ASSIGNMENT 3: JAVASCRIPT

# Data Types and Variables

# Data Types and Variables:

* What are the different data types used in JavaScript variables in the provided code?

1. String – e.g., `let first\_name = 'Eluid Murithi';`

2. Number – e.g., `let myKiswahiliMarks = 67;`

3. Array – e.g., `let myRoom = ['bed', 'chair', 'gas cooker', 'table', 'tv'];`

4. Boolean – e.g., `let isAdmin = false;`

5. Object – e.g., `let countryInfo = { citizenShip: 'Kenyan', idNumber: 44455567 };`

6. Undefined – e.g., `let student;`

7. Null – e.g., `let age = null;`

* Explain the difference between var, let, and const in JavaScript.

- `var`: is a function-scoped keyword used to declare variables that can be redeclared or updated. E.g., `var first\_name = 'Eluid';`

- `let`: is a block-scoped, can be updated but not redeclared in the same scope. E.g., `let name = 5;`

- `const`: is a block-scoped, must be initialized and cannot be updated or redeclared. E.g., `const phoneNumber = 254789567364;`

* Why does JavaScript allow assigning different data types to the same variable?

JavaScript is dynamically typed, meaning variables can change types at runtime. This means that variables in JavaScript do not have a fixed data type; instead, their type is determined at runtime based on the value that is assigned to them. For example:

let name = 5; // initially a number  
name = "John"; // can change to a string

* How does JavaScript handle variables declared but not initialized? Illustrate with an example from the code.

JavaScript handles variables declared without initialization as `undefined`. E.g., `let student;` has no value, so `console.log(student)` outputs `undefined`.

* Discuss the significance of variable names in programming and how they are used in JavaScript.

Variable names are essential for identifying and storing data values in a program. They should be descriptive to indicate what kind of data they hold, making the code easier to read and maintain. For example, `myRoom` clearly refers to a room and its contents.

# Numeric Data Types

* What are the various numeric data types used in JavaScript, as shown in the code?

- Integer - Integers are whole numbers. As such integers have no fraction  
 e.g., `let myKiswahiliMarks = 67;`

- Float/Double A double or a float is a numeric data type that contains a fraction. Fractions are shown using a decimal point inside a number. In JavaScript, a float is a 32-bit type with single-precision, while double is a 64-bit type with double-precision. Float uses less memory but offers less precision, while double uses more memory but provides higher precision.   
`let bankBalance = 23.78;`

- Infinity – Infinity is a special value that represents a number that is greater than any other number.  
 e.g., `let yearsInHeaven = Infinity;`

* Explain the difference between integers, doubles, and Infinity in JavaScript with examples.

- Integer are whole numbers without a decimal. E.g., `let myKiswahiliMarks = 67;`

- Double/Float are numbers with decimals. E.g., `let bankBalance = 23.78;`

- Infinity represents an unbounded value or a number that is greater than any other number.  
E.g., `let yearsInHeaven = Infinity;`

* How does JavaScript handle arithmetic operations involving different numeric data types?

In JavaScript, all numbers (integers, floats, etc.) are represented using the **IEEE 754 double-precision floating-point format**, which means that the language doesn't differentiate between integers and floating-point numbers internally. JavaScript treats them as the same data type: **Number**.When performing arithmetic operations involving different numeric values, JavaScript handles them as follows:

1. Single Number Type: - JavaScript does not have separate types for integers and floating-point numbers. All numeric values are represented as floating-point numbers, so operations between integers and floats follow floating-point arithmetic rules.

For example:

let a = 5; // Integer  
 let b = 2.5; // Float  
 let result = a + b; // result is 7.5

2. Arithmetic with Mixed Numeric Types: - Even if the operands are integers and floating-point numbers, JavaScript will perform the operations using floating-point arithmetic.

For example:

let x = 10; // Integer  
 let y = 3.5; // Float

let sum = x + y; // 13.5  
 let product = x \* y; // 35.0  
 let division = x / y; // ~2.8571 (floating-point result)

3. Integer Division: - Even when performing division between two integers, JavaScript returns a floating-point number, unless the result is exactly an integer.

let result = 10 / 4; // 2.5 (floating-point result)  
 let exactResult = 12 / 4; // 3 (integer result but stored as a number)

4. Precision Limitations: - Since JavaScript uses floating-point arithmetic, precision errors can occur when performing operations on numbers with many decimal places.

let sum = 0.1 + 0.2; // Expected 0.3, but result is 0.30000000000000004

5. Special Numeric Values:

-Infinity: Dividing a non-zero number by `0` results in `Infinity` or `-Infinity` (based on the sign of the numerator).

let inf = 1 / 0; // Infinity  
 let negInf = -1 / 0; // -Infinity

- NaN (Not a Number): If an arithmetic operation cannot produce a meaningful result (e.g., dividing `0/0`), JavaScript returns `NaN`.

let nanResult = 0 / 0; // NaN  
 let invalidOperation = "string" \* 10; // NaN

6. Implicit Conversion in Operations: - JavaScript will attempt to convert non-numeric values into numbers when used in arithmetic operations. This is known as \*\*type coercion.

let result = "5" - 2; // result is 3 (string "5" is coerced to number 5)  
 let invalid = "five" \* 2; // result is NaN (invalid conversion)

# String Data Type

* How are strings represented in JavaScript?

