-:JAVASCRIPT:-

🔹 1. What is dt and how many data types are there?

Data Types (dt) refer to the different types of values a variable can store.
JavaScript has **7 primitive** and **1 non-primitive** data type.

Primitive Data Types (Immutable)

1. **String** → "Hello"
2. **Number** → 123, 4.56
3. **Boolean** → true, false
4. **BigInt** → 12345678901234567890n
5. **Undefined** → let x;
6. **Null** → let y = null;
7. **Symbol** → Symbol('unique')

Non-Primitive Data Type (Mutable)

- **Object** → { name: "John", age: 25 }
- Arrays, Functions, Dates are also objects.

🧪 **Fun Example:**

```
let superhero = "Batman"; // String

let powerLevel = 9000; // Number

let isHuman = false; // Boolean

let bigNumber = 12345678901234567890n; // BigInt

let gadget; // Undefined

let enemy = null; // Null

let symbol = Symbol("uniquePower"); // Symbol

let batCar = { brand: "Batmobile", speed: "500km/h" }; // Object


console.log(typeof superhero, typeof powerLevel, typeof isHuman);
```

🔹 2. What is a variable? Global and Local Scope?

Variable:

A variable is a container for storing data values.

Scope:

- **Global Scope**: Accessible from anywhere in the script.
- **Local Scope**: Accessible only inside a function or block.

🧪 **Fun Example:**

```
let globalHero = "Superman"; // Global variable


function showHero() {
  let localHero = "Flash"; // Local variable
  console.log("Inside function:", localHero);
}


console.log("Outside function:", globalHero);
console.log(localHero); // ❌ Error: Not defined outside function
```

🔹 3. Keywords to Declare Variables

JavaScript provides 3 ways:

1. var (Function-scoped)
2. let (Block-scoped)
3. const (Block-scoped, Immutable)

🔹 4. Difference Between var, let, and const

Feature	**var**	**let**	**const**
Scope	Function-scoped	Block-scoped	Block-scoped

Feature	var	let	const
Hoisting	✅ Yes	✅ Yes	✅ Yes
Reassignable	✅ Yes	✅ Yes	❌ No
Redeclarable	✅ Yes	❌ No	❌ No

🧪 **Fun Example:**

```
var a = 10;
let b = 20;
const c = 30;

a = 15; // ✅ Allowed
b = 25; // ✅ Allowed
c = 35; // ❌ Error: Assignment to constant variable
```

🔹 5. Declaring and Assigning Variables

```
let city;        // Declaration
city = "Gotham"; // Assignment

let country = "USA"; // Declaration & Assignment in one line
```

🔹 6. Ways to Create Variables (Example for Each Data Type)

```
let name = "Bruce Wayne"; // String
let age = 35; // Number
let isRich = true; // Boolean
let bigValue = 9007199254740991n; // BigInt
let gadget; // Undefined
let enemy = null; // Null
let skill = Symbol("Martial Arts"); // Symbol
```

let car = { brand: "Batmobile", speed: "500km/h" }; // Object

---

🔹 7. Ways to Generate Output

1. console.log("Hello, Batman!");
2. document.write("Hello, Batman!");
3. alert("Hello, Batman!");
4. prompt("Enter your name:");
5. confirm("Are you sure?");
6. document.getElementById("demo").innerHTML = "Hello!";

🔹 8. Operators in JavaScript

Types of Operators

- **Arithmetic** → +, -, *, /, %, **
- **Comparison** → ==, ===, !=, !==, >, <, >=, <=
- **Logical** → &&, ||, !
- **Assignment** → =, +=, -=, *=, /=
- **Bitwise** → &, |, ^, <<, >>
- **Ternary** → condition ? expr1 : expr2

🧪 **Fun Example:**

```
let batmanStrength = 90;
let supermanStrength = 100;

console.log(batmanStrength > supermanStrength ? "Superman Wins!" : "Batman Wins!");
```

🔹 9. Conditional Statements

if Statement

```
let speed = 120;
if (speed > 100) {
  console.log("Speeding! Slow down!");
}
```

if...else Statement

```
let age = 16;
if (age >= 18) {
  console.log("You can vote.");
} else {
  console.log("You cannot vote.");
}
```

switch Statement

```
let hero = "Batman";

switch (hero) {
  case "Batman":
    console.log("Gotham needs me!");
    break;
  case "Superman":
    console.log("I can fly!");
    break;
  default:
    console.log("Unknown hero");
}
```

🔹 10. Difference Between == and ===

Operator	Type Checking	Example
==	Checks only values	"5" == 5 → ✅ true
===	Checks values & types	"5" === 5 → ❌ false

🔹 11. String to Number Conversion

```
let str = "123";

let num1 = Number(str); // Using Number()

let num2 = parseInt(str); // Using parseInt()

let num3 = +str; // Using + operator
```

