JAVASCRIPT THEORY ASSIGNMENT

Javascript Introduction

1. **What is JavaScript? Explain the role of JavaScript in web development?**

   Ans. JavaScript was invented by Brendan Eich in the year 1995. The purpose for inventing this language was to make the website much more dynamic. This JavaScript is mainly used to add functionalities to the web documents.

   It is a dynamically type language. Interpreted language because it uses interpreter.

   Scripting language because it does line by line execution. Object oriented programming language. It is not only used in client-side activities but also in server side same like the other backend programming language.

   In the client side it is basically used to validate the user input.

   These are used to develop dynamic web pages, and user-interactive.

   JavaScript is executed by JavaScript engine which will be present in the browser.

2. **How is JavaScript different from other programming languages like Python or Java?**

   Ans. JavaScript:- Mainly used for web development to make web pages interactive (e.g., buttons, forms).

   - It runs in web browsers and on servers.
   - Lightweight and designed for real-time, dynamic web apps.

   Python:- General-purpose language used for data science, AI, backend development, and scripting.

   - It runs servers or locally.
   - Focuses on readability and simplicity, making it beginner-friendly.

   Java:- Used for large-scale applications like Android apps, enterprise software, and banking systems.

   - It runs on any device with a java virtual machine.
   - It is strict and structured, emphasizing performance and stability.

Main Difference:-

- JavaScript is mostly for the web; Python and Java are for broader tasks.
- JavaScript is more flexible but less strict compared to Java, while Python is known for clean and simple syntax.
- Java is faster for complex applications; JavaScript and Python prioritize simplicity and usability.

3. **Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?**

Ans. <script> is use for include JavaScript code within a web page. Or you can add JavaScript externally.

1. **Inline JavaScript**: You can write JavaScript directly between the opening and closing <script> tags. This method is simple but makes your HTML document less organized.

2. **Internal JavaScript**: You can place JavaScript in a <script> tag inside the <head> or <body> section of your HTML document.

3. **External JavaScript**: For better organization and reusability, you can link to an external JavaScript file using the src attribute in the <script> tag.

Linking an External JavaScript File

1. Save the JavaScript code in a file with the .js extension (e.g., script.js).
2. Use the <script> tag with the src attribute to include the file. Place it in the <head> or just before the closing </body> tag (recommended for performance).

Variables and Data Types

4. What are variables in JavaScript? How do you declare a variable using var, let, and const?

Ans. Variables is like containers used to store data in JavaScript**.**

- In JavaScript, you can declare a variable using var, let, or const.

1. **Using** var
   - Used in older JavaScript versions. It is function-scoped.
   - We can redeclared and reassign the value.

2. **Using** let
    - Introduced in modern JavaScript. It is block-scoped
    - We can reassign the value but not redeclare it.

3. **Using** const
    - Also introduced in modern JavaScript. It is block-scoped.
    - We cannot redeclare and reassign the value.

5. Explain the different data types in JavaScript. Provide examples for each.

ANS. Data types is used for the what type of data will be store in the variables.

- There are two types of data types primitive and non-primitive.

1. Primitive Data Types

These are simple, single values.

a) Number

- Used for any number, including integers or decimals.
- Example:

let age = 25 ---This is the integer

let price = 99.99 ---This is the decimal

b) String

- Used for text (a sequence of characters). Strings are enclosed in quotes (single, double, or backticks).
- Example:

```
let name = “sayar”;             double quotes
let last_name = ‘Patel’;        single quotes
let greet = `Hi, ${name}`;      Backticks for template strings
```

c) Boolean

- Boolean will return the true or false. Often used in conditions.
- Examples:

```
let isSunny = true;
let isRaining = false;
```

d) Undefined

- A variable that is declared but not assigned a value.
- Example:

let x; ---No value assigned
-output will be the undefined

e) Null

- Represents an empty or non-existent value (manually assigned).
- Example:
  let emptyValue = null;
  - Output will be the null

2. Non – primitive Data Types (Object Data Type)

- This data type will store the collections of data

a) Object

- A collection of key-value pairs.
- Example:

```
let person =
{
 name: "Alice",
age: 25,
isStudent: true
};
```

b) Array

- A special type of object to store lists of values.
- Example:
  let colors = ["red", "green", "blue"];

c) Function

- A block of reusable code.
- Example:

```
function greet() {
    return "Hello!";
}
greet()
```

6. What is the difference between undefined and null in JavaScript?

Ans.

undefined	Null
• Something has not been assigned a value.	• Represents "nothing" or any empty value.

• JavaScript sets it automatically.	• Sets by the programmer or manual
• A variable is declared but not assign a value.	• It indicates that a variable is empty or has no value.
• let x; - No value assign • **output**: - undefined	• let y=null; - Intentionally empty • **output: -** null

JavaScript Operators

7). What are the different types of operators in JavaScript? Explain with examples.

ANS.

1. **Arithmetic Operator**

Operator	Description	Example	Output
+	Addition	5+3	8
-	Subtraction	10-4	6
*	Multiplication	3*2	6
/	Division	10/2	5
%	Modulus	10%3	1
**	Power	2**3	8

2. **Assignment Operators**

Operator	Description	Example	Explanation
=	Assign value	x=10	x becomes 10
+=	Add and assign	x+=5	x = x + 5

-=	Subtraction and assign	x-=2	x = x - 2
*=	Multiply and assign	x*=3	x = x * 3
/=	Divide and assign	x/=2	x = x / 2

3. Comparison Operators

Operator	Description	Example	Output
==	Equal to	5 == “5”	true
===	Strict equal to checks type too)	5 === “5”	false
!=	Not equal to	5 != 3	true
>	Greater than	10 > 5	true
<	Less than	5 < 10	true
>=	Greater than or equal to	5>=5	true
<=	Less than or equal to	3<=5	true

4. Logical Operators

Operator	Description	Example	Output
&&	Logical AND	true && false	false
||	Logical OR	true || false	true
!	Logical NOT	!true	false

5. String operators

Operator	Description	Example	Output
+	Concatenation	“Hello” + “World”	Hello World
+=	Append text to variables	let x = “Hi”; x += “there!”	Hi there!

6. Increment and decrement operators

Operator	Description	Example	Output
x++	Post Increment by 1	let x = 5; x++	6
x--	Post Decrement by 1	let x = 5; x--	4

7. Ternary Operator

- Short hand of the if - else statement
- Syntax:
  Condition ? true block : false block

8. Type operators

Operator	Description	Example	Output
typeof	Checks the type of a value	typeof 42	Number

8) What is the difference between == and === in JavaScript?

==(Loose Equality)	===(Strict Equality)
• Compares only values	• Compare its value and data types
• Auto type conversion	• It does not perform type conversion
• (5 == “5”)----true	• (5 === “5”)---false

Control Flow (If-Else, Switch)

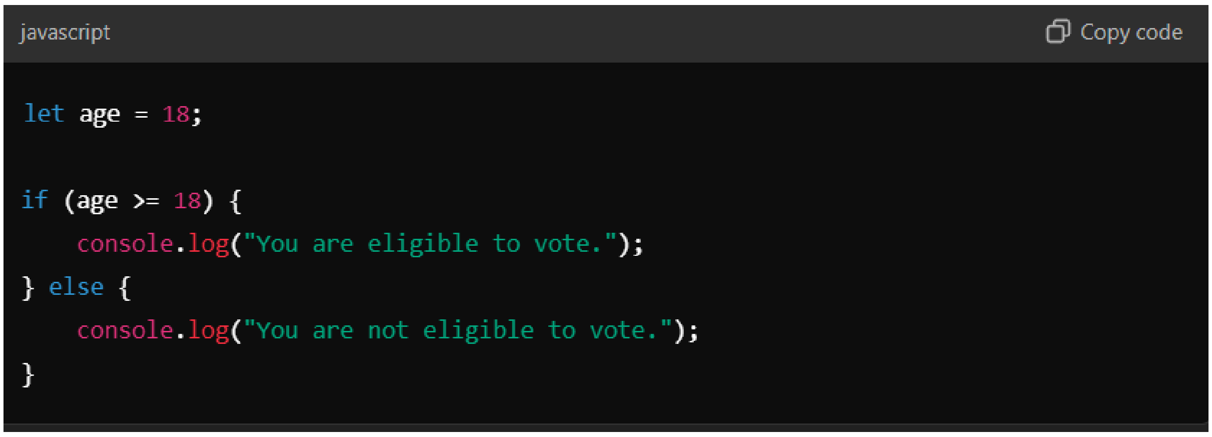
9) What is control flow in JavaScript? Explain how if-else statements work with an example.

ANS.

- Control flow refers to the order in which the computer executes the code in your program.
- Instead of running line by line, you can change the flow based on conditions or loops. This allows your program to make decisions or repeat actions.

