1. JavaScript Introduction

Theory Assignment

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

Javascript is a scripting language.it is use for syncronus and unsyncronus maners. JavaScript enables users to interact with web elements, such as buttons, forms, and images, and it handles real-time updates, making web pages more engaging and user-friendly.

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

JavaScript	Python	Java
Scripting language	High-level scripting language	Object-oriented programming language
Web development	General-purpose	Enterprise applications
Interpreted	Interpreted	Compiled to bytecode

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

We use the src attribute within the <script> tag in the HTML file to specify the source file (JavaScript) location. The external JavaScript file is then linked to the HTML document, enabling the execution of its code.

2. Variables and Data Types Theory Assignment

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

Variables in JavaScript are containers used for storing data values. They act as named storage locations that can hold information that can be reused or updated later in a program.

Here's how to declare variables using var, let, and const:

```
var x = 10;
console.log(x);
x = 20;
console.log(x);

let y = 30;
console.log(y);
y = 40;
console.log(y);

const country = "India";
console.log(country);
```

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

Primitive Data Types:

- **String:** Represents textual data. It is enclosed in single quotes (' '), double quotes (" "), or backticks (\` \`).

```
let name = "John Doe";

let message = 'Hello World';

let template = `My name is ${name}`;
```

- **Number:** Represents numeric values, including integers and floating-point numbers. JavaScript does not differentiate between integers and floating-point numbers. It also includes special values like NaN (Not a Number), Infinity, and -Infinity.

```
let age = 30;

let price = 99.99;
```

```
let notANumber = NaN;

let infinity = Infinity;
```

- **BigInt:** Represents integers with arbitrary precision, allowing you to store and operate on larger numbers beyond the integer limit.

```
let largeNumber = 123456789012345678901234567890n;
```

- **Boolean:** Represents logical values, either true or false.

```
let isAdult = true;

let isStudent = false;
```

- **Null:** Represents the intentional absence of any object value.

```
let emptyValue = null;
```

- **Undefined:** Represents a variable that has not been assigned a value.

```
let notAssigned;
```

- **Symbol:** Represents unique and immutable values, often used as keys in objects.

```
let uniqueId = Symbol('id');
```

Non-Primitive Data Type:

- **Object:** Represents complex data structures that can store collections of key-value pairs. Objects can contain other objects, arrays, functions, and primitive values. Arrays and functions are also considered objects in JavaScript.

```
let person = {

    name: "John Doe",

    age: 30,

    isAdult: true,

    address: {
```

```
        city: "New York",
        country: "USA"
    }
};

let numbers = [1, 2, 3, 4, 5];

function myFunction() {
    console.log("Hello from function");
}
```

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

undefined	JavaScript
Indicates a variable that has been declared but not yet assigned a value.	Represents the intentional absence of any object value.
Primitive data type in JavaScript.	Primitive data type in JavaScript.
Automatically assigned to variables that are declared but not initialized.	Must be explicitly assigned to a variable.

3. JavaScript Operators Theory Assignment

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

Arithmetic operators Assignment operators

Comparison operators Logical operators

JavaScript operators are symbols that perform operations on values and variables.

- **Arithmetic Operators**: These operators perform mathematical calculations..

```
let x = 10
let y = 5
console.log(x + y)
console.log(x - y)
console.log(x * y)
console.log(x / y)
console.log(x % y)
console.log(x++)
console.log(++y)
console.log(x--)
console.log(--y)
```

- **Assignment Operators**: These operators assign values to variables.

```
let a = 5
a += 2
a -= 2
a *= 2
a /= 2
a %= 2
```

- **Comparison Operators**: These operators compare two values and return a boolean result (true or false).

```
let p = 10
```

```
let q = 5

console.log(p == q)

console.log(p != q)

consoleiner.log(p > q)

console.log(p < q)

console.log(p >= q)

console.log(p <= q)
```

- **Logical Operators**: These operators perform logical operations on boolean values.

```
let condition1 = true

let condition2 = false

console.log(condition1 && condition2)

console.log(condition1 || condition2)

console.log(!condition1)
```

2. **What is the difference between == and === in JavaScript?**
   == is use for compare two values and === is use for compare Two data types and values.

4. Control Flow (If-Else, Switch)

1: What is control flow in JavaScript? Explain how if-else statements work withan example.

- Definition:

Control flow in JavaScript means the order in which code is executed in a program

It helps us write logic:
“If this happens, do that... otherwise do something else.”

--- If-else Statement ---

Definition:
if-else is a conditional statement that runs different blocks of code depending on whether a condition is true or false.

Syntax:

```
if (condition) {
  // runs if condition is true
} else {
  // runs if condition is false
}
```

Example:

```
let marks = 75;

if (marks >= 90) {
  console.log("Grade: A");
} else if (marks >= 75) {
  console.log("Grade: B");
} else if (marks >= 60) {
  console.log("Grade: C");
} else {
  console.log("Fail");
}
```

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

A switch statement is used to compare a single expression against multiple possible values (called cases).
It’s a cleaner alternative to writing many if-else if blocks.

Syntax:

```
switch (expression) {
  case value1:
    // runs if expression === value1
    break;
  case value2:
    // runs if expression === value2
    break;
  default:
    // runs if no case matches
}
```

Example:

```
let fruit = "apple";

switch (fruit) {
  case "banana":
    console.log("Yellow fruit");
    break;
  case "apple":
    console.log("Red fruit");
    break;
  case "orange":
    console.log("Orange fruit");
    break;
  default:
    console.log("Unknown fruit");
}
```

5. Loops (For, While, Do-While)

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

- A function in JavaScript is a block of reusable code designed to perform a specific task

- Syntax :-

```
function functionName(parameters)
{

}
functionName(arguments);
function greet(name)
{
  console.log("Hello, " + name + "!");
}
greet("Amit");
greet("Priya");
```

Output:
Hello, Amit!
Hello, Priya!

