Day 1 of JS30Xplore

Today's Topic: -

What is Javascript?

- Why do we use Javascript?

- How to set up a development environment for JS.

- Including JavaScript in HTML

- Writing the first "hello world" program in JS

- Running Your First Program

1. What is JavaScript?

JavaScript is a high-level, interpreted programming language. It is one of the three core technologies of the World Wide Web, alongside HTML and CSS. JavaScript is used to make web pages interactive and dynamic. It can also be used to develop web applications, mobile applications, and games. JavaScript is a text-based language, which means that it is written in plain text files. JavaScript code is then interpreted by a JavaScript engine, which is a software component that executes JavaScript code. The JavaScript engine is typically built into a web browser, but it can also be found in other environments, such as servers and mobile devices. JavaScript is a front-end programming language, which means that it is used to develop the user interface of a web application. JavaScript can also be used to develop back-end web applications, which are the servers that power web applications. However, JavaScript is most commonly used for front-end development.

2. Why Javascript is required?

JavaScript is indispensable in web development for its ability to introduce interactivity and responsiveness to web pages. By running on the client side, it reduces server loads and enables dynamic content updates without full page reloads. JavaScript also validates user input, accesses web APIs for additional functionality, supports asynchronous operations for seamless user experiences, and enhances websites with animations and interactive elements. These capabilities make JavaScript a fundamental tool for creating modern web applications that engage users and deliver dynamic content.

3. Setting Up Your Development Environment

To get started with JavaScript, you'll need a development environment. Here are the essential tools: **- Code Editor:** Use a code editor like Visual Studio Code, Sublime Text to write your JavaScript code. **- Web Browser:** Most modern browsers like Chrome, Firefox, or Edge are suitable for running JavaScript code.

4. Including JavaScript in HTML

JavaScript code can be included in an HTML document using the <script> tag. There are two primary ways to do this:

- Inline script

- External script

5. Writing Your First JavaScript Code

Let's write your very first JavaScript program: a simple "Hello World" message.
console.log("Hello, world!");

The **console.log()** function is used to display output in the browser's console.

6. Running Your First Program

To run your JavaScript program:

- Save your HTML file.

- Open the HTML file in your web browser.

- Open the browser's developer console. You can usually do this by right-clicking on the page and selecting "**Inspect"** or "**Inspect Element**," and then navigating to the "**Console**" tab.

Day 2 of JS30Xplore

Today's topics:

- What is Variable

- What are data types
- Naming rule in variable
- Assigning and reassigning variable
- Working with data types

- Type coercion and conversion

1. **What are Variables?**

Variables are fundamental in JavaScript; they act as containers for storing data values. In JavaScript, you can declare variables using three different keywords: ''**var"**, ''**let"**, and ''**const".**
The choice between them depends on the scope and mutability of the variable.

- var: Variables declared with var are function-scoped and can be redeclared within the same scope. However, it's considered outdated and is rarely used in modern JavaScript.

- let: Introduced in ES6 (ECMAScript 2015), let allows you to declare block-scoped variables that can be reassigned. It's the preferred choice for most variable declarations

- const: Variables declared with const are also block-scoped but cannot be reassigned once they are given a value. Use const for values that should not change.

2. **What are Data Types?**

Data types in programming are like categories that tell the computer what kind of data you're working with. JavaScript has several built-in data types, which can be categorized as follows:
- String: Used for textual data and enclosed in single or double quotes.
- Number: Represents numeric values, including integers and floating-point numbers.
- Boolean: Represents true or false values, often used in conditional statements.
- Undefined: "Undefined" means a variable exists but has no value;
- Null: "null" means it's intentionally empty, like an empty box.
- Object: A collection of key-value pairs, allowing you to create complex data structures.
- Array: An ordered list of values, accessible by their index.

3. **Variable Naming Rules**

When naming variables in JavaScript, follow these rules:
• Variable names are case-sensitive (''**myVariable**" and ''**myvariable**" are different).
• They can include letters, digits, underscores, and the dollar sign.
• Variable names cannot start with a digit. • Avoid using JavaScript-reserved keywords as variable names.

4. **Assigning and Reassigning Variables**
   JavaScript allows you to assign and reassign values to variables. The choice of “**let** ” or ''**const**" will determine whether a variable can be reassigned.

5. **Working with Data Types in JavaScript:**
   • **Concatenate Strings:** Use the (+)operator to combine strings, like adding names to greetings.
   let greeting = "Hello,";
   let name = "Aina";
   let message = greeting + name;
   // Results in "Hello, Aina"

   • **Mathematical Operations:** Perform calculations with numbers, such as addition or subtraction. These tools allow you to handle different types of data, manipulate text and numbers, make choices based on conditions, and manage missing or empty data.

6. **Type Coercion and Conversion**
   JavaScript employs type coercion, which means it automatically converts data types when needed. For instance, adding a string and a number will result in type coercion, where the number is converted to a string.
   You can also explicitly convert data types using functions like ''**parseInt()**", ''**parseFloat()"**, and ''**String()**".