How if – else statement work

- An if – else statement helps your code make decisions based on conditions.
- The if block runs the condition is true
- The else block runs when the condition is false.



```javascript
let age = 18;

if (age >= 18) {
    console.log("You are eligible to vote.");
} else {
    console.log("You are not eligible to vote.");
}
```

- **Explanation:**

- If the age is greater than or equal to 18. The true block is execute means if condition is true.
- If the age is less than 18, the program skip the if block and runs code in the else block

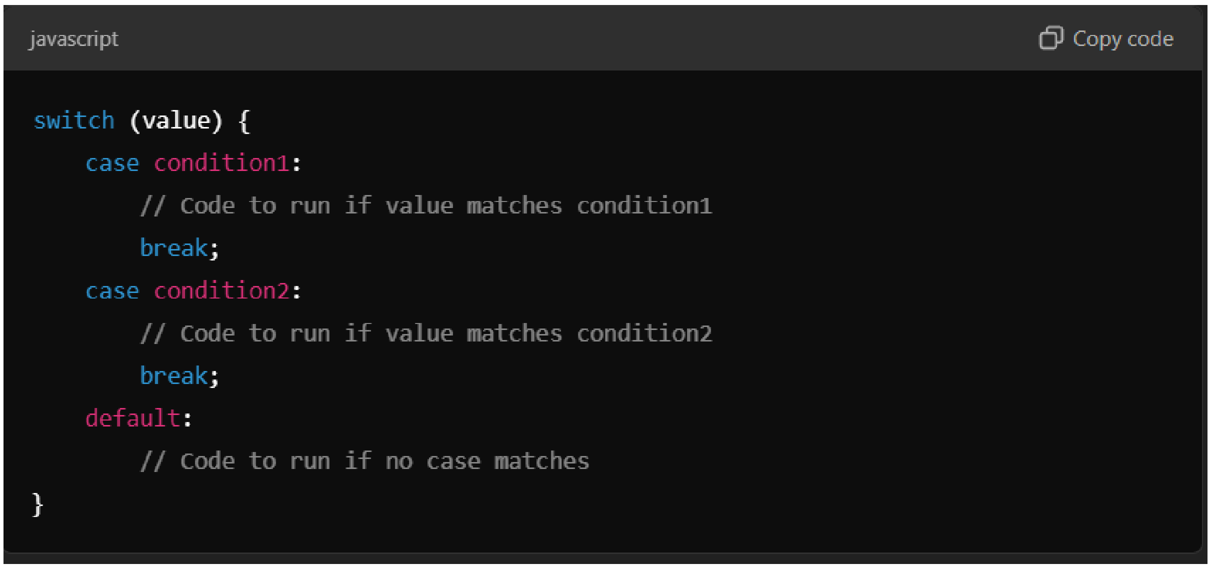
- **Using if – else statements gives your program logic to make based on conditions!**

10) Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

ANS.

A switch statement is another way to make decisions in your program, like if-else. It checks a value against multiple cases and runs the code for the matching case.

Syntax:



```javascript
switch (value) {
    case condition1:
        // Code to run if value matches condition1
        break;
    case condition2:
        // Code to run if value matches condition2
        break;
    default:
        // Code to run if no case matches
}
```

- The value in switch is compared with each case.
- The break keyword stops the execution after a match. Without break, all cases below the match will also run.
- The default block is optional and runs if no cases match.

- **Use** if – else
  - When conditions are based on ranges or comparisons.
  - Example: if (age > 18)
  - When you have only 1-2 conditions**.** Comparisons, ranges, or logical operators.
- **Use** switch
  - when checking for multiple possible exact values.
  - Example: switch (day) for days of the week.
  - When there are many cases to check. Single variable against multiple exact values.

Loops (For, While, Do-While)

11) Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

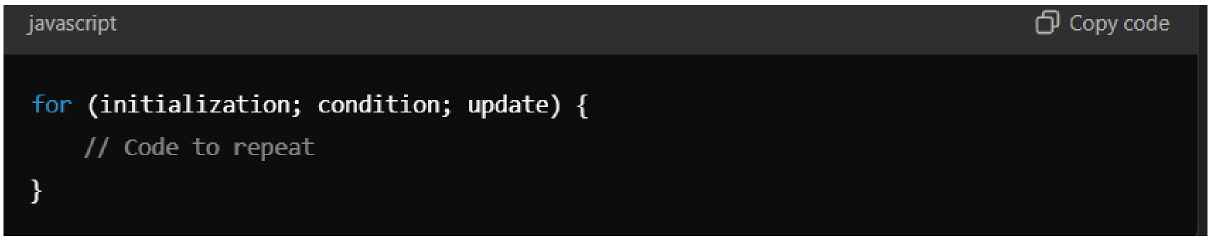
ANS. Loops in JavaScript allow you to repeat a block of code multiple times. There are three main types: for, while, and do-while.

1. for Loop

- A for loop is used when you know how many times you want to repeat the code. It has three parts:
- It is an entry control loop

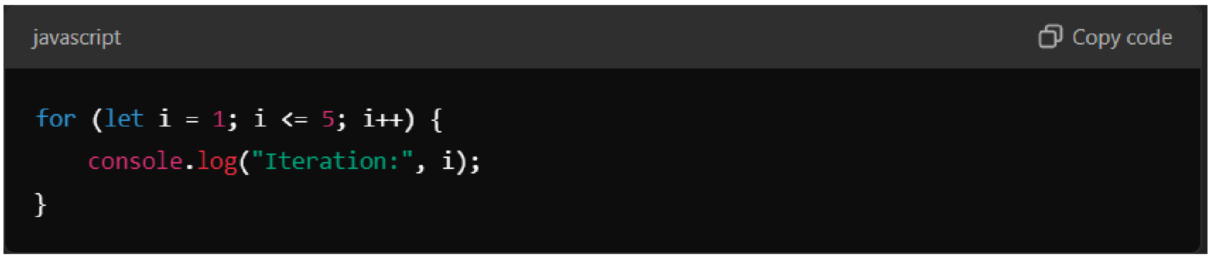
1. **Initialization**: Start the loop (e.g., set a variable).
2. **Condition**: Run the loop as long as this is true.
3. **Update**: Change the variable after each iteration.

Syntax:



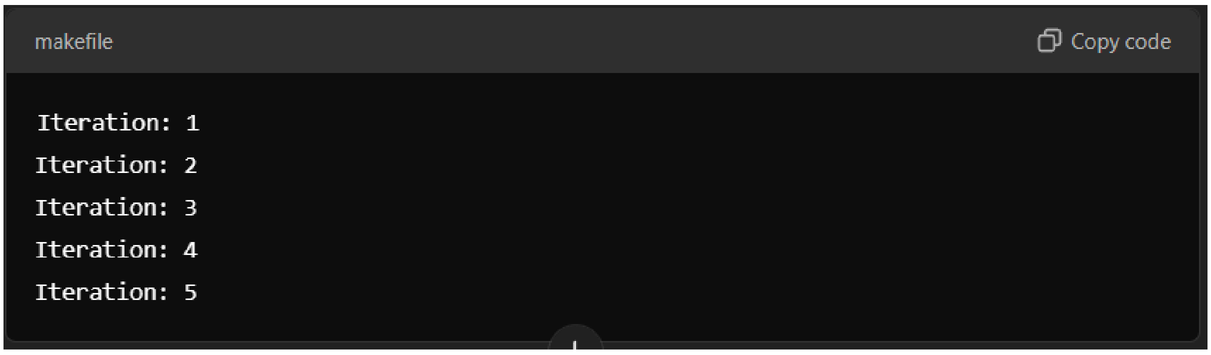
```javascript
for (initialization; condition; update) {
    // Code to repeat
}
```

Example:



```javascript
for (let i = 1; i <= 5; i++) {
    console.log("Iteration:", i);
}
```

Output:

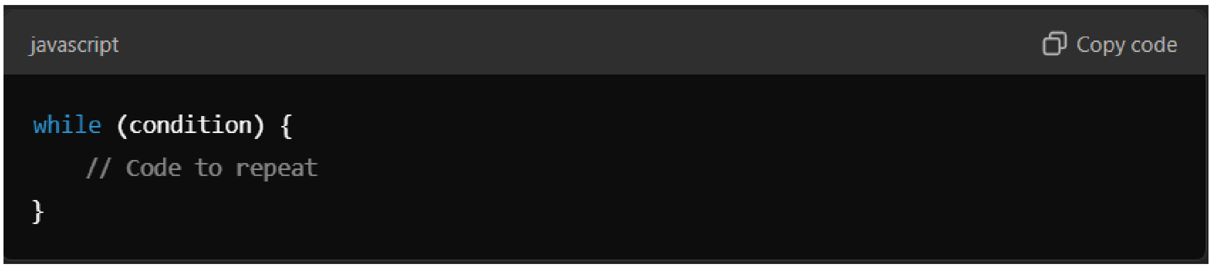


```
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
```

2. While Loop

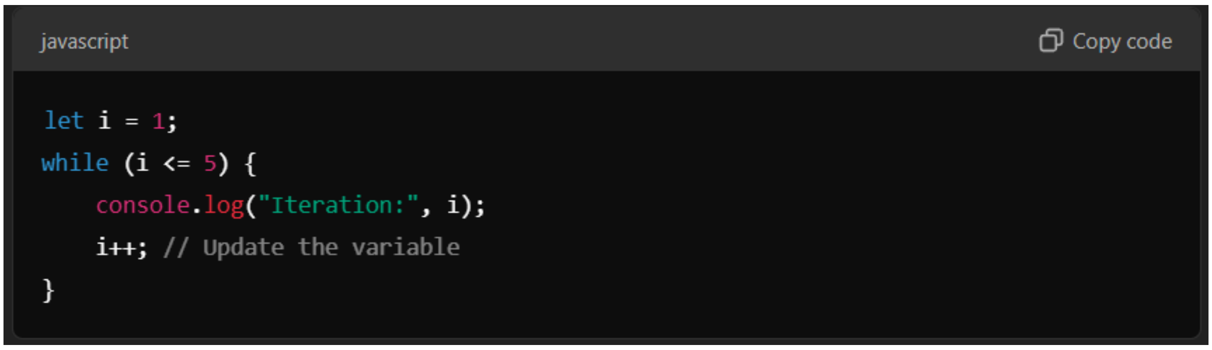
- A while loop runs as long as the condition is true. Use it when you don't know how many times the condition will execute.
- It is an entry control loop.