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

A function defined with the function keyword directly and independently.

```
function greet()
```

```
{
  console.log("Hello!");
}
```
You can call it even before it's defined (hoisting).

A function assigned to a variable. It can be named or anonymous.

Syntax:

```
const greet = function()
 {
  console.log("Hello!");
};
```
You cannot call it before it's defined.

3 : Discuss the concept of parameters and return values in functions

They are used when declaring a function.

Example:

```
function greet(name)
{
  console.log("Hello, " + name);
}
greet("Amit");
```
A function can return a value back to the place where it was called using the return keyword.
This value can be stored in a variable or used directly.

Example:

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

let sum = add(5, 3);
console.log(sum);
```

7.Arrays

8. Objects

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

- An object in JavaScript is a collection of key-value pairs.

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

"name", "age", "isStudent" are keys (or properties)
"Amit", 25, true are values

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

- Objects:

Objects store data in key-value pairs.
Each value has a name (key), like:

```
let person = {
  name: "Amit",
```

```
  age: 25,
  isStudent: true
};
```

- Arrays:

Arrays store data in a list, using numbered positions (indexes):

```
let fruits = ["apple", "banana", "mango"];
```

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

```
let person = {
  name: "Amit",
  age: 25
};

console.log(person.name); // Output: Amit
console.log(person.age);  // Output: 25
person.age = 30;
console.log(person.age);  // Output: 30
console.log(person["name"]); // Output: Amit
person["name"] = "Rahul";
console.log(person.name);    // Output: Rahul
```

9. JavaScript Events

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

In JavaScript, events are actions or occurrences that happen in the browser — often triggered by the user

- Clicking a button (click)
- Typing in an input box (keydown)
- Moving the mouse (mousemove)
- Page loaded (load)

An event listener is a function in JavaScript that waits for a specific event to happen, and then runs code in response.

```
<button id="myBtn">Click Me</button>

<script>
  // select the button
  let btn = document.getElementById("myBtn");

  // add an event listener to the button
  btn.addEventListener("click", function() {
    alert("Button was clicked!");
  });
</script>
```

10. DOM Manipulation

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

The DOM (Document Object Model) is a programming interface that represents the structure of a web page as a tree of objects.

- Select HTML elements by ID, class, tag name, or other selectors.
- Change the text or HTML inside an element.
- Modify the style of elements (like changing colors, size, visibility, etc.).
- Add new elements or remove existing ones from the page.
- Listen for events like clicks, form submissions, or key presses and respond to them with specific actions.

2: Explain the methods getElementById(), getElementsByClassName(),andquerySelector() used to select elements from the DOM.

This method is used to select a single element from the DOM using its unique ID.

It returns the first element that has the specified ID.

let title = document.getElementById("mainTitle");

This will select the element with the ID mainTitle.

This method is used to select multiple elements that share the same class name.

It returns a collection (HTMLCollection) of elements, not a single one.

let items = document.getElementsByClassName("listItem");

This will select all elements with the class listItem.

This method is used to select the first element that matches a CSS selector (like class, ID, or tag).

It's more flexible because it allows you to use complex selectors.

let firstButton = document.querySelector(".btn");

This will select the first element with the class btn.

11. JavaScript Timing Events (setTimeout, setInterval)

1: Explain the setTimeout() and setInterval() functions in JavaScript. Howare they used for timing events?

setTimeout() is used to run a function once after a specific delay (in milliseconds).

- Syntax:

setTimeout(function, delay);

-s Example:

```
setTimeout(() => {
  console.log("This runs after 2 seconds");
}, 2000);
```
- This will print the message after 2 seconds, only once.

setInterval() is used to run a function repeatedly at a fixed time interval (in milliseconds).

- Syntax:

```
setInterval(function, interval);
```

- Example:

```
setInterval(() => {
  console.log("This runs every 2 seconds");
}, 2000);
```
- This will print the message every 2 seconds, again and again.
- Use setTimeout() when you want to delay something once (e.g., showing a popup after 5 seconds).
- Use setInterval() when you want to repeat something (e.g., a clock, slider, or auto-refreshing data).

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

---- setTimeout()

```
console.log("Wait for it...");

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

12. JavaScript Error Handling

1: What is error handling in JavaScript? Explain the try, catch, and finally

Error handling in JavaScript means managing and responding to errors in your code

JavaScript uses three main blocks for error handling:

1. try block:
Code that may throw an error is placed inside try.

2. catch block:
If an error occurs in the try, it's caught and handled here.

3. finally block (optional):
This block runs no matter what — whether an error occurred or not. It's used for cleanup code.

```
try {
  let a = 10;
  let b = a + c;  // 'c' is not defined — this will cause an error
  console.log("This line won't run.");
} catch (error) {
  console.log("An error occurred: " + error.message);
} finally {
  console.log("This runs no matter what.");
}
```

2: Why is error handling important in JavaScript applications?

1. Prevents Application Crashes

Without error handling, even a small bug or unexpected input can crash the entire app. Proper handling keeps the app running smoothly.

2. Provides User-Friendly Messages

Instead of showing confusing error codes or crashing the page, you can show clear messages like “Something went wrong, please try again.”

3. Improves Debugging

Using catch blocks, you can log or display error details which helps developers understand what went wrong and where.

4. Ensures Safe Execution

You can use finally blocks to clean up resources whether or not an error occurred.

5. Essential for Async Code

JavaScript heavily uses async operations . Errors in these need to be handled properly to avoid silent failures.