**📝 JavaScript Notes**

JavaScript (JS) is the programming language that makes websites interactive and dynamic. If HTML is the skeleton of a webpage and CSS is the skin and clothes, then **JavaScript is the brain and muscles** that make it move and react.

**## 1. Getting Started: How to Run JavaScript**

**Method : Inside an HTML File**

You can place JS code directly inside an HTML file using the <script> tag.

**Example:** Create a file named index.html and put this code in it. Open the file in your browser.

<!DOCTYPE html>

<html>

<head>

<title>My First JS Page</title>

</head>

<body>

<h1>Look at the alert!</h1>

<script>

// This is JavaScript code inside the HTML file

alert("Hello from the HTML file!");

</script>

</body>

</html>

**## 2. The Absolute Basics: Syntax & Comments**

**Comments**

Comments are notes in your code that the computer ignores. They are for you and other programmers to understand what the code does.

// This is a single-line comment.

/\*

This is a

multi-line comment.

It can span across

several lines.

\*/

**Variables: Storing Information**

Variables are like labeled boxes where you can store data. In modern JS, we use let and const.

* let: Use for variables whose value might change.
* const: Use for variables whose value will **not** change (constants).

**Rule of thumb:** Always use const unless you know you will need to reassign the variable's value.

**Example:**

// Using let because the age will change next year.

let userAge = 25;

console.log(userAge); // Outputs: 25

// We can change the value of a 'let' variable

userAge = 26;

console.log(userAge); // Outputs: 26

// Using const because a person's name usually doesn't change.

const userName = "Alice";

console.log(userName); // Outputs: Alice

// This would cause an error, because you can't change a const!

// userName = "Bob";

**console.log()** is your best friend for debugging. It prints values to the browser console so you can see what's happening in your code.

**## 3. Data Types: The Building Blocks**

JavaScript has several types of data you can store in variables.

**String**

Textual data. You must wrap strings in single quotes '...' or double quotes "...".

**Example:**

const greeting = "Hello";

const subject = 'World';

// You can join strings together (concatenation)

const fullGreeting = greeting + " " + subject + "!"; // "Hello World!"

console.log(fullGreeting);

**Number**

Used for both whole numbers (integers) and decimal numbers (floats).

**Example:**

const score = 100;

const price = 99.95;

let total = score + 20; // Addition: 120

let difference = score - 10; // Subtraction: 90

let product = score \* 2; // Multiplication: 200

let division = score / 4; // Division: 25

**Boolean**

Represents one of two values: true or false. They are essential for making decisions in your code.

**Example:**

let isUserLoggedIn = true;

let isGamePaused = false;

console.log(isUserLoggedIn); // Outputs: true

**Null and Undefined**

* undefined: A variable has been declared but has not been given a value yet.
* null: Represents the intentional absence of any value. You, the programmer, set it.

**Example:**

JavaScript

let myFutureVariable;

console.log(myFutureVariable); // Outputs: undefined

let selectedCharacter = null; // The user hasn't selected a character yet.

console.log(selectedCharacter); // Outputs: null

**## 4. Interacting with the User**

You can make your scripts interactive.

* alert(message): Shows a pop-up message to the user.
* prompt(question): Asks the user for input and returns it as a **string**.

**Example:**

const userName = prompt("What is your name?");

alert("Hello, " + userName + "! Welcome to our website.");

**## 5. Making Decisions: Conditional Logic**

Sometimes you need your code to run only if a certain condition is met.

**if, else if, else**

This is the most common way to make decisions.

**Example:**

JavaScript

const userAge = prompt("How old are you?");

if (userAge >= 18) {

alert("You are an adult and can enter.");

} else if (userAge >= 13) {

alert("You are a teenager. You can enter with supervision.");

} else {

alert("Sorry, you are too young to enter.");

}

**Comparison Operators**

* === : Strictly equal to (value and type) - **Always use this!**
* !== : Strictly not equal to
* > : Greater than
* < : Less than
* >= : Greater than or equal to
* <= : Less than or equal to

**Example of === vs ==:**

console.log(7 === "7"); // Outputs: false (Number is not the same type as String)

console.log(7 == "7"); // Outputs: true (JS tries to convert types, can be confusing. Avoid!)

**## 6. Storing Lists of Data: Arrays**

An array is an ordered list of items. You can store multiple values in a single variable.

**Example:**

// An array of strings

const fruits = ["Apple", "Banana", "Cherry", "Mango"];

// Arrays use zero-based indexing (the first item is at index 0)

console.log(fruits[0]); // Outputs: "Apple"

console.log(fruits[2]); // Outputs: "Cherry"

// You can find out how many items are in an array

console.log(fruits.length); // Outputs: 4

// You can change an item in the array

fruits[1] = "Blueberry";

console.log(fruits); // Outputs: ["Apple", "Blueberry", "Cherry", "Mango"]

// You can add an item to the end of the array

fruits.push("Strawberry");

console.log(fruits); // Outputs: ["Apple", "Blueberry", "Cherry", "Mango", "Strawberry"]

**## 7. Repeating Actions: Loops**

Loops allow you to run a block of code over and over again.

**The for Loop**

The most common loop. It's great when you know how many times you want to repeat something, like going through an array.

**Syntax:** for (initialization; condition; increment) { ... }

**Example: Looping through our fruits array**

const fruits = ["Apple", "Banana", "Cherry"];

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

// 'i' will be 0, then 1, then 2

console.log("I love to eat " + fruits[i]);

}

**Output:**

I love to eat Apple I love to eat Banana I love to eat Cherry

**## 8. Reusable Code Blocks: Functions**

Functions are blocks of code designed to perform a specific task. You define them once and can run (or "call") them anytime you need.

**Example: A simple greeting function**

// Define the function

function greet(name) {

const message = "Hello, " + name + "! How are you?";

return message;

}

// Call the function and store its result

const greetingForAlice = greet("Alice");

console.log(greetingForAlice); // Outputs: "Hello, Alice! How are you?"

const greetingForBob = greet("Bob");

console.log(greetingForBob); // Outputs: "Hello, Bob! How are you?"

**Example: A function that does a calculation**

// This function takes two numbers and returns their sum

function add(number1, number2) {

return number1 + number2;

}

// Call the function

const sum = add(5, 10);

console.log(sum); // Outputs: 15