   Type conversion is when you intentionally change a value from one data type to another.
   In JavaScript, you can do this using specific functions or operations. For example, you might want to convert a string containing a number into an actual number so that you can perform mathematical calculations with it.

   Day 3 of JS30Xplore:

   -What are Operators?
   Operators in JavaScript are special symbols or keywords used to perform operations on values and variables. They are essential for tasks like calculations, comparisons, and logical operations.
   -Arithmetic Operators

Arithmetic operators are used for mathematical calculations:
• Addition (+): Adds two numbers.
• Subtraction (-): Subtracts the second number from the first.
• Multiplication (*): Multiplies two numbers.
• Division (/) : Divides the first number by the second.
• Modulus (%) : Returns the remainder of a division operation.

- Comparison Operators:
Comparison operators help you compare values:
• Equal == and Strict Equal ===: Checks if two values are equal.
The strict equal === also compares data types.

• Not Equal != and Strict Not Equal !==: Checks if two values are not equal.
The strict not equal !== also compares data types.
• Greater Than > and Less Than <: Compares if one number is greater or less than another.
• Greater Than or Equal To >= and Less Than or Equal To <=: Compares if one number is greater than or equal to or less than or equal to another.

- Logical Operators Logical operators work with boolean values:
• Logical AND (&&) : Returns true if both conditions are true.
• Logical OR (||): Returns true if at least one condition is true.
• Logical NOT (!): Inverts the boolean value, turning true into false and vice versa.

- Assignment Operators Assignment operators are used to assign values to variables:
• Assignment (=): Assigns a value to a variable.
• Addition Assignment (+=): Adds a value to the variable.
• Subtraction Assignment (=): Subtracts a value from the variable.
• Multiplication Assignment =: Multiplies the variable by a value.
• Division Assignment (/=): Divides the variable by a value.
• Modulus Assignment (%=): Assigns the remainder of a division to the variable.

- Operator Precedence Operators have different levels of precedence, determining the order in which operations are executed in an expression.
For example, Multiplication (*) and division (/) have higher precedence than addition (+) and subtraction (-).

- Expressions Expressions are combinations of values, variables, and operators that produce a result. Examples include arithmetic expressions (2 + 3), comparison expressions (x > 5), and logical expressions ((a && b) || c).

Day 4 of JS30Xplore

1. **Introduction to Conditional Statements**
   Conditional statements in JavaScript allow you to make decisions in your code based on specific conditions. These statements are essential for controlling the flow of your program.
2. **The if Statement**
   The if statement is the most fundamental type of conditional statement in JavaScript. It allows you to execute a block of code if a condition is true.

3. **The else if Statement** The else if statement lets you test multiple conditions in sequence. If the first condition is not met, it checks the next one.

   4. The else Statement The else statement provides a default action when none of the previous conditions are met.
   5. Nested Conditional Statements You can nest (if, else if, and else) statements inside one another to handle complex decision-making scenarios.

   6. The Ternary Operator The ternary operator (condition ? expression1: expression2) provides a concise way to perform simple conditional assignments.
   7. Switch Statement The switch statement is used for multi-way conditional branching based on a specific value.
   8. Control Flow Conditional statements control the flow of your program. Depending on the conditions met, the program takes different paths, executing specific blocks of code.

Day 5 of JS30Xplore:

1. What are Functions? Functions are at the heart of JavaScript and programming in general. They are reusable blocks of code that serve specific purposes or return values. Functions are essential for structuring your code in an organized, modular, and efficient

manner. They allow you to break your code into smaller, manageable parts that can be used whenever needed.

2. Declaring Functions:

In JavaScript, you declare a function using the function keyword. A function declaration typically includes: - The "function" keyword. - A function name that you choose (e.g., greet). - A set of parentheses that can hold parameters (input values). - A code block enclosed in curly braces {} that contains the instructions for what the function should do.

3. Calling Functions: Once you've defined a function, you can call it to execute the code inside the function. To call a function, simply use its name followed by parentheses. If the function expects any arguments (parameters), you provide those within the parentheses.

4. Function Parameters: Functions can receive parameters, which are like variables that hold values you pass into the function. Parameters make functions versatile because you can pass different data to them when calling the function

5. Return Statements: Functions can use the "return" statement to send a value back when they're called. This value can be assigned to a variable or used directly

6. Function Expressions: Functions can also be defined as expressions and assigned to variables. These are known as function expressions. Function expressions are useful when you need to pass functions as arguments to other functions or store them in data structures.

7. Arrow Functions: Arrow functions provide a concise way to define functions, especially for simple, one-liner functions. They are a shorthand for writing functions and are often used for anonymous functions.

8. Function Scope: In JavaScript, variables declared inside a function are scoped to that function. They are not accessible from outside the function, which is known as function scope. This scoping behavior helps prevent variable name collisions and isolates variables within the context of the function.

9. Function Declarations vs. Function Expressions: It's essential to understand the difference between function declarations and function expressions. Function declarations are hoisted, meaning they are available throughout their scope, even before the declaration.