Syntax:



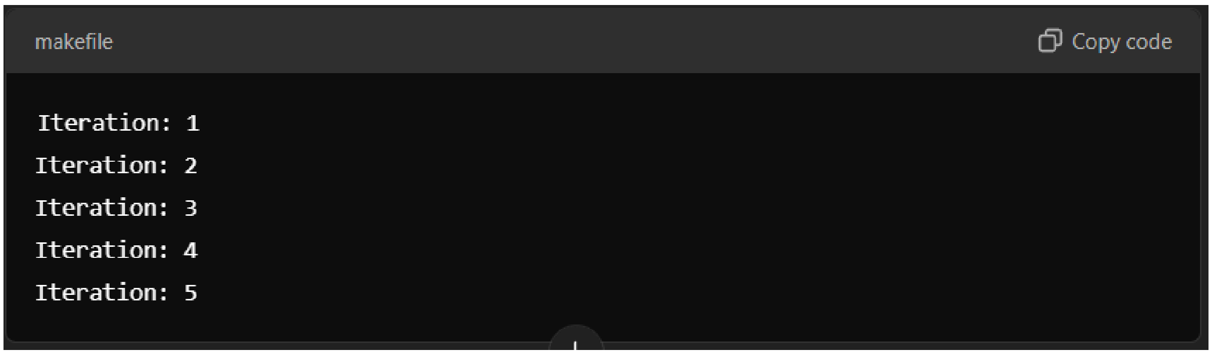
```javascript
while (condition) {
    // Code to repeat
}
```

Example:



```javascript
let i = 1;
while (i <= 5) {
    console.log("Iteration:", i);
    i++; // Update the variable
}
```

Output:

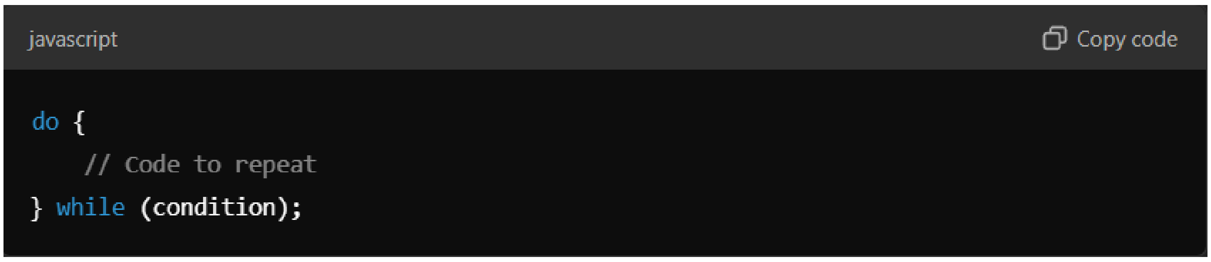


```makefile
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
```

3. do-while Loop

- A do-while loop is similar to while, but it always runs at least once, even if the condition is false. This is because the condition is checked after the loop runs.
- It is an exit control loop.

Syntax:



```javascript
do {
    // Code to repeat
} while (condition);
```

Example:

```javascript
let i = 1;
do {
    console.log("Iteration:", i);
    i++; // Update the variable
} while (i <= 5);
```

Output:

```makefile
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
```

Functions

12) What are functions in JavaScript? Explain the syntax for declaring and calling a function.

ANS. A function in JavaScript is a block of code designed to perform a specific task. Functions help you reuse code, making your programs easier to read, debug, and maintain.

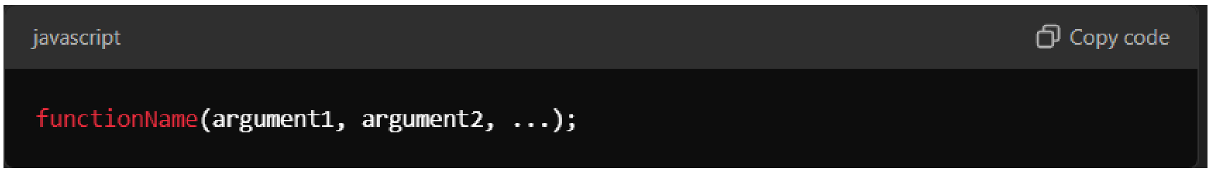
You declare a function once and call it whenever you need to perform the task.

Syntax:

```javascript
function functionName(parameter1, parameter2, ...) {
    // Code to be executed
}
```

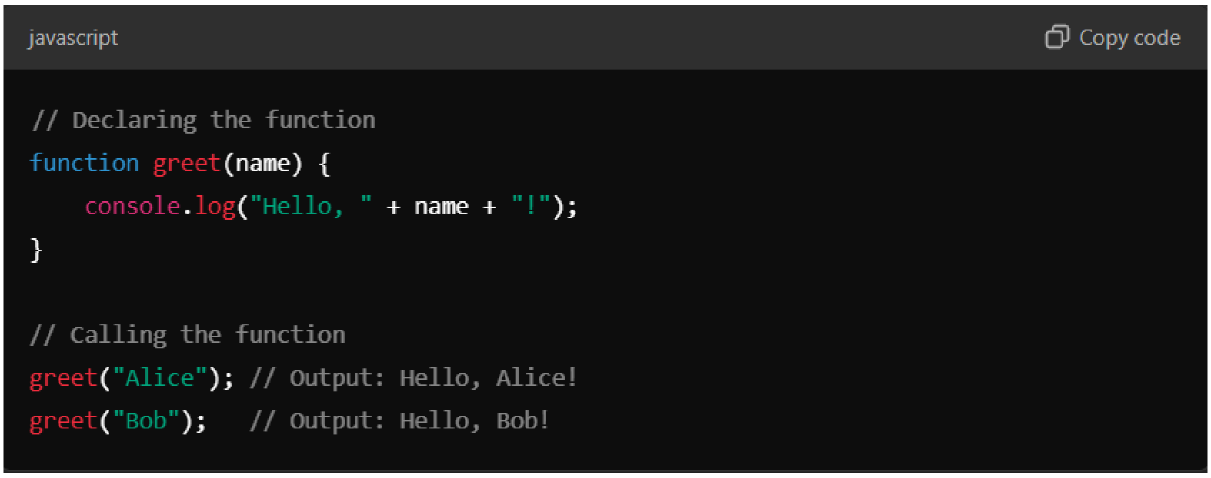
How to call a function

You "call" or "invoke" a function by writing its name followed by parentheses (). If the function has parameters, you provide the values (called **arguments**) inside the parentheses.



```javascript
functionName(argument1, argument2, ...);
```

Declaring and calling a function



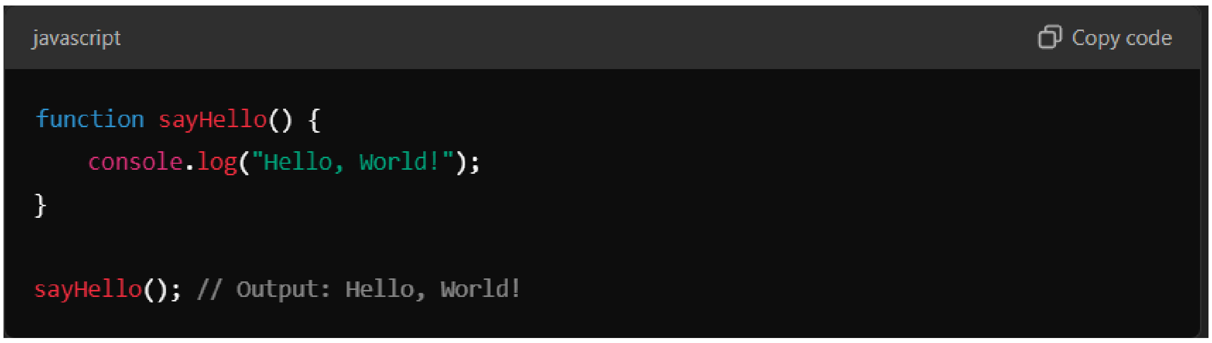
```javascript
// Declaring the function
function greet(name) {
    console.log("Hello, " + name + "!");
}

// Calling the function
greet("Alice"); // Output: Hello, Alice!
greet("Bob");   // Output: Hello, Bob!
```

Four types of function

1. **Function without parameters and without return type.**
2. **Function without parameters and with return type.**
3. **Function with parameters and without return type.**
4. **Function with parameters and with return type.**

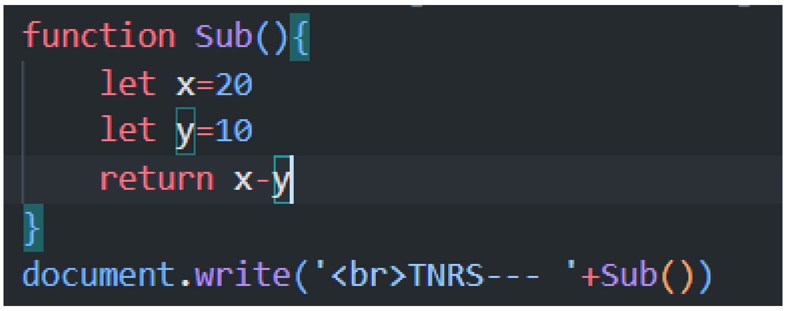
1. Function without parameters and without return type.