🔹 12. Loops in JavaScript

For Loop

```
for (let i = 1; i <= 5; i++) {

   console.log(`Bat-Signal ${i} sent!`);

}
```

While Loop

```
let i = 1;

while (i <= 5) {

   console.log(`Warning! Joker is on the loose!`);

   i++;

}
```

Do-While Loop

```
let power = 5;

do {

   console.log(`Power Level: ${power}`);

   power--;

} while (power > 0);
```

🔹 13. Creating HTML Elements Using JS

```
let para = document.createElement("p");

para.textContent = "I'm Batman!";
```

```
document.body.appendChild(para);
```

🔹 14. innerHTML vs innerText

```
element.innerHTML = "<b>Bold Text</b>"; // Renders as bold
element.innerText = "<b>Bold Text</b>"; // Displays "<b>Bold Text</b>"
```

15. Difference Between Pre & Post Increment/Decrement

Operation	Pre (++x / --x)	Post (x++ / x--)
Increments/Decrements First	✅ Yes	❌ No
Returns Updated Value	✅ Yes	❌ No (Returns old value first)

🧪 **Fun Example:**

```
let x = 5;
console.log(++x); // 6 (Pre-increment: Increases first, then returns)
console.log(x++); // 6 (Post-increment: Returns first, then increases)
console.log(x);   // 7 (Now incremented)
```

🔹 16. Difference Between while and do...while

Feature	while Loop	do...while Loop
Condition Check	At the beginning	At the end
Runs at least once	❌ No	✅ Yes

🧪 **Fun Example:**

```
let count = 3;
while (count < 3) {
  console.log("While Loop: Runs only if condition is true");
}
```

```
do {
   console.log("Do-While Loop: Runs at least once!");
} while (count < 3);
```

🔹 17. How to Apply Styles from JavaScript?

You can change CSS styles dynamically using JavaScript.

🧪 **Fun Example:**

```
let myDiv = document.getElementById("myDiv");
myDiv.style.color = "red"; // Change text color
myDiv.style.backgroundColor = "black"; // Change background
myDiv.style.fontSize = "20px"; // Increase font size
```

🔹 18. How to Insert and Delete Tags Using JavaScript?

Insert (Create and Append)

```
let newElement = document.createElement("p");
newElement.textContent = "I am a new paragraph!";
document.body.appendChild(newElement);
```

Delete an Element

```
document.body.removeChild(newElement); // Removes the element
```

🔹 19. How to Get Document Attributes in JavaScript?

To get attributes like class, id, etc.

🧪 **Fun Example:**

```
let element = document.getElementById("myElement");
console.log(element.getAttribute("class")); // Get class name
console.log(element.getAttribute("id")); // Get ID
```

🔹 20. What is a Function? Types of Functions

A **function** is a block of reusable code that performs a task.

Types of Functions

1. **Named Function**:

```
function greet() {
  console.log("Hello, JavaScript!");
}
greet();
```

2. **Anonymous Function**:

```
let sayHello = function() {
  console.log("Hello, World!");
};
sayHello();
```

3. **Arrow Function**:

```
const add = (a, b) => a + b;
console.log(add(5, 3));
```

4. **Immediately Invoked Function Expression (IIFE)**:

```
(function() {
  console.log("I run immediately!");
})();
```

🔹 21. How to Get and Set Attribute Values?

```
let button = document.getElementById("btn");

// Get attribute value
console.log(button.getAttribute("type"));

// Set attribute value
```

```
button.setAttribute("class", "btn-primary");
```

🔹 22. What is a Callback Function?

A **callback function** is a function passed as an argument to another function.

🧪 **Fun Example:**

```
function greet(name, callback) {
   console.log("Hello, " + name);
   callback();
}

function sayGoodbye() {
   console.log("Goodbye!");
}

greet("Alice", sayGoodbye);
```

🔹 23. Difference Between Named and Anonymous Functions

Type	Named Function	Anonymous Function
Has Name?	✅ Yes	❌ No
Can be called by name?	✅ Yes	❌ No, must be assigned to a variable

🧪 **Example:**

```
// Named function
function hello() {
   console.log("Hello!");
}
hello();
```

```
// Anonymous function
let greet = function() {
  console.log("Hi there!");
};
greet();
```

🔹 24. Difference Between querySelector and querySelectorAll

Feature	querySelector	querySelectorAll
Returns	First matching element	All matching elements (NodeList)
Can access multiple?	❌ No	✅ Yes

🧪 **Example:**

```
let firstParagraph = document.querySelector("p"); // Gets first <p>
let allParagraphs = document.querySelectorAll("p"); // Gets all <p>
```

25. Definition and Explanation of setInterval, clearInterval, setTimeout, and clearTimeout

---

1. setInterval()

Definition:
setInterval() is a built-in JavaScript function that repeatedly executes a given function **at fixed time intervals** (in milliseconds) until it is stopped using clearInterval().

