1. DATATYPES AND VARIABLES.

Primitive data types.

* Numbers- represents both integers and floating point numbers
* Strings- meant for textual data
* Boolean- usually represents logical values i.e true or false

Non Primitive data types.

* Object- An object is a collection of key value pairs.
* Arrays
* Function

2 DIFFERENCE BETWEEN VAR, LET AND CONST

* Const- meant for variables that cannot be reassigned or redeclared.
* Let-meant for variables that can be reassigned but cannot be redeclared.
* Var- meant for variables that can be reassigned and re-declared.

3. WHY DOES JAVASCRIPT ALLOW ASSIGNING DIFFERENT DATA TYPES TO THE SAME VARIABLE?

* Dynamic typing- In dynamically typed languages such as JS variables do not usually have fixed types.
* Type coercion and flexibility- Since JS aims to be user friendly it allows code to be written without the need to declare strict types for variables.

4.HOW DOES JAVASCRIPT HANDLE VARIABLES DECLARED BUT NOT INITIALIZED? ILLUSTRATE WITH AN EXAMPLE FROM THE CODE.

In JavaScript, if a variable is declared but not initialized, the variable is normally assigned the value undefined. It usually shows that the variable exists but hasn't been given a value.

let x; // Declared but not initialized

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

x = 10; // Now initialized

console.log(x); // Output: 10

5.DISCUSS THE SIGNIFICANCE OF VARIABLE NAMES IN PROGRAMMING AND HOW THEY ARE USED IN JAVASCRIPT.

* Variable names provide clarity: They make it easier to understand what the variable is being used for.

Example: totalAmount, userAge, customerID convey the purpose of the variables clearly.

* Maintenance: It helps during maintenance of the code since other developers get to easily understand what certain code is meant to do.

6.NUMERIC DATATYPES IN JAVASCRIPT

* Number- represents both integers and floating point numbers.
* BigInt- represents integers beyond the range of number.

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

* Integers- These are usually whole numbers without a decimal point. Eg let x=3;
* Doubles- these are numbers with decimal points. Eg let x = 3.14;
* Infinity - This is a special value that results from operations like dividing by zero or exceeding the representable numeric range

8.HOW DOES JAVASCRIPT HANDLE ARITHMETIC OPERATIONS INVOLVING DIFFERENT NUMERIC DATA TYPES?

* **Integer and Floating-Point Number**

When performing operations between an integer and a floating-point number, JavaScript converts the result to a floating-point number, if necessary

* **Integer and Integer**

When you perform arithmetic operations between two integers, JavaScript returns an integer (if the result is a whole number) or a floating-point number (if the result has a decimal)

* **Floating-Point and Floating-Point**

Arithmetic between two floating-point numbers yields a floating-point result.

* **Division by Zero**

If you divide a number by zero in JavaScript, you don’t get an error; instead, you get either Infinity or -Infinity depending on the sign of the number.

9.HOW ARE STRINGS REPRESENTED IN JAVASCRIPT?

Strings are data types used to represent textual data. They are a sequence of characters enclosed within either **single quotes (')**, **double quotes (")**, or **backticks (`)**.

10.DISCUSS THE DIFFERENCE BETWEEN DECLARING STRINGS WITH SINGLE QUOTES ('') AND DOUBLE QUOTES ("") IN JAVASCRIPT.

11.EXPLAIN WHY CHARACTERS ARE AUTOMATICALLY TREATED AS STRINGS IN JAVASCRIPT.

* Unified Data Type for Text: JavaScript has a single data type, String, to represent any sequence of characters, whether it's one character or multiple characters.
* Ease of Use: By treating characters as strings, JavaScript provides a consistent and simple way to handle text.
* Memory efficiency- Since Javascript is designed to be simple and flexible, managing separate data types for single characters versus strings would add complexity and potentially use more memory since JavaScript would have to keep track of both datatypes.

12.FUNCTION OF BOOLEAN VARIABLES IN JAVASCRIPT.

* They are used in conditional statements.
* Loops
* Boolean expressions

13. DISCUSS THE CONCEPT OF UNDEFINED IN JAVASCRIPT VARIABLES AND PROVIDE EXAMPLES FROM THE CODE.

Undefined is a special value that is normally assigned to a variable that has been declared but has not been assigned any value.

Eg let a=””;

Console.log(a);

14. HOW ARE BOOLEAN VARIABLES USEFUL IN CONDITIONAL STATEMENTS AND CONTROL FLOW IN JAVASCRIPT?

If statements. The if statement evaluates a boolean condition, and if it is true, the block of code inside the if statement is executed. If it is false, the block is skipped.

eg let showered = true;

if (showered) {

console.log("Good.");

} else {

console.log("Bad.");

}

// Output: "Good."

**else if and else Statements**: These statements are used to chain multiple conditions together. If the first condition evaluates to false, the program moves to the next condition, and so on.

let temperature = 30;

if (temperature > 35) {

console.log("It's very hot.");

} else if (temperature > 20) {

console.log("The weather is pleasant.");

} else {

console.log("It's cold.");

}

// Output: "The weather is pleasant."

 **while Loops**: A while loop continues to execute as long as its condition is true. This is useful when you want to repeat a block of code until a certain condition changes to false.

Example:

javascript

Copy code

let counter = 0;

while (counter < 5) {

console.log("Counter is at: " + counter);

counter++; // Increments the counter

}

// Output:

// Counter is at: 0

// Counter is at: 1

// Counter is at: 2

// Counter is at: 3

// Counter is at: 4

 **for Loops**: The condition in a for loop is evaluated before each iteration. If the condition evaluates to true, the loop continues. If it evaluates to false, the loop stops.