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

sayHello(); // Output: Hello, World!
```

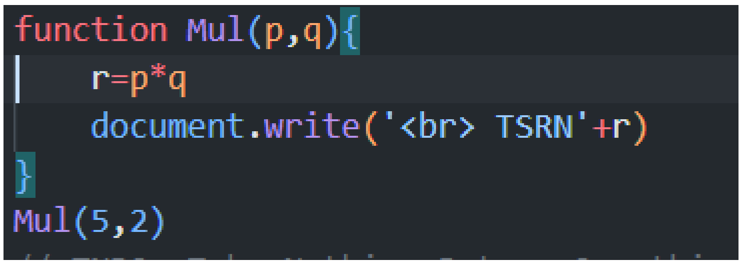
2. Function without parameters and with return type



```
function Sub(){
    let x=20
    let y=10
    return x-y
}
document.write('<br>TNRS--- '+Sub())
```

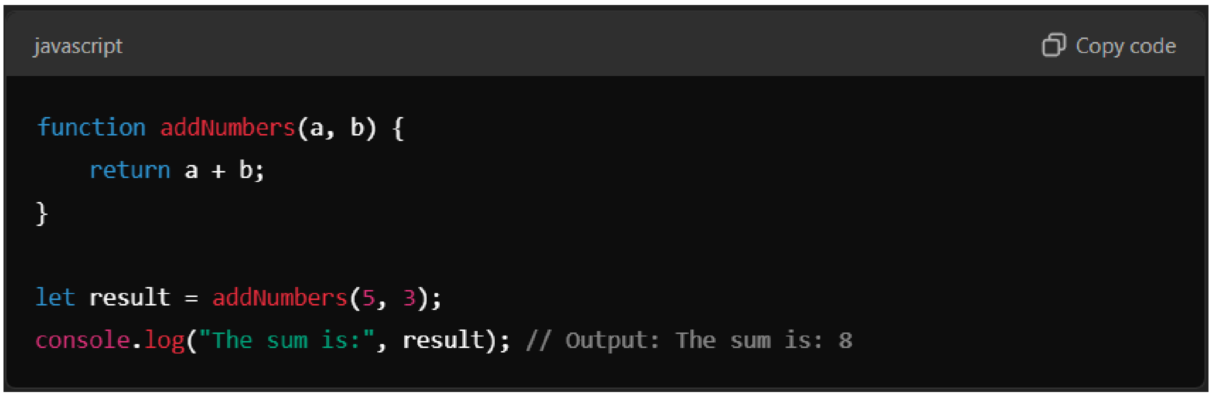
- When you use return type you have to print that function

3. **Function with parameters and without return type**



```
function Mul(p,q){
    r=p*q
    document.write('<br> TSRN'+r)
}
Mul(5,2)
```

4. **Function with parameters and with return type**



```javascript
function addNumbers(a, b) {
    return a + b;
}

let result = addNumbers(5, 3);
console.log("The sum is:", result); // Output: The sum is: 8
```

- A function is a reusable block of code.
- Declare a function with the function keyword.
- Call the function using its name followed by parentheses.

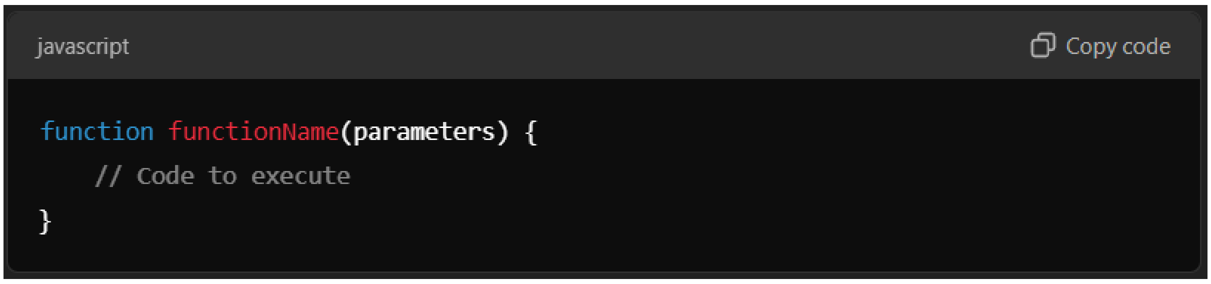
13) What is the difference between a function declaration and a function expression?

ANS.

1. **Function Declaration**

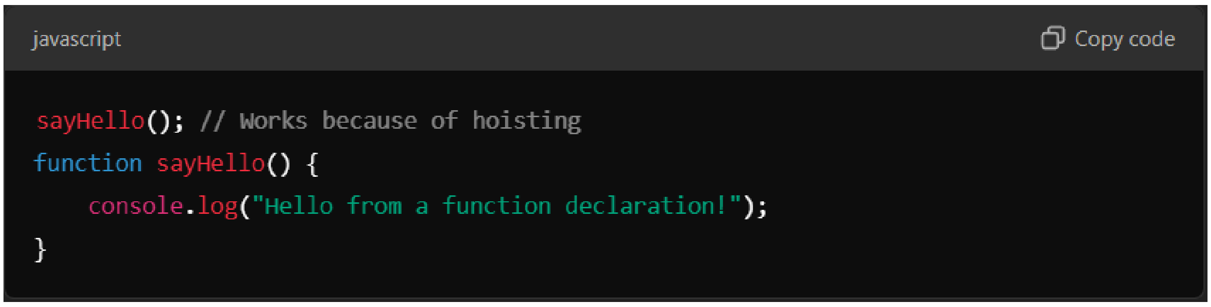
- A function that is declared using the function keyword and has a name.
- Function declaration are hoisted, meaning they are moved to the top of their scope during execution. You can call them before they are defined in the code.

Syntax:



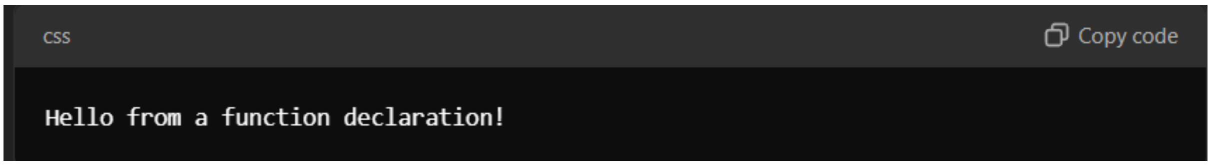
```javascript
function functionName(parameters) {
    // Code to execute
}
```

Example:



```javascript
sayHello(); // Works because of hoisting
function sayHello() {
    console.log("Hello from a function declaration!");
}
```

Output:

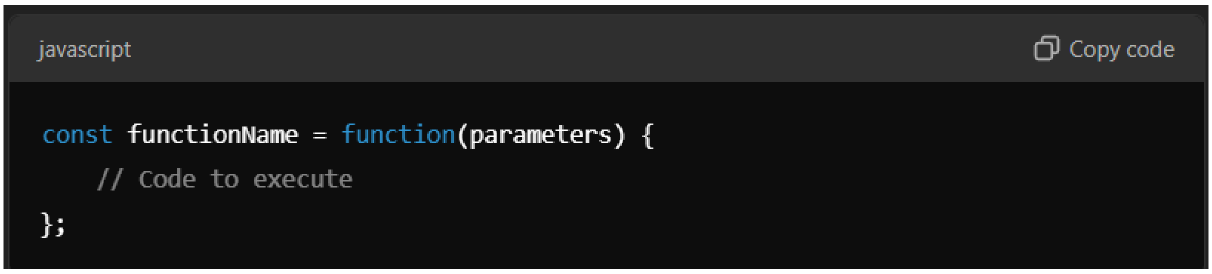


```css
Hello from a function declaration!
```

2. Function Expression

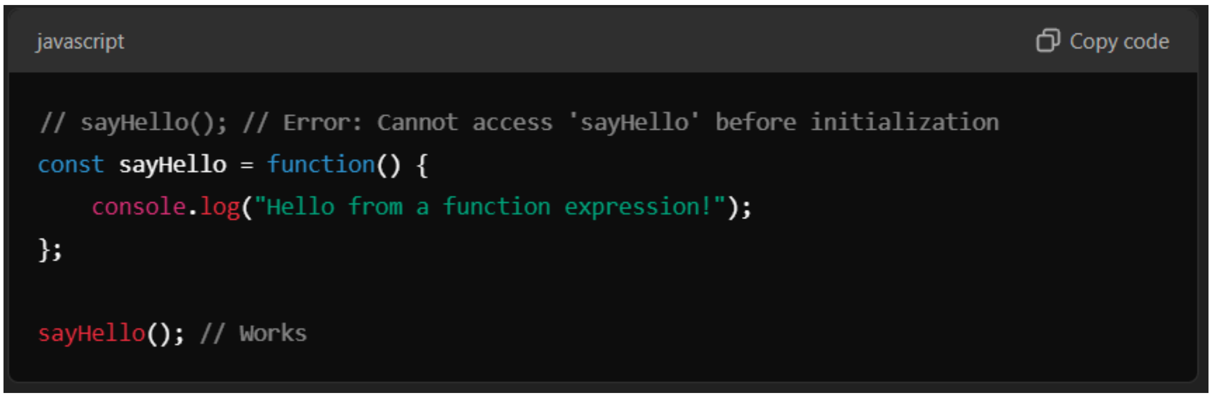
- A function defined as part of an expression. It can be
- **Anonymous**: Has no name.
- **Named**: Includes a name
- Function expressions are **not hoisted**, so you cannot call them before they are defined in the code.

Syntax:



