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

JavaScript is a programming language that runs mainly in web browsers. It's used to make websites interactive and dynamic by manipulating the page content, handling user actions, and communicating with servers without reloading the page. It's a core web technology essential for building modern web applications.

Question 2: How is JavaScript different from other programming languages like Python or Java?

- JavaScript runs mainly in browsers, designed for web interactivity,
  while Python and Java are general-purpose languages used for many kinds of software.
- JavaScript is dynamically typed and prototype-based,
  Python is dynamically typed but class-based,
  Java is statically typed and class-based.
- JavaScript is interpreted at runtime by browsers,
  Python is usually interpreted but can be compiled,
  Java is compiled to bytecode running on the JVM.
- JavaScript's syntax and environment are optimized for manipulating web pages (DOM), unlike Python and Java which have broader uses.
- JavaScript supports event-driven, asynchronous programming naturally for user interfaces, whereas Python and Java need extra tools or frameworks for that.

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

The `<script>` tag in HTML is used to add JavaScript code to a webpage, either directly inside the tag or by linking an external JavaScript file using the `src` attribute. Linking external files keeps code separate and organized.

Question 4: What are variables in JavaScript? How do you declare a variable using var, let, And const ?

Variables are containers used to **store data values**
var x = 10;
let y = 20;
const z = 30;

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

- primitive data types
  1. Number
     Ex: let age = 25;
     let price = 19.99;
  2. String
     Ex: let name = "Good Morning";

let greeting = 'Hello';

3.Boolean

Ex: let isActive = true;

let isComplete = false;

4.Undefined

Ex: let data;

console.log(data);

5.Null

Ex: let result = null;

- Non-primitive data type
  1. Object

     Ex: let person = { name: "Alice", age: 30 };

     let numbers = [1, 2, 3];
  2. Arrays

     Ex: let arr=[1,M,4,3,r];
  3. Functions

     Ex:`function functionname() {}`

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

- **`undefined`** means a variable is declared but not assigned a value yet — it's automatic.
- **`null`** means a variable is explicitly set to "no value" by the programmer.

Alright, here's a clear and direct explanation for both questions:

Question 7: **What are the different types of operators in JavaScript? Explain with examples.**

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Logical operators**

1. Arithmetic Operators

Operator	Description	Example	Result
+	Addition	`5 + 3`	8
-	Subtraction	`5 - 3`	2
*	Multiplication	`5 * 3`	15
/	Division	`6 / 3`	2
%	Modulus (remainder)	`5 % 2`	1

++	Increment	`let x = 5; x++`	6
--	Decrement	`let x = 5; x--`	4

2. Assignment Operators

Operator	Description	Example
=	Assign	`x = 5`
+=	Add and assign	`x += 3` (x = x + 3)
-=	Subtract and assign	`x -= 2`
*=	Multiply and assign	`x *= 4`
/=	Divide and assign	`x /= 2`
%=	Modulus and assign	`x %= 3`

3. Comparison Operat

Operator	Description	Example
==	Equal (loose equality)	`5 == '5'` → true
===	Strict equal (type + value)	`5 === '5'` → false
!=	Not equal (loose)	`5 != '6'` → true
!==	Strict not equal	`5 !== '5'` → true
>	Greater than	`5 > 3` → true
<	Less than	`3 < 5` → true
>=	Greater than or equal	`5 >= 5` → true
<=	Less than or equal	`3 <= 4` → true

4. Logical Operators

Operator	Description	Example	Result
&&	Logical AND	`true && false`	false
||	Logical OR	true || false	true
!	Logical NOT (negation)	`!true`	false

Question 8: Difference between == and === in JavaScript

- == checks equality after converting types if needed (type coercion). So `5 == '5'` is `true` because it converts the string `'5'` to number 5 before comparing.
- === checks equality without any type conversion — both value and type must be exactly the same. So `5 === '5'` is `false` because one is a number, the other is a string.

Qustion 9:What is control flow in JavaScript? Explain how if-else statements work with an Example.

Control flow in JavaScript controls the order in which code runs based on conditions. The `if-else` statement lets you execute code blocks conditionally: if the `if` condition is true, its block runs; otherwise, the `else` block runs. You can chain multiple conditions using `else if.`

```
let score = 75;

if (score >= 90) {
  console.log("Grade A");
} else if (score >= 70) {
  console.log("Grade B");
} else {
  console.log("Grade C or below");
}
```

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

- Use `switch` when:
  I. You're comparing **one variable/expression against many fixed values**.

  II. You have multiple discrete cases, making `switch` cleaner and easier to read than many `if-else` statements.

- Use `if-else` when:
  I. Your conditions are complex or involve ranges/expressions (e.g., `x > 10 && y < 5`).

  II. You need more flexible boolean logic.

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

1. for Loop

Used when you know how many times you want to repeat something.

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

2. while Loop

Runs as long as a specified condition is true. Good when you don't know how many times you'll loop.

```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

3. do-while Loop

Similar to `while`, but the code inside runs **at least once** before the condition is checked.

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
```

Question 12: What is the difference between a while loop and a do-while loop.