Example:

javascript

Copy code

for (let i = 0; i < 3; i++) {

console.log("i is: " + i);

}

// Output:

// i is: 0

// i is: 1

// i is: 2

 **Logical Operations**: Boolean variables are often the result of logical operations, such as && (AND), || (OR), and ! (NOT). These operations allow you to combine or invert conditions to create more complex control flows.

* **AND (&&)**: Both conditions must be true.

javascript

Copy code

let hasID = true;

let isAbove18 = true;

if (hasID && isAbove18) {

console.log("You can enter.");

} else {

console.log("You cannot enter.");

}

// Output: "You can enter."

* **OR (||)**: At least one condition must be true.

javascript

Copy code

let hasTicket = false;

let isVIP = true;

if (hasTicket || isVIP) {

console.log("You can attend the event.");

} else {

console.log("You cannot attend the event.");

}

// Output: "You can attend the event."

* **NOT (!)**: Negates a condition (i.e., makes true into false, and false into true).

javascript

Copy code

let isLoggedIn = false;

if (!isLoggedIn) {

console.log("Please log in.");

}

// Output: "Please log in."

 **Comparison Operators**: Boolean variables often result from **comparison operators** like ==, ===, !=, !==, >, <, >=, and <=. These operators compare two values and return true or false.

Example:

javascript

Copy code

let age = 18;

if (age >= 18) {

console.log("You are an adult.");

} else {

console.log("You are a minor.");

}

// Output: "You are an adult."

 **Ternary Operator (? :)**: This is a shorthand way to write simple if-else statements. It evaluates a condition and returns one of two values based on whether the condition is true or false.

Example:

javascript

Copy code

let isMember = true;

let message = isMember ? "Welcome, member!" : "Please sign up.";

console.log(message);

// Output: "Welcome, member!"

 **Boolean Functions**: Functions can also return boolean values, which are then used in conditional logic to control flow.

Example:

javascript

Copy code

function isEven(number) {

return number % 2 === 0;

}

let num = 4;

if (isEven(num)) {

console.log(num + " is even.");

} else {

console.log(num + " is odd.");

}

// Output: "4 is even."

15.DESCRIBE THE SIGNIFICANCE OF THE NULL VALUE IN JAVASCRIPT.

Null- is a special value used to represent the **intentional absence of any object value**

16. DIFFERENTIATE BETWEEN NULL AND UNDEFINED IN JAVASCRIPT

Null- This is used  when a variable or object property is intentionally set to "no value" or "nothing."

Eg let obj = null;

Undefined- assigned by JavaScript to a variable that has been declared but not yet initialized.

let x;

console.log(x);

17.EXPLAIN HOW OBJECTS ARE REPRESENTED IN JAVASCRIPT.

An **object** is a data structure used to store collections of key-value pairs. It allows you to group related data (such as properties) and functions (called methods) under a single entity.

let person = {

name: "John",

age: 30,

greet: function() {

console.log("Hello!");

}

};

18. HOW CAN OBJECTS BE NESTED WITHIN OTHER OBJECTS IN JAVASCRIPT?

Nesting objects means that an object can contain other objects as values of its properties. This capability is useful for organizing and managing related data in a structured way.

eg let library = {

name: "City Library",

address: {

street: "123 Main St",

city: "Springfield",

zip: "12345"

},

books: [

{ title: "1984", author: "George Orwell" },

{ title: "To Kill a Mockingbird", author: "Harper Lee" }

],

staff: {

manager: { name: "Emma", age: 45 },

assistants: [

{ name: "Jake", age: 30 },

{ name: "Sophia", age: 25 }

]

}

};

19.DESCRIBE THE PURPOSE AND STRUCTURE OF ARRAYS IN JAVASCRIPT.

* Storing Collections of Data:

Arrays are used to store multiple values in a single variable, which is particularly useful when you need to work with lists or sequences of items.

* Index-Based Access:

Arrays allow you to access elements using numeric indices, which makes it easy to retrieve, update, and manipulate individual elements.

* Iteration:

Arrays support iteration methods that simplify processing each element, making it easier to perform operations on collections of data.

20. DISCUSS THE CONCEPT OF "ARRAY OF ARRAYS" AND ITS SIGNIFICANCE.

An "array of arrays" in JavaScript is a data structure where each element of an array is itself an array. This concept allows you to create multidimensional arrays, which are useful for representing more complex data structures like matrices or tables.

21. RULES WHEN DECLARING VARIABLES

* Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( $ ) sign.
* After first letter we can use digits (0 to 9), for example value1.
* JavaScript variables are case sensitive, for example x and X are different variables.

22. DISCUSS THE IMPORTANCE OF CHOOSING MEANINGFUL AND DESCRIPTIVE VARIABLE NAMES.

* Variable names provide clarity: They make it easier to understand what the variable is being used for.

Example: totalAmount, userAge, customerID convey the purpose of the variables clearly.

* Maintenance: It helps during maintenance of the code since other developers get to easily understand what certain code is meant to do.

23. EXPLAIN THE USE OF CONST KEYWORD IN JAVASCRIPT

- meant for variables that cannot be reassigned or redeclared

24. DISCUSS WHY REASSIGNING A VALUE TO A CONSTANT VARIABLE RESULTS IN AN ERROR.

When you declare a variable with const, you are creating a constant binding to a value. This means that the reference to the value is fixed and cannot be changed to point to a different value.

25.

