Javascript Introduction:

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

Answer:1 JavaScript is a high-level, dynamic, untyped, and interpreted programming language that is widely used for web development.

Role of Javascript in Web Development :

* Javascript is a client-side-scripting language : JavaScript is primarily a client-side language, meaning it runs in the user's web browser rather than on the web server.
* JavaScript adds allows websites by enabling actions such as button clicks, mouse movements, scrolling events, and keyboard inputs.
* DOM Manipulation.
* JavaScript is essential for creating complex user interfaces (UIs). With libraries and frameworks like React, Vue.js, and Angular.
* Cross-Platform Development: JavaScript can be used not only for web development but also for mobile app development (using frameworks like React Native) .

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

Primary use: javascript can be used for front end and backend (like node js)web development

Python can be used for web development,data science,automation

Java can be used for enterprise apps,android,large systems

Typing:

Javascript is dynamically typed

Python is dynamically typed

Java is statically typed

Syntax:

JAvascript is flexible,event driven uses {} and ;

Python is clean indentation based

Java is verbose,object-oriented uses {} and ;

Execution:

Client side with react js and server side with node js

Python interpreted,typically on server

Java is compiled to bytecode,run on JVM

Performance:

Javascript is fast for web tasks,limited by browser

Python is slower but highly optimised for I/O

Java is high performance for large apps.

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

Inline: The <script> tag can contain JavaScript code directly within it. This is called inline scripting, where the JavaScript code is written inside the <script> element.

Linking to an External JavaScript File: You can also link to an external JavaScript file using the src attribute of the <script> tag. This is the preferred method for organizing and managing JavaScript code, especially when the script is large or needs to be reused across multiple HTML pages.

How to Link an External JavaScript File to an HTML Document

To Place the External Script File: First, ensure that your JavaScript file is saved with the .js extension (e.g., script.js) in the desired directory.

To Use the <script> Tag: In your HTML document, include the <script> tag and specify the path to the external JavaScript file using the src attribute.

If the JavaScript file is in the same directory as your HTML file:

<script src="script.js"></script>

If the JavaScript file is hosted on a different domain or a CDN:

<script src="https://example.com/path/to/script.js"></script>

2. Variables and Data Types

**Question 1: What are variables in JavaScript? How do you declare a variable using var, let, and const?**

In Javascript variables are used to store data values.They act as a container to data. Javascript supports three ways to declare variables.var,let and const.

1. Using var :

var is the traditional way to declare variables in JavaScript. can be re-declared and reassigned.

Var n=’Pinjal’;

Console.log(n);

Var n=’Kinjal’;

Console.log(n);

Output will be :

Pinjal

Kinjal

1. Using let :

Here redeclare is not allowed but can be re assigned

Ex:

Let n=25;

Console.log(n);

N=30;

Console.log(n);

Output will be 25

30

// but writing like let n =30 will cause an error

1. Using const:

Here redeclaration and re assigned both are not allowed

Const pi=.3.14;

Console.log(pi)

Output 3.14

But writing like const pi=3.50

Or pi=3.60

Will cause errors

**Question 2: Explain the different data types in JavaScript. Provide examples for each.**

1.String: represents a sequence of characters

Let greeting=’Hello World”;

Let name=’Pinjal’;

Let message= ‘Welcome, ${name}!’;

2.Number:represents both integer and floating point. Javascript does not differentiate between different types of numbers.

Let age=30;

Let price=14.45;

3.Boolean:represents logical entities can have two values true and false

Let isActive=true;

Let isLoggedIn=false;

4.Undefined

A variable that has been declared but has not been assigned a value is of type undefined.

Let x;

Console.log(x);

5.Null:Has no value.

Let user=null;

**Question 3: What is the difference between undefined and null in JavaScript?**

In JavaScript, undefined and null are both primitive values that represent the absence of a value, but they are used in different contexts and have distinct meanings. Here are the key differences between the two:

* undefined is a type and a value. It indicates that a variable has been declared but has not yet been assigned a value.

‘Null’is an intentional assignment value that represents "no value" or "no object." It is used to indicate that a variable should be empty or that an object is intentionally set to have no value.

3. JavaScript Operators  
**Question 1: What are the different types of operators in JavaScript? Explain with examples. o Arithmetic operators o Assignment operators o Comparison operators o Logical operators**

Answer: In JavaScript, operators are special symbols used to perform operations on variables and values. JavaScript includes several types of operators, each serving a specific purpose.