```javascript
const functionName = function(parameters) {
    // Code to execute
};
```

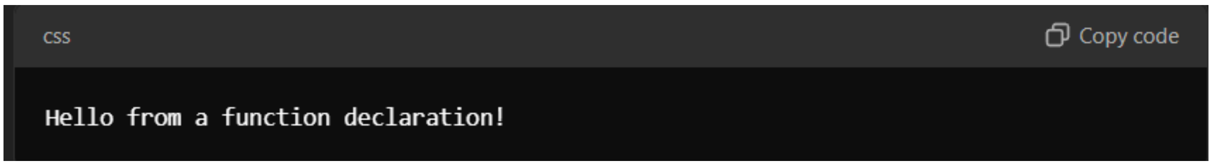
Example:



```javascript
// sayHello(); // Error: Cannot access 'sayHello' before initialization
const sayHello = function() {
    console.log("Hello from a function expression!");
};

sayHello(); // Works
```

Output:



```
Hello from a function declaration!
```

Function Declaration	Function expression
Starts with function and has a name	Assigned to a variable; can be anonymous or named.
Hoisted	Not hoisted
Reusable and globally accessible	More flexible, block scoped

14) Discuss the concept of parameters and return values in functions.

ANS.

1. Parameters

- Parameters are pass at the time of function definition. They act as placeholder for values also called arguments.
- Parameters allow you to pass data into a function, making it dynamic and reusable.
- Parameters are optional.

2. Return Values

- A return value is the output of a function.

- It will return where the function was called.
- This is done using return keyword.
- Once a return statement is executed, the function stops running.
- Any code after return is ignored.

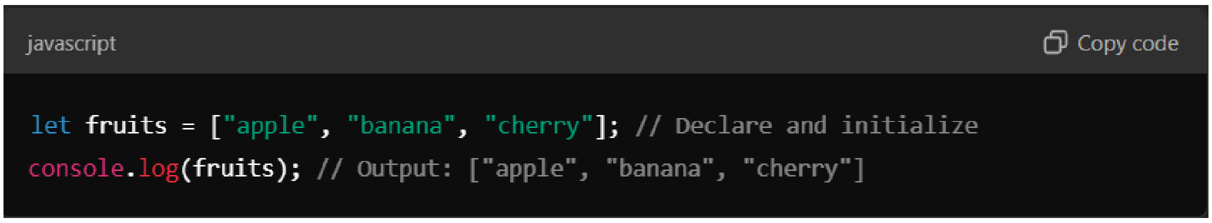
Arrays

15) What is an array in JavaScript? How do you declare and initialize an array?

ANS. An array is collection of a multiple values in a single variable.

- You can store any kind of data types for example number, string, objects
- Each element in an array has an index, starting from 0 for the first element.
- Using [] brackets you can create an array.

Example:



```javascript
let fruits = ["apple", "banana", "cherry"]; // Declare and initialize
console.log(fruits); // Output: ["apple", "banana", "cherry"]
```

16) Methods of Array.

Methods	Description
Push()	**Add element at the end of the array**
Pop()	**Remove the last element from the array**
Shift()	**Remove the first element from the array**
Unshift()	**Add the element at the first index**

Splice()	Add and remove the element from an array
Sort()	Sort the element of the array in descending order
reverse()	Reverse the array
Function(a,b){a-b}	Compare function

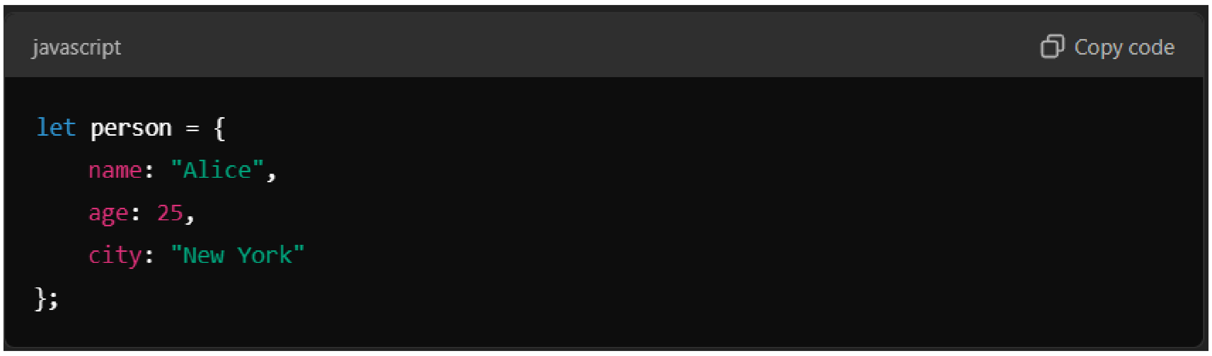
Objects

17) What is an object in JavaScript? How are objects different from arrays?

ANS. Object is a collection of data. Store the data in the form of key-value pairs.

- Where key is a string and the value can be any data type

Example:



```javascript
let person = {
    name: "Alice",
    age: 25,
    city: "New York"
};
```

- You can access object property using objectName.propertyName or objectName[“propertyName”]

Object	Array
Store data as key-value pairs.	Stored data as order list(index based).
Access by property name(key).	Access by index number.
Store the complex data	Store the list of items.

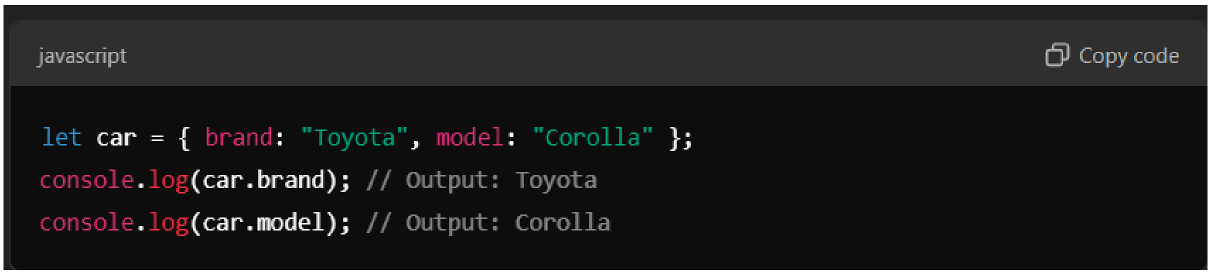
Properties describe the data meaning	**Elements not have descriptive labels that is index no.**
Example:{name:"dinga", age:25}	**Example: ["dinga", 25]**

18) Explain how to access and update object properties using dot notation and bracket notation.

ANS.

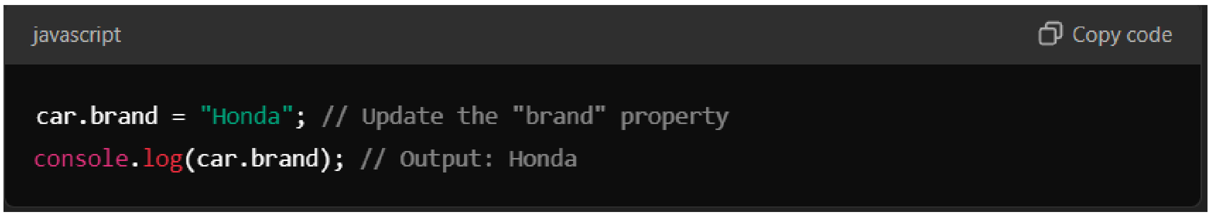
1. **Dot Notation**
   - Use a dot (.) followed by the property name to access or update the property.
   - Use dot notation when the property name is simple, valid identifier (no spaces, special characters, or variables).

Accessing a property:



```javascript
let car = { brand: "Toyota", model: "Corolla" };
console.log(car.brand); // Output: Toyota
console.log(car.model); // Output: Corolla
```

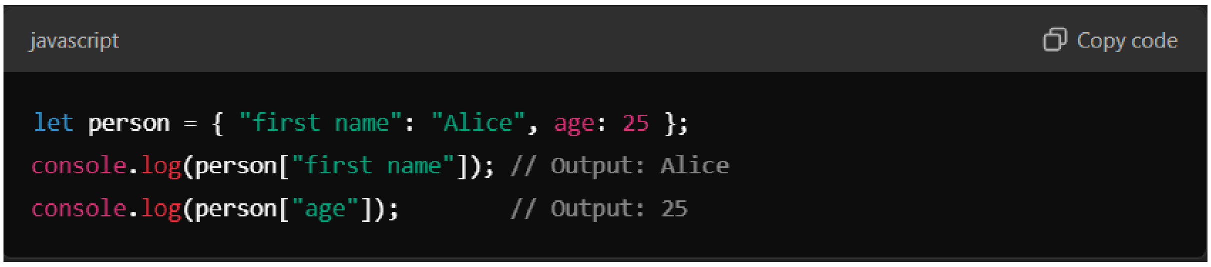
Updating a property:



