You can define functions with parameters and return values.

functions

```
function greet(name) {
    return "Hello, " + name + "!";
}
console.log(greet("John")); // Outputs: Hello, John!
```



6. DOM Manipulation:

- The Document Object Model (DOM) represents the webpage structure. JavaScript can access and modify elements within the DOM, allowing you to:
  - Change element content
  - Add or remove elements
  - Apply styles dynamically

7. Events:

- Events are user actions or browser occurrences that trigger JavaScript code. Examples include:
  - Button clicks
  - Mouse movements
  - Page loading

8. Additional Concepts:

- Comments: Lines ignored by the browser, used for explanations within your code.
- Console: A browser tool for debugging and inspecting JavaScript output.

Adding Interactivity with JavaScript:

- JavaScript shines in making webpages interactive.
- Adding interactivity to a webpage with JavaScript involves responding to user actions, manipulating the DOM (Document Object Model), and dynamically updating content. Here's how you can do it:

1. **Event Handling:** JavaScript allows you to respond to user actions like clicks, mouse movements, key presses, etc., using event listeners.

event handling

```
// Adding a click event listener to a button
const button = document.querySelector('button');
button.addEventListener('click', function() {
    alert('Button clicked!');
});
```

2. **DOM Manipulation:** You can manipulate the DOM to dynamically change the content, structure, and style of your webpage based on user interactions or other events**.**

DOM

```
// Changing text content
const heading = document.querySelector('h1');
heading.textContent = 'New Heading';

// Adding or removing HTML elements
const newDiv = document.createElement('div');
newDiv.textContent = 'New Div Element';
document.body.appendChild(newDiv);

// Changing CSS styles
const paragraph = document.querySelector('p');
paragraph.style.color = 'red';
```

3. **Form Handling:** JavaScript can be used to validate form inputs and perform actions when a form is submitted.

```
form handling

// Form validation
const form = document.querySelector('form');
form.addEventListener('submit', function(event) {
    event.preventDefault(); // Prevent default form
submission
    const input = document.querySelector('input');
    if (input.value === '') {
        alert('Please enter a value!');
    } else {
        alert('Form submitted with value: ' + input.value);
    }
});
```

4. **Ajax and Fetch**: JavaScript allows you to make asynchronous HTTP requests to fetch data from a server without reloading the entire page.

ajax and fetch

```
// Fetching data from a server
fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => {
        console.log(data);
    })
    .catch(error => {
        console.error('Error fetching data:', error);
    });
```

5. **Animations:** JavaScript can be used to create animations by manipulating CSS properties over time.

```functions
// Creating a simple animation
const element = document.querySelector('.box');
let position = 0;
setInterval(() => {
    position += 5;
    element.style.left = position + 'px';
}, 100); // Move the element 5 pixels to the right every 100
milliseconds
```


JS
THANK YOU!