JS Basic - 1.5(Hours Course)

Introduction to JavaScript

> JavaScript is a versatile programming language commonly used for web development. It is a high-level, interpreted language that integrates seamlessly with HTML and CSS, allowing developers to enhance the functionality and design of web pages. By enabling the creation of interactive and dynamic web features such as form validation, animations, and user interface enhancements, JavaScript makes websites more engaging and user-friendly. Additionally, JavaScript can be executed on both the client-side (in the browser) and server-side (using environments like Node.js). On the client-side, it allows for real-time updates and interactions without needing a page reload, while on the server-side, Node.js enables JavaScript to manage backend development tasks, including database operations, API integrations, and server configurations, thereby providing a unified language for the entire web development stack.

Brief History of JavaScript

- JavaScript was created by Brendan Eich in 1995 while he was working at Netscape Communications Corporation.
- Initially named Mocha, it was later renamed to LiveScript and finally to JavaScript.
- JavaScript has since evolved through various versions and standards, with ECMAScript being the standard specification.

JavaScript in Web Development

- **Manipulating the DOM (Document Object Model)**
  - JavaScript can access and modify HTML and CSS to update the content and design of a webpage dynamically.
  - Example: Changing the text of an HTML element.
- **Event Handling**
  - JavaScript can respond to user actions like clicks, mouse movements, and key presses to create interactive experiences.
  - Example: Alerting a message when a button is clicked.
- **Form Validation**
  - JavaScript can validate user input in forms before submitting them to the server, enhancing user experience and reducing server load.
  - Example: Checking if a text field is empty
- **AJAX (Asynchronous JavaScript and XML)**
  - JavaScript can send and receive data from a server asynchronously, allowing web pages to update without reloading.
  - Example: Fetching data from an API
- **Creating Interactive Elements**
  - JavaScript can be used to create interactive elements like image sliders, modals, and navigation menus.
  - Example: Toggling a navigation menu.

Why Learn JavaScript?

- JavaScript is essential for modern web development.
- It enhances user experience by making web pages interactive.
- It's a versatile language with a vast ecosystem and community support.

▼ Tools and Setup

- **Text Editors**: Visual Studio Code, Sublime Text, Atom.

- **Browsers**: Chrome, Firefox, Safari with built-in developer tools.
- **Node.js**: For running JavaScript on the server side.
- **Version Control**: Using Git and GitHub for code management.

▼ Basic Syntax and Conventions

- **Case Sensitivity**: JavaScript is case-sensitive.
- **Semicolons**: Optional but recommended for ending statements.
- **Comments**: Single-line (`//`) and multi-line (`/* */`).

Basic Syntax and Conventions in JavaScript

Case Sensitivity

- JavaScript is case-sensitive. This means that `myVariable` and `myvariable` are treated as different variables.

Semicolons

- Semicolons (`;`) are used to terminate statements in JavaScript. While they are optional, it's good practice to use them to avoid potential errors.

Comments

- Comments are used to annotate code and make it more readable.
  - **Single-line comments** start with `//`
  -

```
// This is a single-line comment
```

  - **Multi-line comments** are enclosed between `/*` and `*/`

```
/*
This is a multi-line comment
that spans multiple lines
*/
```

Variables

- Variables are used to store data values. JavaScript supports three ways to declare variables: `var`, `let`, and `const`.
  - `var` is function-scoped and can be re-assigned.

```
var name = "John";
name = "Doe";
```

  - `let` is block-scoped and can be re-assigned.

```
let age = 25;
age = 30;
```

  - `const` is block-scoped and cannot be re-assigned.

```
const pi = 3.14;
```

Data Types

  - JavaScript supports different data types including:
    - **String**: Textual data enclosed in quotes (`""` or `''`).

```
let greeting = "Hello, World!";
```

    - **Number**: Numeric data, both integers and floating-point numbers.

```
let age = 25;
let price = 19.99;
```

    - **Boolean**: Represents true or false values.

```
let isStudent = true;
```

    - **Array**: Ordered list of values.

```
let colors = ["red", "green", "blue"];
```

- **Object**: Collection of key-value pairs.

```
let person = {
    firstName: "John",
    lastName: "Doe",
    age: 30
};
```

- **Null**: Represents the intentional absence of any object value.

```
let emptyValue = null;
```

- **Undefined**: Indicates a variable has been declared but not assigned a value.

```
let notAssigned;
```

Operators

- Operators are used to perform operations on variables and values. Common operators include:
  - **Arithmetic Operators**: `+`, `-`, `*`, `/`, `%` (modulus), `++` (increment), `--` (decrement).

```
let sum = 5 + 2;  // Addition
let difference = 5 - 2;  // Subtraction
let product = 5 * 2;  // Multiplication
let quotient = 5 / 2;  // Division
let remainder = 5 % 2;  // Modulus
```