1)Arithmetic Operators  
2)Comparison Operators  
3)Assignment Operators  
4)Logical Operators  
5)String  
6)Type of  
7)Ternary Operators  
  
1) Arithmetic Operators: Inludes addition(+),subtraction(-),Multiplication(\*),Division(/),Exponentiation(\*\*),Modulous(%)  
2)Comparison Operators:Equal to (==),Strictt Equal to (===),Not Equal to(!=),Strict Not Equal to(!==),greater than(>),greater than equal to(>=),Less than(<),Less than Equal to(<=)

3)Assignment Operators:Simple Assignment(=),Addition Assignment(+=),Subtration Assignment(-=),Multiplication Assignment(\*=),Division Assignment(/=),Modulous Assignment(%=)  
4)Logical Operator:AND(&&),OR(||),Not(!)

Question:2What is the difference between == and === in JavaScript?

== is used to compare values whereas === is used to compare (check) values and data types.

4. Control Flow (If-Else, Switch)

**Question:1 What is control flow in JavaScript? Explain how if-else statements work with an example.**

Answer:1 Control flow in JavaScript is the order in which statements are executed by the JavaScript interpreter. The default control flow is to read and execute statements in order from top to bottom and left to right in a program file.

How if-else statements work: The if...else statement checks the condition and executes code in two ways: If condition is true, the code inside if is executed. And, the code inside else is skipped. If condition is false, the code inside if is skipped.

**Question:2 : Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?**A switch statement in JavaScript is used to execute one of many blocks of code based on a specific condition. It is typically used when you have multiple possible conditions (like multiple possible values for a variable) and need to evaluate which one matches.

switch (expression) {

case value1:

// code to run if expression === value1

break;

case value2:

// code to run if expression === value2

break;

// You can have more cases here

default:

// code to run if no case matches

}

Why you should use switch instead of if-else condition:

1. Multiple Conditions on a Single Variable:

When you have a single variable or expression that is being compared to many different possible values, the switch statement provides a more readable and efficient solution than multiple if-else statements.

1. When you have constant value

switch is particularly useful when checking against a set of constant values (like numbers or strings), rather than conditions that require complex comparisons.

1. Readability: If you have many conditions to check that all compare against the same variable or expression, a switch statement makes the code easier to read, as all conditions are neatly grouped together
2. Efficiency: A switch statement can be more efficient than multiple if-else checks in terms of performance, particularly when there are many conditions. JavaScript engines can optimize switch statements more effectively than a series of if-else chains.

5 Loops

**Question 1: Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.**

In JavaScript, loops are used to repeat a block of code multiple times. There are several types of loops, with the most common ones being for, while, and do-while. Each type of loop is useful in different scenarios.

1. for Loop

The for loop is generally used when the number of iterations is known beforehand. It consists of three parts: initialization, condition, and increment/decrement. The loop runs as long as the condition evaluates to true.

for (let i = 0; i < 5; i++) {

console.log(i);

}

Output: 0

1

2

3

4

2. while Loop

The while loop runs as long as the given condition evaluates to true. The condition is checked before each iteration, so if the condition is false initially, the code inside the loop will not execute at all.

let i = 0;

while (i < 5) {

console.log(i);

i++;

}

Output:0

1

2

3

4

3. do-while Loop

The do-while loop is similar to the while loop, but with one key difference: the code inside the do block is always executed at least once, because the condition is checked after the loop body executes.

let i = 0;

do

{

console.log(i);

i++;

}

while (i < 5);

Output:0

1

2

3

4

**Question 2: What is the difference between a while loop and a do-while loop?**

In while loop condition check happens before entering the loop. In do while loop condition checks after entering body…

While loop executes and generates output after condition check where as do while loop executes and generates output first at least once in loop later checks condition.

While loop is entry control loop.. Do while loop is exit control loop.

6.Functions

**Question:1 What are functions in JavaScript? Explain the syntax for declaring and calling a function**

Function is a set of statements that takes input and produces output.   
Declaring a Function Syntax:

Function add (x,y){

Return z

}

Above is the syntax for declaring a function  
Calling a function:

Function add(x,y){

Let z = x+y

Return z

}

Document.write(‘<br>’,add(5,10))

Above line is the syntax for calling a function

**Question2: What is the difference between a function declaration and a function expression?**

Function declaration is about declaring a function… Syntax of function declaration is :

Function add (x,y){

Return z

}

Where as creating a function and assigning it to a variable is called function expression

Const add = function (x,y){

}

Above is the syntax of function expression

**Question3: : Discuss the concept of parameters and return values in functions.**

Parameters are also called as arguments. Generally it is written just after function keyword inside parenthesis.

Return values are the values that a function produces and sends back to the place where it was called.