```javascript
car.brand = "Honda"; // Update the "brand" property
console.log(car.brand); // Output: Honda
```

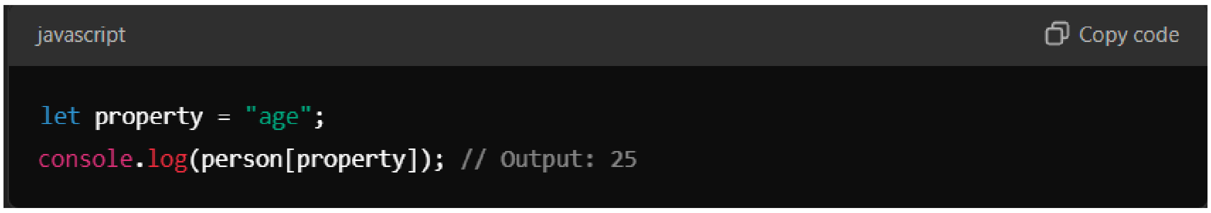
2. **Bracket Notation**
   - Use square brackets [] with the property name as a string to access or update the property**.**
   - when the property name contains spaces or special characters we can use bracket notation.
   - When the property name is store in a variable.

Accessing a property:



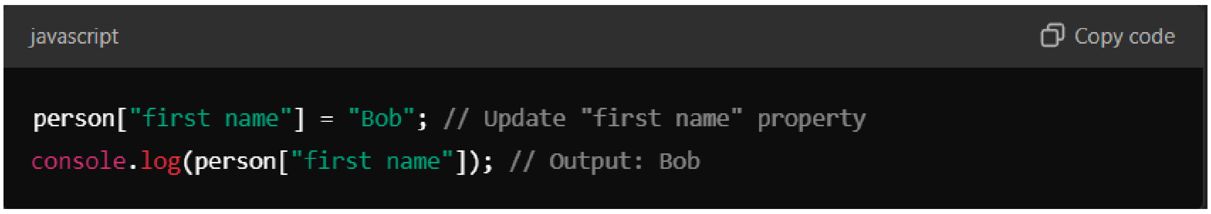
```javascript
let person = { "first name": "Alice", age: 25 };
console.log(person["first name"]); // Output: Alice
console.log(person["age"]);         // Output: 25
```

Using a variable as the property name:



```javascript
let property = "age";
console.log(person[property]); // Output: 25
```

Updating a property:



```javascript
person["first name"] = "Bob"; // Update "first name" property
console.log(person["first name"]); // Output: Bob
```

Dot Notation	Bracket Notation
Object.property	**Object["property"]**
When the property name is simple	**When the property name has spaces, special characters, or is a variable**
Person.name	**Person["first name"]**

JavaScript Events

19) What are JavaScript events? Explain the role of event listeners.

ANS. Javascript events are actions or occurrences that happen in the browser and can be detected by JavaScript.

- For example clicking a button, hovering over an element, typing into a text field.
- Events allow you to make web pages interactive by responding to user actions.
- JavaScript events detect user actions or change in the browser.
- Event listeners respond to these events by running specified function.
- Use addEventListener for clean, modern, and flexible way to handle events.

Example:

- On click event
- On double click
- On change event
- On submit

Event listeners:

- An **event listener** is a JavaScript function that "listens" for a specific event on an element and runs a callback function when that event occurs.

Adding event listeners using addEventListener Method

This method allows you to attach multiple event listeners to the same element.

Syntax:

```javascript
element.addEventListener(event, callbackFunction);
```

Example:

```javascript
// Select the button element
let button = document.querySelector("button");

// Attach a click event listener
button.addEventListener("click", function() {
    console.log("Button was clicked!");
});
```

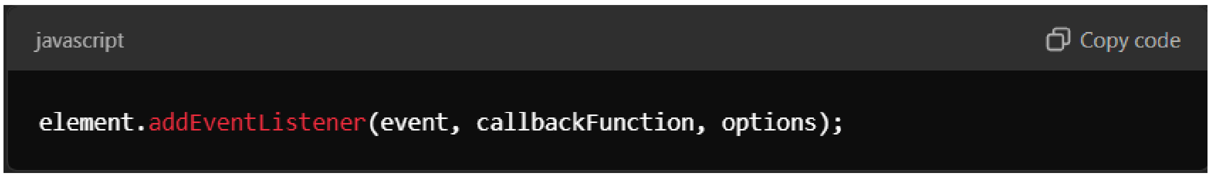
Adding event listeners using HTML attribute.

```html
<button onclick="alert('Button clicked!')">Click Me</button>
```

20) How does the addEventListener() method work in JavaScript? Provide an example.

ANS. The addEventListener() method in JavaScript allows you to attach an event listener to an element. This listener waits for a specific event (like a click, hover, or key press) to occur and then executes a callback function.

Syntax:



```javascript
element.addEventListener(event, callbackFunction, options);
```

- **Event:** The type of event you want to listen for, e.g., "click", "mouseover", "keydown", etc.
- **callbackFunction:** The function to execute when the event occurs.
- **Option(optional):** An object to control additional settings, like event capturing and passive behaviour.

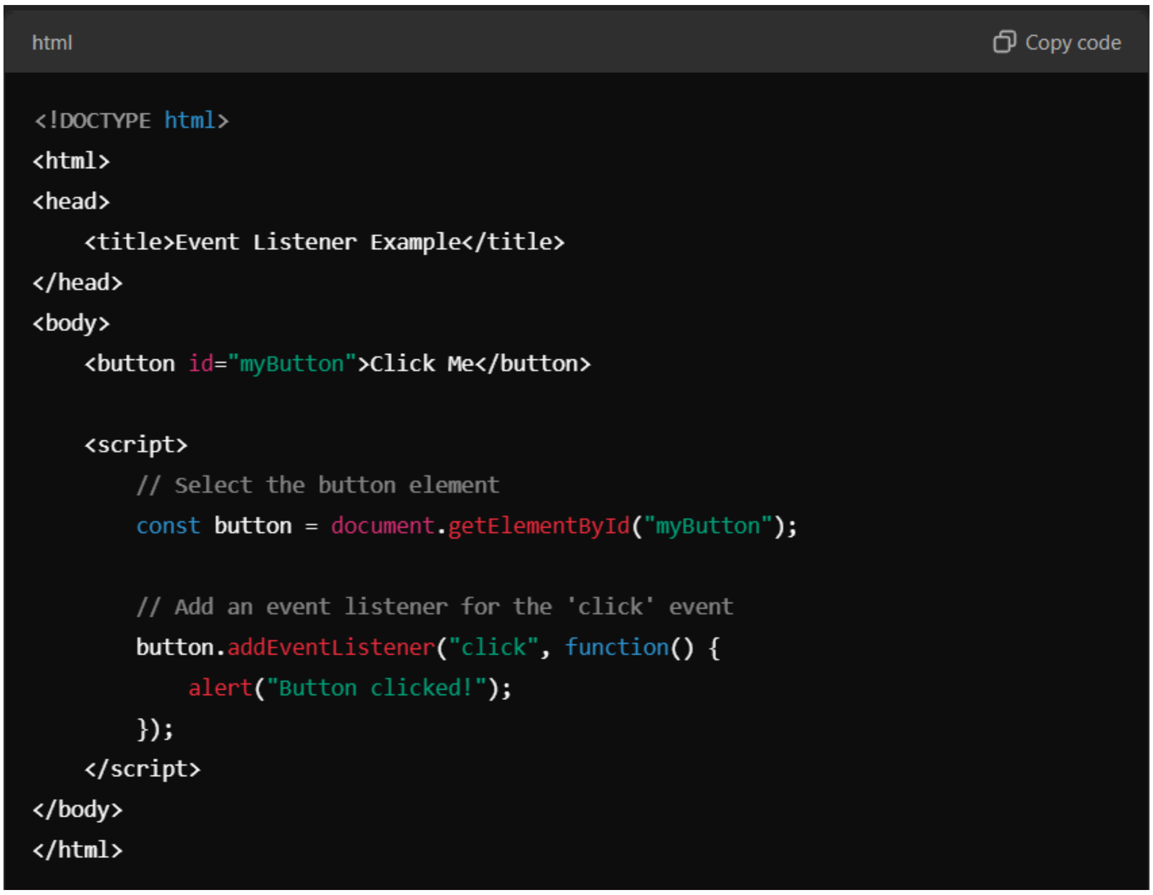
How it works:

1. **Attach the Listener:** Use addEventListener() to specify the event and the function to run.

2. **Wait for the Event:** JavaScript keeps "listening" for the specified event on the element.

3. **Trigger the Function**: When the event occurs, the callback function is executed.

Example: simple click event

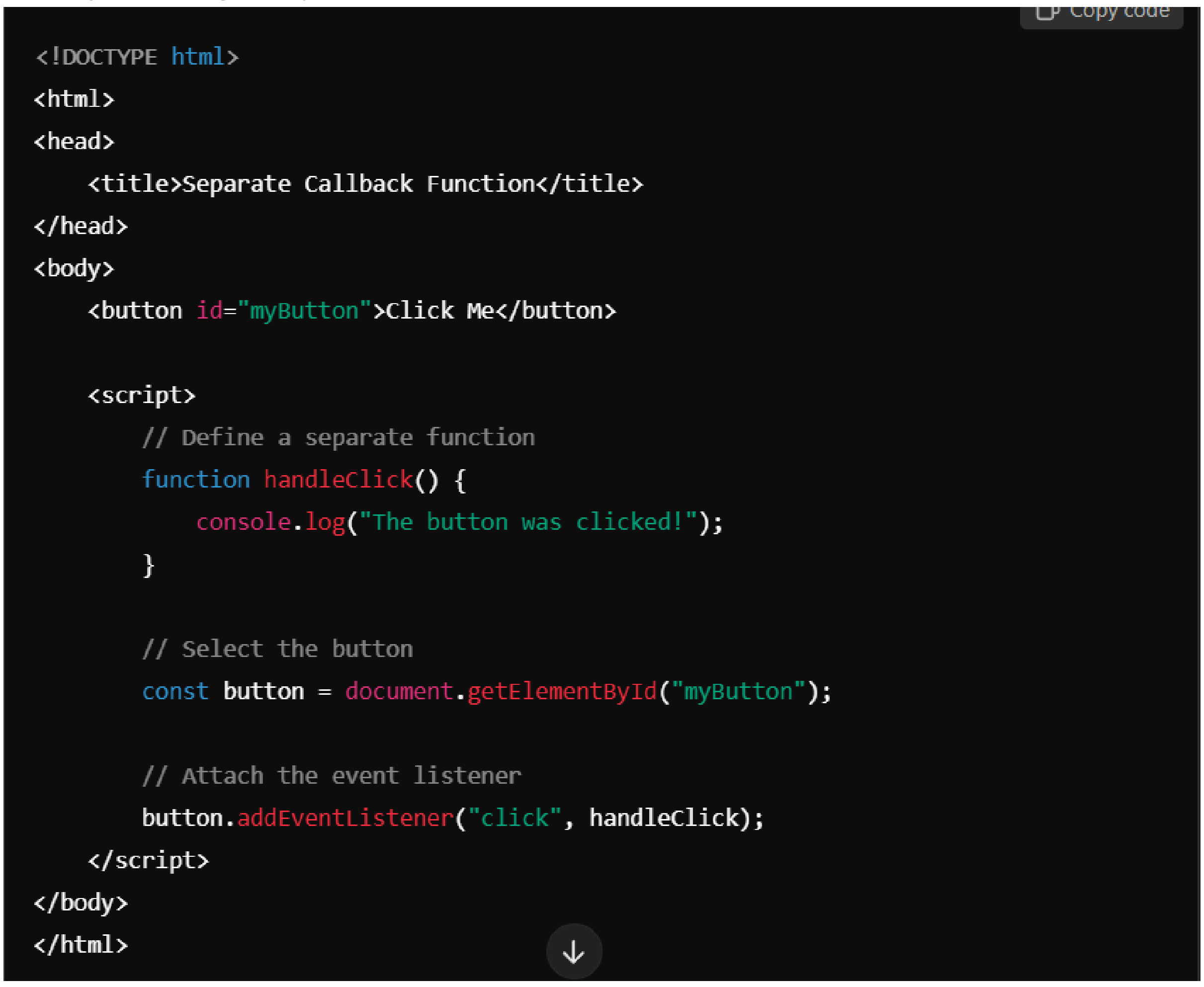


