Day 1: Introduction to JavaScript, Variables, Data Types, and Console Basics

Welcome to Day 1 of your 30-day JavaScript journey! Today, we'll lay the foundational bricks of your programming knowledge. We'll understand what JavaScript is, how to store information using variables, the different types of data we can work with, and how to see our code's output in the browser console.

What is JavaScript?

JavaScript (JS) is a versatile, high-level, and interpreted programming language primarily used to make web pages interactive. It allows you to create dynamic content, handle multimedia, animate images, and much more. While it's famous for web development (frontend and backend with Node.js), it's also used in mobile apps, desktop apps, and even game development.

Key Characteristics:

Client-Side Scripting: Runs directly in the user's web browser, reducing server load.

Interpreted: Code is executed line by line by the browser without a separate compilation step.

Object-Oriented: Supports object-oriented programming paradigms, allowing for organized and reusable code.

Dynamic and Flexible: Allows for a lot of flexibility in how you write code, including dynamic typing.

Variables: Storing Information Think of variables as named containers that hold data. You give them a name, and then you can put different values into them. In JavaScript, you declare variables using var, let, or const.

var (Older way):

Historically used, but has functional scope and can be re-declared and updated. This can lead to unexpected behavior and bugs in larger applications.

```
var myName = "Ali";
var myName = "Ahmed"; // Can be re-declared (reassigns the variable)
myName = "Kamran";   // Can be updated
console.log(myName); // Output: Kamran
```

let (Modern way - ES6+):

Block-scoped. This means it's only accessible within the block of code (e.g., inside {}) where it's defined. It can be updated but not re-declared within the same scope. This is generally preferred when your variable's value might change.

```
let age = 25;
```

```
age = 26; // Can be updated
console.log(age); // Output: 26
// let age = 27; // Error: Cannot re-declare block-scoped variable 'age'
```

const (Modern way - ES6+):

Also block-scoped. It cannot be re-declared or updated once assigned. Use const for values that should remain constant throughout your program. This prevents accidental reassignments and makes your code more predictable.

```
const PI = 3.14159;
// PI = 3.14; // Error: Assignment to constant variable.
console.log(PI); // Output: 3.14159
```

☑Best Practice: Always prefer const if the value won't change. If you know the value will need to be updated, then use let. Avoid var in new JavaScript code to prevent potential issues related to its scope.

Data Types: Types of Information

JavaScript supports several data types to categorize the kind of value a variable holds. Understanding these is crucial for performing correct operations. You can use the typeof operator to check the data type of a variable.

Primitive Data Types (represent a single, immutable value):

String:

Represents textual data. Enclosed in single (' '), double (" "), or backticks ( ). Backticks allow for template literals, which enable embedding expressions (${expression}) and multiline strings.

```
let greeting = "Hello, World!";
let userName = 'Sara';
let message = `Today is a good day.`; // Using backticks
let fullName = `${userName} Khan`; // Template literal with variable
console.log(typeof greeting); // Output: string
```

Number:

Represents both integer and floating-point numbers. JavaScript handles all numbers as floating-point internally.

```
let score = 100; // Integer
```

```
let price = 19.99; // Floating-point
let temperature = -5;
console.log(typeof score); // Output: number
```

Boolean:

Represents a logical entity and can only have two values: true or false. Used for conditional logic.

```
let isLoggedIn = true;
let hasPermission = false;
console.log(typeof isLoggedIn); // Output: boolean
```

Undefined:

A variable that has been declared but not yet assigned a value is undefined. It means "no value has been assigned yet."

```
let phoneNumber; // phoneNumber is undefined
console.log(phoneNumber);      // Output: undefined
console.log(typeof phoneNumber); // Output: undefined
```

Null:

Represents the intentional absence of any object value. It means "no value," explicitly set by the programmer.

```
let selectedUser = null; // No user selected yet (intentional absence)
console.log(selectedUser);      // Output: null
console.log(typeof selectedUser); // Output: object (This is a historical bug in
JS, null is a primitive type)
```

Symbol (ES6+):

A unique and immutable primitive value. Used for unique object property keys to avoid naming collisions. (More advanced, we'll touch on this later if needed).

BigInt (ES2020):

Represents integers with arbitrary precision. Used for numbers larger than Number can safely handle (i.e., beyond 2^53 - 1). (More advanced, not commonly used in basic web dev).

Non-Primitive Data Type (represent collections of values, and are mutable):

Object:

A complex data type that allows you to store collections of data. Arrays and functions are special types of objects. Objects are fundamental in JavaScript.

```
let person = { // An object with properties
    firstName: "John",
    lastName: "Doe",
    age: 30
};
let colors = ["red", "green", "blue"]; // An Array (which is a type of object)
console.log(typeof person); // Output: object
console.log(typeof colors); // Output: object
```

The Console: Your Debugging Friend The browser console is an essential tool for JavaScript developers. It allows you to:

- See the output of your code.
- Check for errors.
- Interact with your code dynamically.
- Debug your programs.

You can open the console in most browsers by pressing F12 (or right-clicking on a page and selecting "Inspect" or "Inspect Element", then navigating to the "Console" tab).

console.log(): The most commonly used console method. It prints whatever you pass to it onto the console.

```
console.log("Hello from JavaScript!"); // Prints a string
let myAge = 30;
console.log("My age is:", myAge);    // Prints a string and a variable's value
console.log(10 + 5);                 // Prints the result of an expression
console.log(true);                   // Prints a boolean
```

Practice Set Day 1:

Variable & Data Type Exploration Test your understanding with these small exercises. You can open your browser console and type these directly or create a new