  - **Comparison Operators**: `==` (equal to), `===` (strict equal to), `!=` (not equal to), `!==` (strict not equal to), `>` (greater than), `<` (less than), `>=` (greater

than or equal to), `<=` (less than or equal to).

```
let isEqual = (5 == "5");  // true
let isStrictEqual = (5 === "5");  // false
```

- **Logical Operators**: `&&` (and), `||` (or), `!` (not).

```
let andOperation = (true && false);  // false
let orOperation = (true || false);  // true
let notOperation = (!true);  // false
```

- **Assignment Operators**: `=`, `+=`, `=`, `=`, `/=`, `%=`.

```
let x = 10;
x += 5;  // x = x + 5
```

Math in JavaScript is handled through the built-in `Math` object, which provides a variety of mathematical constants and functions. Here are some commonly used methods and properties of the `Math` object

Basic Math Operations

JavaScript handles basic arithmetic operations directly with operators:

```
let a = 10;
let b = 3;

let sum = a + b; // Addition
let difference = a - b; // Subtraction
let product = a * b; // Multiplication
let quotient = a / b; // Division
let remainder = a % b; // Modulus
```

Math In Js

Math Object Methods

Rounding Numbers

- `Math.ceil(x)`: Rounds `x` up to the next largest integer.
- `Math.floor(x)`: Rounds `x` down to the next smallest integer.
- `Math.trunc(x)`: Truncates the decimal part of `x`.

```
let num = 5.7;

Math.round(num); // 6
Math.ceil(num); // 6
Math.floor(num); // 5
Math.trunc(num); // 5
```

Exponents and Logarithms

- `Math.pow(base, exponent)`: Returns the base raised to the exponent power.
- `Math.sqrt(x)`: Returns the square root of `x`.
- `Math.cbrt(x)`: Returns the cube root of `x`.
- `Math.exp(x)`: Returns `e^x`.
- `Math.log(x)`: Returns the natural logarithm (base e) of `x`.
- `Math.log10(x)`: Returns the base-10 logarithm of `x`.
- `Math.log2(x)`: Returns the base-2 logarithm of `x`.

```
Math.pow(2, 3); // 8
Math.sqrt(16); // 4
Math.cbrt(27); // 3
Math.exp(1); // 2.718281828459045 (approximately e)
Math.log(10); // 2.302585092994046 (approximately ln(10))
```

```
Math.log10(100); // 2
Math.log2(8); // 3
```

Generate Random Number In Js

- random is a fuction in js→

```
let a = Math.random() → generate number form 0 to 0.9
```

Creating Dates

1. **Current Date and Time**

```
let now = new Date();
console.log(now); // Current date and time
```

1. **Specific Date and Time**

```
let specificDate = new Date('2024-07-22T10:20:30Z');
console.log(specificDate); // Mon Jul 22 2024 10:20:30 GMT+0000 (Coordina
ted Universal Time)
```

- Getting Date and Time Components

```
let date = new Date();

console.log(date.getFullYear()); // 2024
console.log(date.getMonth()); // 6 (July, since months are 0-indexed)
console.log(date.getDate()); // 22
console.log(date.getDay()); // 1 (Monday, as days are 0-indexed with Sunday
as 0)
console.log(date.getHours()); // 10
console.log(date.getMinutes()); // 20
console.log(date.getSeconds()); // 30
```

```
console.log(date.getMilliseconds()); // 0
console.log(date.getTime()); // Milliseconds since January 1, 1970
```

- Setting Date and Time Components

```
let date = new Date();

date.setFullYear(2025);
date.setMonth(11); // December
date.setDate(25);
date.setHours(12);
date.setMinutes(30);
date.setSeconds(45);

console.log(date); // Thu Dec 25 2025 12:30:45 GMT+0000 (Coordinated Univ
ersal Time)
```

- Date Formatting

While JavaScript's `Date` object doesn't provide extensive formatting options, you can use methods to construct formatted date strings:

```
let date = new Date();

console.log(date.toDateString()); // 'Mon Jul 22 2024'
console.log(date.toTimeString()); // '10:20:30 GMT+0000 (Coordinated Univer
sal Time)'
console.log(date.toISOString()); // '2024-07-22T10:20:30.000Z'
console.log(date.toLocaleDateString()); // Locale-specific date
console.log(date.toLocaleTimeString()); // Locale-specific time
console.log(date.toLocaleString()); // Locale-specific date and time
```

- Date Arithmetic

```
let today = new Date();
let tomorrow = new Date(today);
```

```
tomorrow.setDate(today.getDate() + 1);

console.log(tomorrow); // Date object representing tomorrow's date
```