```html
<!DOCTYPE html>
<html>
<head>
    <title>Event Listener Example</title>
</head>
<body>
    <button id="myButton">Click Me</button>

    <script>
        // Select the button element
        const button = document.getElementById("myButton");

        // Add an event listener for the 'click' event
        button.addEventListener("click", function() {
            alert("Button clicked!");
        });
    </script>
</body>
</html>
```

Example: using a separate function



```html
<!DOCTYPE html>
<html>
<head>
    <title>Separate Callback Function</title>
</head>
<body>
    <button id="myButton">Click Me</button>

    <script>
        // Define a separate function
        function handleClick() {
            console.log("The button was clicked!");
        }

        // Select the button
        const button = document.getElementById("myButton");

        // Attach the event listener
        button.addEventListener("click", handleClick);
    </script>
</body>
</html>
```

DOM Manipulation

21) What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

ANS. The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of nodes, where each node corresponds to an element, attribute, or text in the document.

Using the DOM, JavaScript can interact with and manipulate the content, structure, and style of a web page dynamically**.**

How does the DOM works

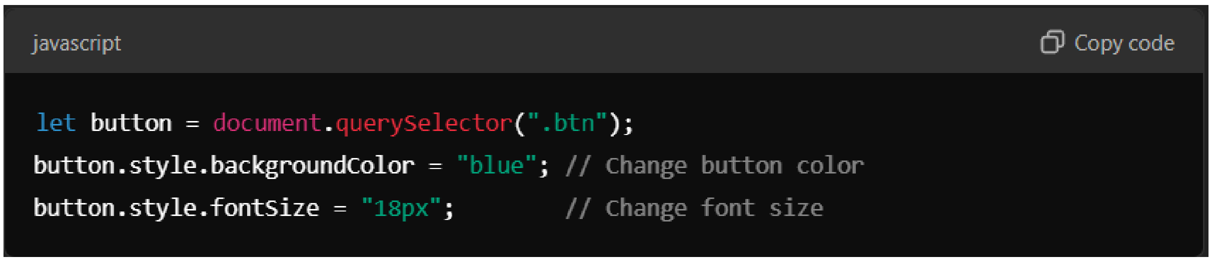
- The browser converts the HTML file into the DOM.
- The DOM represents the document as a tree structure, where each element is a node.
- JavaScript can use this structure to read, modify, add, or delete elements on the web page.

How does JavaScript interact with the DOM?

- **Selecting elements**
    - **getElementById()**: Selects an element by its id.
    - **getElementsByClassName()**: Selects elements by their class.
    - **querySelector()**: Selects the first element matching a CSS selector.
    - **querySelectorAll()**: Selects all elements matching a CSS selector.

- **Modifying Content**
    - **innerHTML**: Modifies the HTML content inside an element.
    - **textContent**: Modifies only the text inside an element.

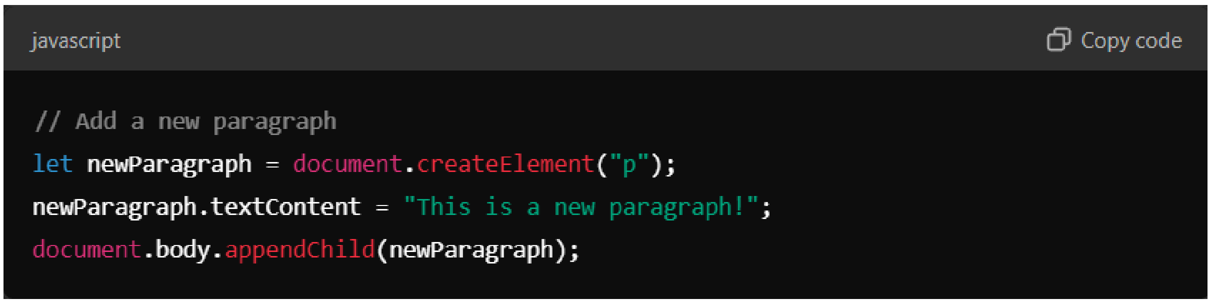
- **Changing Style**
    - JavaScript can change an element's CSS styles dynamically using the style property.



```javascript
let button = document.querySelector(".btn");
button.style.backgroundColor = "blue"; // Change button color
button.style.fontSize = "18px";        // Change font size
```

- **Adding and Removing Elements**
    - **createElement()**: Creates a new element.
    - **appendChild()**: Adds a child element.
    - **removeChild()**: Removes a child element.

Example:



```javascript
// Add a new paragraph
let newParagraph = document.createElement("p");
newParagraph.textContent = "This is a new paragraph!";
document.body.appendChild(newParagraph);
```

- **Adding Event Listeners**

```javascript
let button = document.querySelector(".btn");
button.addEventListener("click", function() {
    alert("Button clicked!");
});
```

- **The DOM Tree Structure**

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1 id="header">Welcome!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

```markdown
- Document
  - html
    - head
      - title
    - body
      - h1
      - p
```

- **The DOM is crucial because it:**

  - **Connects HTML with JavaScript**: Allows JavaScript to manipulate the structure, content, and styles of a web page.
  - **Enables Interactivity**: JavaScript can respond to user actions like clicks, key presses, and more through the DOM.

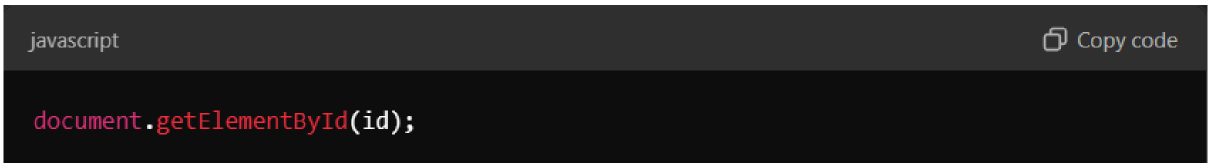
22) Explain the methods getElementById(), getElementsByClassName(),and querySelector() used to select elements from the DOM.

ANS.

1. getElementById()

- Selects a single element with a specific id attribute.
- Returns the first element (unique) with the matching id.
- Best for selecting a single, unique element.

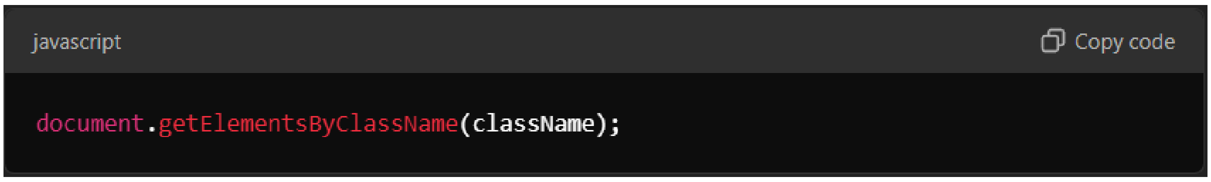
syntax:



```javascript
document.getElementById(id);
```

2. getElementsByClassName()

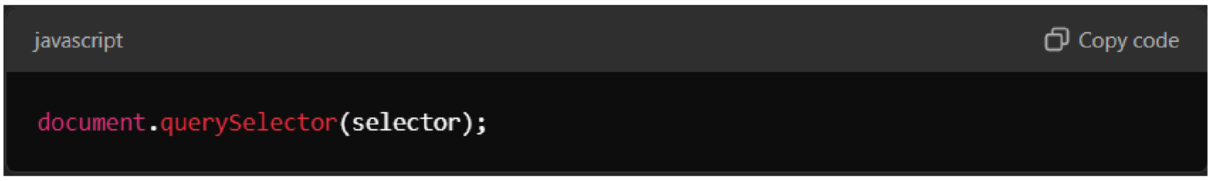
- Selects all elements with a specific class attribute.
- Returns a **live HTMLCollection** (similar to an array) of matching elements.
- Best for selecting multiple elements with the same class.



