**Lesson 2: Variables and Data Types**

By the end of this lesson, students will be able to:

* Students will be able to identify and use various data types in JavaScript, including strings, numbers, booleans, objects, and arrays, and understand how to manipulate them effectively.
* Students will be able to declare variables using var, let, and const, and understand the scope and best practices for variable usage in different contexts within a JavaScript program.

**Variables**

Variables are fundamental in programming as they store data values that can be used and manipulated throughout your code. In JavaScript, you can create variables using three different keywords: ***var***, ***let***, and ***const***.

* ***var***
  + Scope: Function-scoped or globally scoped.
  + Usage: It is generally discouraged in modern JavaScript due to issues with scope and hoisting.

var name = "Alice";

console.log(name); // Output: Alice

* ***let***
  + Scope: Block-scoped (limited to the block, statement, or expression in which it is used).
  + Usage: Preferred for variables that may change their value.

let age = 25;

age = 26; // valid reassignment

console.log(age); // Output: 26

* ***const***
  + Scope: Block-scoped.
  + Usage: It is used for variables that should not be reassigned after their initial value is set. It does not make the value immutable, but it prevents the reassignment of the variable itself.

const PI = 3.14159;

console.log(PI); // Output: 3.14159

// Uncommenting the following line will result an error

// PI = 3.14; // Error: Assignment to constant variable

Key Points:

* Use let for variables that will change their value.
* Use const for variables that should remain constant.
* Avoid using var in modern JavaScript development due to its function scope and potential issues with variable hoisting.

**Data Types**

JavaScript supports several data types, which are categorized into primitive types and object types.

* Primitive Data Types

Primitive data types are the most basic data types and are immutable (they cannot be changed).

* Number: Represents both integer and floating-point numbers.

let score = 95; // Integer

let price = 19.99; // Floating-point Number

* String: Represents a sequence of characters enclosed in quotes.

let greeting = "Hello, world!";

let singleQuoteString = 'Single quotes are also valid.';

* Boolean: Represents one of two values: true or false.

let isStudent = true;

let hasGraduated = false;

* Undefined: It represents a variable that has been declared but has yet to be assigned a value.

let notDefined;

console.log(notDefined); // Output: undefined

* Null: Represents the intentional absence of any value. It is a primitive value that is often used to denote "no value."

let emptyValue = null;

console.log(emptyValue); // Output: null

* Object Data Types

Object data types are more complex and can hold collections of values.

* Object: Represents a collection of key-value pairs. Keys are strings (or Symbols), and values can be any data type.

let person = {

firstName: "John",

lastName: "Doe",

age: 30

};

console.log(person.firstName); // Output: John

* Array: Represents a list-like collection of values indexed by numbers. Arrays can hold elements of any data type.

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

console.log(colors[1]); // Output: green

Key Points:

* Primitive types are immutable and compared by value.
* Object types are mutable and compared by reference.