Syntax:

```
let intervalID = setInterval(function, delay, param1, param2, ...);
```

- function → The function to execute repeatedly.
- delay → The time interval (in milliseconds).
- param1, param2, ... → Optional parameters to pass to the function.
- Returns an intervalID that can be used to stop execution.

✅ **Example of setInterval()**:

```
let count = 0;
let intervalID = setInterval(() => {
   console.log("Count:", count++);
   if (count > 5) {
      clearInterval(intervalID); // Stops the interval after 5 iterations
   }
}, 1000); // Runs every 1 second
```

🔹 **Output (Runs every second until count > 5):**

makefile

CopyEdit

```
Count: 0
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

---

2. clearInterval()

Definition:
clearInterval() is used to **stop an interval** that was started using setInterval().

Syntax:

```
clearInterval(intervalID);
```

- intervalID → The ID of the interval returned by setInterval().

✅ **Example of clearInterval()**:

```
let counter = 0;
let myInterval = setInterval(() => {
   console.log("Executing every 2 seconds:", counter++);
   if (counter === 3) {
      clearInterval(myInterval); // Stops execution after 3 times
```

```
        console.log("Interval stopped!");
    }
}, 2000);
```

◆ **Output:**

```
Executing every 2 seconds: 0

Executing every 2 seconds: 1

Executing every 2 seconds: 2

Interval stopped!
```

---

3. setTimeout()

Definition:
setTimeout() is a built-in JavaScript function that executes a given function **only once** after a specified delay.

Syntax:

```
let timeoutID = setTimeout(function, delay, param1, param2, ...);
```

- function → The function to execute once.
- delay → The time (in milliseconds) to wait before executing the function.
- param1, param2, ... → Optional parameters to pass to the function.
- Returns a timeoutID that can be used to stop execution before it happens.

✅ **Example of setTimeout()**:

```
setTimeout(() => {
    console.log("This message appears after 3 seconds!");
}, 3000);
```

◆ **Output (after 3 seconds delay):**

```
This message appears after 3 seconds!
```

---

4. clearTimeout()

Definition:
clearTimeout() is used to **cancel a timeout** before it executes.

Syntax:

```
clearTimeout(timeoutID);
```

- timeoutID → The ID of the timeout returned by setTimeout().

✅ **Example of clearTimeout()**:

```
let timeoutID = setTimeout(() => {
  console.log("This message will never appear!");
}, 5000);

clearTimeout(timeoutID); // Cancels the timeout before execution
```

🔹 **Output:**

(No output, as the timeout was cleared)

🚀 Summary Table

Function	Purpose	Runs Repeatedly?	Can be Stopped?	Example Use Case
setInterval()	Runs function at intervals	✅ Yes	✅ clearInterval()	Updating a clock, auto-refreshing data
clearInterval()	Stops an interval	❌ No	❌ No	Stopping a repeating animation or timer
setTimeout()	Runs function once after delay	❌ No	✅ clearTimeout()	Showing a popup after delay
clearTimeout()	Cancels a timeout before it runs	❌ No	❌ No	Preventing a delayed action

---

🎯 Real-World Example (Combining All Functions)

```
let count = 0;


// Set an interval that runs every second
```

```
let myInterval = setInterval(() => {
    console.log("Repeating every second:", count++);
    if (count === 5) {
        clearInterval(myInterval); // Stop interval after 5 times
        console.log("Interval stopped!");

        // Set a timeout to display a message after 3 seconds
        let timeoutID = setTimeout(() => {
            console.log("Timeout executed after interval stopped!");
        }, 3000);

        // Clear the timeout before it executes (optional)
        clearTimeout(timeoutID);
    }
}, 1000);
```

🔹 **Expected Output:**

```
Repeating every second: 0
Repeating every second: 1
Repeating every second: 2
Repeating every second: 3
Repeating every second: 4
Interval stopped!
```

(The timeout will not execute because we cleared it before execution.)

🎯 **Key Takeaways**

- setInterval() runs a function repeatedly at a set interval.
- clearInterval() stops a repeating interval.
- setTimeout() runs a function once after a delay.

- clearTimeout() cancels a scheduled timeout before execution.

💡 **Use setInterval() for repeating tasks like a clock, and setTimeout() for one-time delays like notifications!** ⏳ ⏰

26. How to Open a Particular Window Using JavaScript?

```
window.open("https://www.google.com", "_blank");
```

🔹 27. How to Open a Window Popup?

```
window.open("https://example.com", "popupWindow", "width=600,height=400");
```

🔹 28. What is document.hasFocus()?

It checks if the document is currently in focus.

🧪 **Example:**

```
if (document.hasFocus()) {
   console.log("Window is in focus");
} else {
   console.log("Window is not in focus");
}
```