The return value is used to send back the result of the function's operations.

7.Array  
**Question 1: What is an array in JavaScript? How do you declare and initialize an array?**An array is a collection of data with similar datatype. Each element in an array is accessed using an index, starting from 0 for the first element.

Let arr=[10,20,30,40,50]  
Above method declares and initializes an array.

**Question 2: Explain the methods push(), pop(), shift(), and unshift() used in arrays.**

Push(): Add an element at the last of an array

Let arr=[10,20]

Arr.push(30)

Document.write(‘<br>’,arr)

Arr=[10,20,30]  
Pop():Removes the last element of an array

Let arr=[10,20,30]

Arr.pop()

Document.write(‘<br>’,arr)

Arr=[10,20]  
Unshift(): Adds new element to the first position of an array

Let arr=[10,20]

Arr.unshift(30)

Document.write(‘<br>’,arr)

Arr=[30,10,20]  
shift():Removes first element from the array.

Let arr=[30,10,20]

Arr.shift()

Document.write(‘<br>’,arr)

Arr=[10,20]

8.Objects

**Question 1: What is an object in JavaScript? How are objects different from arrays?**

It is a structure that stores data value pairs.

Arrays are different from object in many ways

Array stores data of similar data type whereas object stores data in key value pair.

Arrays use numerical indices (e.g., arr[0], arr[1], etc.).

Objects use string keys (e.g., obj["name"], obj.age).

**Question 2:Explain how to access and update object properties using dot notation and bracket notation.**

1)Using Dot Notation

const person = {

name: "Pinjal",

age: 35,

job: "Python Developer"

};

console.log(person.name);

console.log(person.age);  
2)Using bracket Notation

const person = {

name: "Pinjal",

age: 35,

job: "Python Developer"

};

console.log(person["name"]);

console.log(person["age"]);

9.Javascript Events

**Question 1: What are JavaScript events? Explain the role of event listeners**

JAvascript events allows users to functions of events to create. Events like mouse clicks, keyboard clicks and window resizing.

Role of event listeners:

Event listeners enable interaction by attaching behavior (functions) to events, so that the browser can respond to user actions or other events. It is useful for handling dynamic content.

**Question 2: How does the addEventListener() method work in JavaScript? Provide anexample.**

<button id="add">Click me!</button>

Const button = document.getElementByID(‘add’)

Function manageClick(){

Alert(‘Button clicked’)

}

Button.addEventListener(‘click’,manageClick);

10.Dom Manipulation:

**Question 1:What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?**

The Document Object Models allows javascript to add,update and delete the elements,attributes and content of the webpage without requiring the page reload.

JavaScript can access elements in the DOM using various methods. Some common ones are:

document.getElementById(id) , document.getElementsByClassName(className) , document.getElementsByTagName(tagName), document.querySelector(selector), document.querySelectorAll(selector)

**Question 2: : Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.**

document.getElementById(id) - Selects an element by its id attribute.

document.getElementsByClassName(className) - Selects all elements with a specific class name.

document.querySelector(selector) - Selects the first element that matches the CSS selector.

11. JavaScript Timing Events (setTimeout, setInterval)

**Question 1: : Explain the setTimeout() and setInterval() functions in JavaScript. How are theyused for timing events?**

setTimeout(): The setTimeout() function is used to execute a function or a block of code once, after a specified amount of time

setTimeout(function, delay);

setInterval() : The setInterval() function is used to execute a function or a block of code repeatedly at fixed intervals, after a specified delay.

setInterval(function, interval);

**Question 2: Provide an example of how to use setTimeout() to delay an action by 2 second**s

setTimeout(function()

{

Document.write("This message appears after 2 seconds.");

}, 2000); // 2000 milliseconds = 2 seconds

12.Javascript Error Handling

**Question 1: What is error handling in JavaScript? Explain the try, catch, and finally blockswith an example.**

Error handling in javascript is about responding to potential errors that may occur during the execution of the program.

Try: The try block is used to wrap the code that may potentially throw an error.

Catch: The catch block handles any errors that occur in the try block.

Finally: The finally block contains code that will execute regardless of whether an error occurred or not.

function divide(a, b)

{

try {

if (b === 0) {

throw new Error("Division by zero is not allowed.");

}

const result = a / b; console.log(`Result: ${result}`);

}

catch (error) {

console.log("Error:", error.message);

}

finally {

console.log("Execution of the divide function is complete.");

} }

divide(10, 2);

divide(10, 0);

Question 2:Why is error handling important in JavaScript applications?

It prevents crashes, improves user experience, helps with debugging, Ensures stability,

Enhances security, and provides better control.