- `while` loop: Checks the condition before running the loop body. If the condition is false at the start, the loop body may never run.
- `do-while` loop: Runs the loop body once first, then checks the condition. So the loop body always runs at least once, even if the condition is false initially.

Question13: What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Functions are **blocks of reusable code** designed to perform a specific task. You define a function once and can call (execute) it multiple times throughout your program.

```
function function Name(parameters) {
   // code to execute
}
Function Name(arguments);
```

Question 14: **What is the difference between a function declaration and a function expression?**

- Function declarations are hoisted completely, so you can call them before they appear in code.
- Function expressions are assigned to variables and are not hoisted fully—you can't call them before the assignment.
- Declarations have a fixed name; expressions can be anonymous and assigned to variables.

Question 15: Discuss the concept of parameters and return values in functions.

- Parameters allow functions to receive input values (arguments).
- Return values allow functions to send a result back to the caller using `return`; if no return is given, the function returns `undefined`.

Question 16: What is an array in JavaScript? How do you declare and initialize an array?

An array is a special type of object used to store multiple values in a single variable.
let fruits = ["Apple", "Banana", "Cherry"];

Question 17. Explain the methods push(), pop(), shift(), and unshift() used in arrays

1. 1. push()
   Adds one or more elements to the end of an array.
2. pop()
   Removes the last element from an array.
3. shift()
   Removes the **first element** of an array.
4. unshift()
   Adds one or more elements **to the beginning** of an array.

Question 18. What is an object in JavaScript? How are objects different from arrays?

An **object** is a collection of **key-value pairs** where each key (also called a property) is a string or symbol, and the value can be **any data type** (number, string, array, function, etc.).

Objects store data as named properties (unordered).

Arrays store ordered lists accessed by numeric indexes.

Question 19. Explain how to access and update object properties using dot notation and

bracket notation.

1. . Dot Notation
   Easiest and most common method.
   Only works if the property name is a **valid identifier**

   ```
   let person = { name: "Alice", age: 25 };
   console.log(person.name);
   person.age = 26;
   console.log(person.age);
   ```

2. Bracket Notation
   Property name is given as a **string** inside `[ ]`.
   Works for **dynamic keys**, keys with **spaces**, or **special characters.**

   ```
   let person = { name: "Alice", age: 25, "home city": "Delhi" };
   console.log(person["home city"]);
   person["name"] = "Bob";
   console.log(person["name"]);
   let key = "age";
   console.log(person[key]);
   ```

Question 20**. What are JavaScript events? Explain the role of event listeners.**

JavaScript **events** are actions like clicks, key presses, or page loads. **Event listeners** are functions that wait for these events on elements and execute code in response, enabling interactive web pages.

Question 21: How does the addEventListener() method work in JavaScript? Provide Anexample.

`addEventListener()` attaches a function to an element for a specific event without overwriting existing handlers. When the event occurs (e.g., `"click"`), the function runs, allowing dynamic and flexible event handling.

Question 22: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

The DOM is a tree-like representation of an HTML page that JavaScript can manipulate to change content, structure, and style dynamically. JavaScript interacts with the DOM using methods to select and modify page elements.

Question 23. : Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM

- `getElementById()` selects a single element by its unique ID.
- `getElementsByClassName()` selects all elements with a given class (returns a live collection).
- `querySelector()` selects the first element matching any CSS selector, offering more flexibility.

Question 24. **Explain the setTimeout() and setInterval() functions in JavaScript. How are theyused for timing events?**

setTimeout()

- Executes a function **once after a specified delay** (in milliseconds).
```
setTimeout(() => {
  console.log("Executed after 2 seconds");
}, 2000);
```

setInterval()

- Executes a function **repeatedly at fixed intervals** (in milliseconds).
```
setInterval(() => {
  console.log("Executed every 3 seconds");
}, 3000);
```

Question 25. Provide an example of how to use setTimeout() to delay an action by 2 seconds.

```
setTimeout(() => {
  console.log("This message appears after 2 seconds");
}, 2000);
```

Question 26.: **What is error handling in JavaScript? Explain the try, catch, and finally blockswith an example.**

Error handling is the process of catching and managing runtime errors in your code to prevent the program from crashing and to handle errors gracefully.

try, catch, and finally Blocks

- **`try` block:** Contains code that may throw an error.
- **`catch` block:** Executes if an error occurs in the `try` block; receives the error object.
- **`finally` block:** Executes **after** `try` and `catch`, regardless of whether an error occurred or not. Useful for cleanup.

```
try {
  let result = riskyFunction();  // Might throw an error
  console.log("Result:", result);
} catch (error) {
  console.log("Caught an error:", error.message);
} finally {
  console.log("This always runs, error or not.");
}
```

Question 27. Why is error handling important in JavaScript applications?

Error handling is essential to keep JavaScript applications from crashing unexpectedly. It ensures the app can handle problems gracefully, maintain stability, and provide meaningful feedback to users, improving reliability and user experience.