```javascript
document.getElementsByClassName(className);
```

3. querySelector()

- Selects the **first element** that matches a CSS selector (e.g., #id, .class, or element type).
- Returns a single element or null if no match is found.
- More flexible than getElementById() and getElementsByClassName() because it supports CSS-style selectors.



```javascript
document.querySelector(selector);
```

JavaScript Timing Events (setTimeout, setInterval)

23) Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?

ANS. These are two important JavaScript functions used to execute code after a delay or repeatedly at specified intervals. They are part of the Web APIs provided by the browser.

1. setTimeout()

- The setTimeout() function allows you to execute a piece of code once after a specified delay in milliseconds.

Syntax:

javascript Copy code

```javascript
setTimeout(callbackFunction, delay);
```

- **callbackFunction:** The function to be executed after the delay.
- **delay:** The time in milliseconds (1000 ms = 1 second) to wait before executing the function.

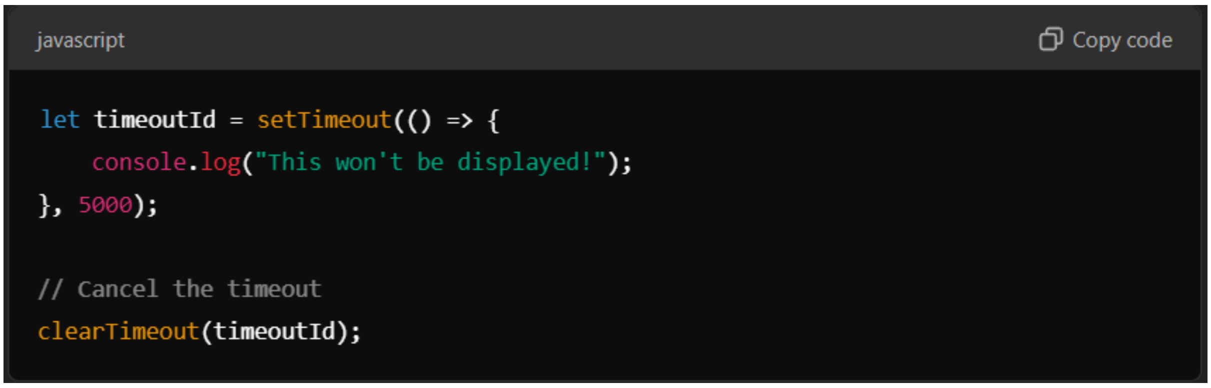
Example: Display a message After 3 seconds

javascript Copy code

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

Stopping a Timeout with clearTimeout():

If you need to cancel a timeout before it executes, use clearTimeout().

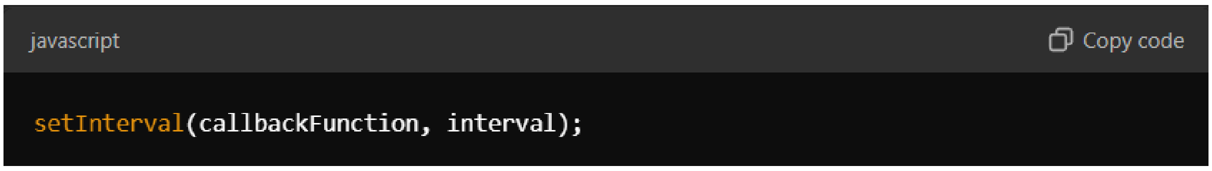


```javascript
let timeoutId = setTimeout(() => {
    console.log("This won't be displayed!");
}, 5000);

// Cancel the timeout
clearTimeout(timeoutId);
```

2. setInterval()

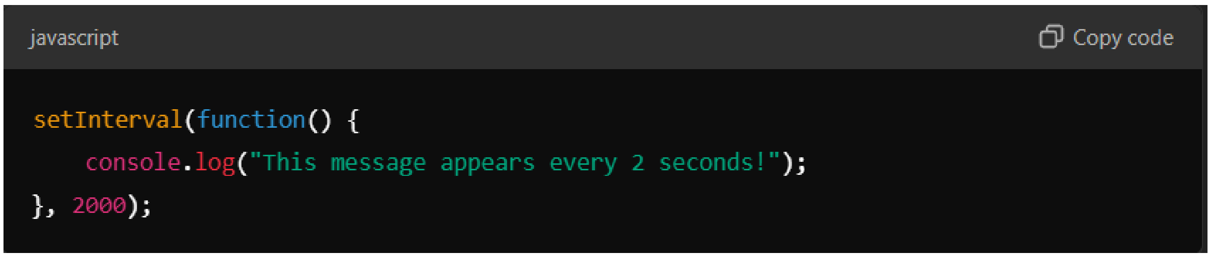
The setInterval() function allows you to execute a piece of code repeatedly at a specified time interval in milliseconds.



```javascript
setInterval(callbackFunction, interval);
```

- **callbackFunction: The function to be executed repeatedly.**
- **interval: The time in milliseconds between each execution.**

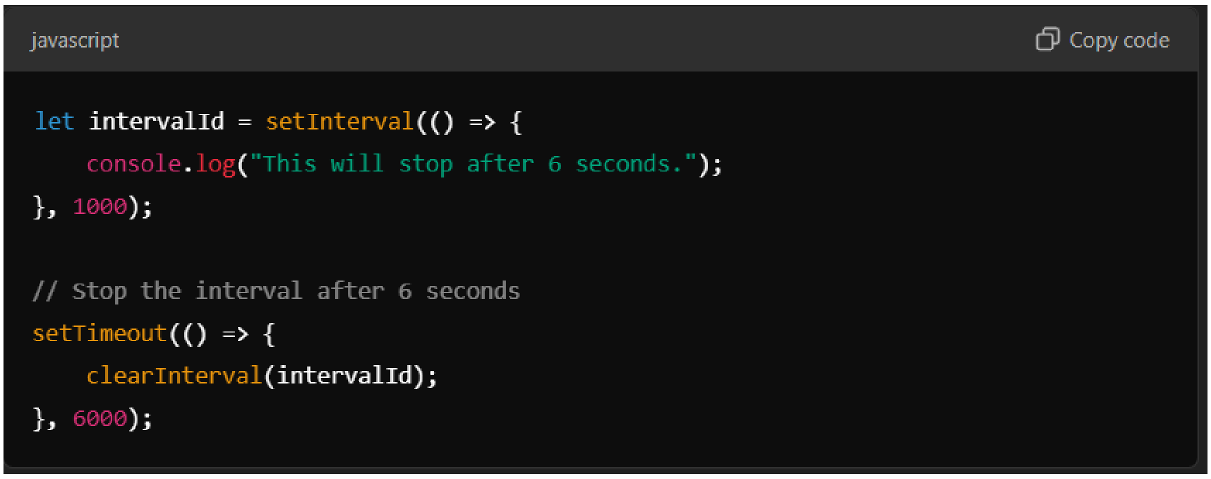
Example: Print a Message Every 2 Seconds



```javascript
setInterval(function() {
    console.log("This message appears every 2 seconds!");
}, 2000);
```

Stopping an Interval with clearInterval():

To stop an interval, use clearInterval().

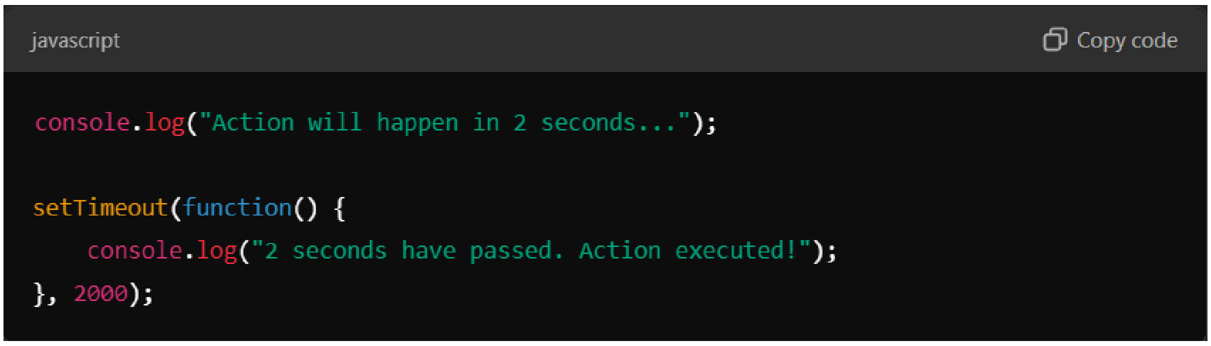


```javascript
let intervalId = setInterval(() => {
    console.log("This will stop after 6 seconds.");
}, 1000);

// Stop the interval after 6 seconds
setTimeout(() => {
    clearInterval(intervalId);
}, 6000);
```

24) : Provide an example of how to use setTimeout() to delay an action by 2 seconds.

ANS.



```javascript
console.log("Action will happen in 2 seconds...");

setTimeout(function() {
    console.log("2 seconds have passed. Action executed!");
}, 2000);
```