Strings are sequences of characters enclosed in single (`''`) or double (`""`) quotes. E.g., `let sname = "John";`.

* Discuss the difference between declaring strings with single quotes ('') and double quotes ("") in JavaScript.

There is no functional difference between single and double quotes in JavaScript; it is mostly a style preference. Both are valid for defining strings.

* Explain why characters are automatically treated as strings in JavaScript.

In JavaScript, any character enclosed in quotes is automatically a string, even if it’s a single character. E.g., `let firstChar = 'A';`. This is because there is no any other way to differentiate characters from strings. Also, this reduces complexities in representing stings and characters.

# Boolean and Undefined Data Types:

* Explain the purpose of Boolean variables in JavaScript.

Booleans datatype is used to store true or false values in JavaScript. Booleans represent logical values: `true` or `false`. They are useful for controlling the flow of a program, such as in conditionals. E.g., `let isAdmin = false;`.

* Discuss the concept of undefined in JavaScript variables and provide examples from the code.

An `undefined` variable is one that has been declared but not assigned a value. . An undefined datatype is the default datatype assigned to JavaScript variables declared using var or let and which have not been initialized.

For example, `let student;` is `undefined` since it has no value

* How are Boolean variables useful in conditional statements and control flow in JavaScript?

Booleans control logic in conditionals, like:

if (isAdmin) {

console.log("Welcome, Admin");

}

In the code snippet above, the compiler checks if the isAdmin variable is set to true. If true then it prints on the console the message “Welcome Admin”. If isAdmin is not set to true then nothing is printed out on the console.

# Null Data Type

* Describe the significance of the null value in JavaScript.

`null` is a special value representing the intentional absence of any value. It differs from `undefined`, which means a variable has been declared but not assigned a value.

* Differentiate between null and undefined in JavaScript.

- `null`: Explicitly assigned to indicate "no value."

- `undefined`: A variable declared but not yet assigned a value.  
for example

let age = null; // no value intentionally assigned

let userName; // userName is not assigned to a value hence JS automatically assigns it undefined datatype.

# Object Data Type

* Explain how objects are represented in JavaScript.

Objects are collections of key-value pairs. To initialize an object use the variable declaration keyword const or let followed by = then {}. Data is then stored inside curly brackets in key-value pairs.  
  
 E.g., `let countryInfo = {   
 citizenShip: 'Kenyan',  
 idNumber: 44455567   
};`.

* Discuss the structure and purpose of the countryInfo object in the provided code.

`countryInfo` is a JavaScript Object which contains two properties/keys: `citizenShip` and `idNumber`. Each of this property has been assigned a value.

* How can objects be nested within other objects in JavaScript?

Objects can be nested within other objects. For example, `countryInfo` is included as part of the `info` object:

let info = {   
 fname: 'Titus',   
 sname: 'Kimutai',   
 countryInfo = {   
 citizenShip: 'Kenyan',  
 idNumber: 44455567   
 };  
 };

# Array Data Type:

* Describe the purpose and structure of arrays in JavaScript.

Arrays hold multiple values in an ordered list. Each element contained in an array is associated with an index, where index start at 0 and ends at the value –   
(length of the array – 1)  
E.g., `let myRoom = ['bed', 'chair', 'gas cooker'];`.

* Provide examples from the code demonstrating arrays containing different data types.

Arrays can contain values of different types.  
  
E.g., `let moreInfo = [countryInfo, marks, info];`.

* Discuss the concept of "array of arrays" and its significance.

An array can contain other arrays, allowing for complex data structures. An **array of arrays** in JavaScript, commonly referred to as a **2D array** or a **matrix**, is an array where each element is itself another array. This structure allows you to store data in a grid-like fashion, often used to represent tables, matrices, or coordinate grids.  
E.g.,

let matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

# Variable Naming Conventions:

* What are the conventions for naming variables in JavaScript?

- Variable names should be descriptive and written in camelCase. camelCamel notation is one in which first letter of the first word should start with a small letter while the second word should start with a capital letter   
E.g., `let phoneNumber = 254789567364;`.

* Discuss the importance of choosing meaningful and descriptive variable names.

Descriptive names like `myRoom` or `bankBalance` make the code more readable and easier to understand.

* Identify any variable naming conventions followed or violated in the provided code.

`first\_name` uses snake\_case instead of camelCase, which is more typical in JavaScript.

# Constants in JavaScript

* Explain the use of const keyword in JavaScript

`const` is used to declare variables whose values cannot be changed. Additionally, a variable declared using const must be assigned a value.  
E.g., `const phoneNumber = 254789567364;`.

* Discuss why reassigning a value to a constant variable results in an error.

Reassigning a value to a constant variable results in an error:

phoneNumber = 345564734893; // Uncommenting will throw an error

* Provide examples from the code demonstrating the declaration and use of constants.

const phoneNumber = 254789567364;