Theory Assigment - Javascript

Q. Write a brief essay on the history of JavaScript, its evolution, and its role in modern web development.

Ans : JavaScript was developed by Brendan Eich in 1995 while working at Netscape. It was originally created in just 10 days to make web pages more interactive. Initially named "Mocha," it was later renamed to "LiveScript," and finally "JavaScript" to align with the popularity of Java at the time.

Over the years, JavaScript has gone through major upgrades. It was standardized by ECMA International in 1997, known as ECMAScript. The most important update was ES6 (released in 2015), which introduced modern features like `let`, `const`, arrow functions, classes, modules, and promises.

Today, JavaScript is one of the most widely used programming languages in the world. It is essential for frontend development and, with the help of Node.js, is also used for backend development. Frameworks like React, Angular, and Vue have made JavaScript even more powerful in building dynamic web applications. JavaScript is also used in mobile (React Native) and desktop (Electron) app development.

Q. Explain the basic syntax of JavaScript, including variables, data types, and operators. Discuss the differences between let, const, and var.

Ans :

JavaScript syntax defines the rules to write code. Variables can be declared using `var`, `let`, or `const`.

Data Types:
JavaScript has primitive types like Number, String, Boolean, Null, Undefined, and complex types like Object and Array.

Operators:
Arithmetic operators: `+`, `-`, `*`, `/`, `%`
Assignment operators: `=`, `+=`, `-=`, `*=`
Comparison operators: `==`, `===`, `!=`, `<`, `>`
Logical operators: `&&`, `||`, `!`

Difference between var, let, and const:

- `var` has function scope, can be re-declared and re-assigned.
- `let` has block scope, cannot be re-declared but can be re-assigned.
- `const` has block scope, cannot be re-declared or re-assigned.

• Describe how control flow statements work in JavaScript, focusing on if, else if, else, and switch statements.

Ans :
- `if`: Condition true ho to code execute hota hai.
- `else if`: Multiple conditions check karne ke liye use hota hai.
- `else`: Sab condition false ho to ye chalega.
- `switch`: Single value ke against multiple cases check karne ke liye.

Q. Explain the different types of loops in JavaScript (for, while, and do...while). Discuss their use cases.

Ans : JavaScript provides different loops to repeat a block of code.

for loop: Used when the number of iterations is known.

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

while loop: Used when the condition is checked before executing the code.

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

do...while loop: Runs the block at least once before checking the condition.

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

Use cases:

- Use `for` loop for counting or fixed repetitions.

- Use `while` loop when the end condition is not fixed.
- Use `do...while` loop when the loop should run at least once.

Q. Define what functions are in JavaScript. Discuss the concepts of function scope, block scope, and hoisting.

Ans : A **function** in JavaScript is a reusable block of code that performs a specific task. It is defined once and can be called multiple times.

```
function greet() {
  console.log("Hello Abhay");
}
greet();
```

Function Scope:
Variables declared inside a function are only accessible within that function.

Block Scope:
Variables declared with `let` or `const` inside `{ }` are only accessible within that block.

Hoisting:
In JavaScript, function declarations and `var` variables are moved to the top of their scope during execution.
Example:

```
greet(); // Works due to hoisting
function greet() {
  console.log("Hi");
}
```

Q. Compare and contrast arrays and objects in JavaScript. Explain how to manipulate them using built-in methods.

Ans : Array: A list of ordered values. Values can be of any type and are accessed by index.

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

Object: A collection of key-value pairs. Used to store data with labels.

```
let person = { name: "Abhay", age: 21 };
```

Array methods: `push()`, `pop()`, `shift()`, `unshift()`, `map()`, `filter()`, `forEach()`
Object methods: `Object.keys()`, `Object.values()`, `Object.entries()`

Use case:

- Use arrays for ordered lists like numbers, names.
- Use objects for structured data like user info, settings.

Q. Explain the concept of higher-order functions and how callbacks work in JavaScript.

Ans : A **higher-order function** is a function that either takes another function as an argument or returns a function.

Example:

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

A **callback** is a function passed as an argument to another function and called later when needed.

Use case:
Callbacks are used in asynchronous tasks like file handling, timers, or API calls.

Q. Describe what promises are and how they are used to handle asynchronous operations in JavaScript.

Ans : A **Promise** is a JavaScript object used to handle asynchronous operations. It represents a value that may be available now, in the future, or never.

States of Promise:

1. **Pending** – initial state
2. **Resolved** – operation completed successfully
3. **Rejected** – operation failed

Syntax:

```
let promise = new Promise(function(resolve, reject) {
  // async code
});
```

Usage :

```
promise
  .then(result => console.log(result))   // success
  .catch(error => console.log(error));   // error
```

Use case: Promises help write cleaner async code and avoid callback hell.

Q. Explain the Document Object Model (DOM) and how JavaScript interacts with it. Discuss event handling and the event loop.

Ans : The **DOM (Document Object Model)** is a tree-like structure of HTML elements created by the browser. JavaScript uses it to access and change web page content dynamically.

JavaScript with DOM:

- Use `document.getElementById()` or `querySelector()` to select elements.
- Use `.innerText`, `.style`, or `.classList` to change content, style, or classes.

Example:

```
document.getElementById("demo").innerText = "Hello Abhay";
```

Event Handling:
JavaScript can respond to user actions like clicks, typing, etc. using `addEventListener()`.

```
button.addEventListener("click", function() {
  alert("Clicked!");
});
```

Event Loop:
It manages async tasks like setTimeout or Promises. It checks the task queue and executes callbacks after the main code finishes.

Q. Discuss the key features introduced in ES6 (ECMAScript 2015), including arrow functions, template literals, and destructuring.

Ans : ES6 introduced many new features to make JavaScript easier and cleaner.

1. Arrow Functions:
Short syntax for writing functions.

```
const sum = (a, b) => a + b;
```

2. Template Literals:
Allows multiline strings and embedding variables using backticks `.

```
let name = "Abhay";
console.log(`Hello ${name}`);
```

3. Destructuring:
Extract values from arrays or objects easily.

```
let [a, b] = [1, 2];
let {name, age} = {name: "Abhay", age: 21};
```

Q. Explain the importance of error handling in JavaScript. Discuss the try, catch, and finally statements.

Ans : Error handling helps find and manage bugs in code without crashing the program. It makes the app more reliable and user-friendly.

try...catch...finally is used to handle errors.

- `try`: Code block to test for errors
- `catch`: Runs if an error occurs
- `finally`: Always runs (error aaye ya na aaye)

Example:

```
try {
  let a = 10 / 0;
} catch (err) {
  console.log("Error:", err);
} finally {
  console.log("Always runs");
}
```

Q. Describe the module system in JavaScript, including CommonJS and ES Modules. Discuss their significance for code organization.

Ans : Modules in JavaScript help break code into reusable and organized pieces.

---

1. CommonJS (used in Node.js):

- Uses `require()` to import and `module.exports` to export.
- Synchronous loading.

```
// math.js
module.exports = function add(a, b) { return a + b; }

// app.js
const add = require('./math');
```

---

2. ES Modules (ESM - used in browser & modern JS):

- Uses `import` and `export`.

- Supports asynchronous loading.

```
// math.js
export function add(a, b) { return a + b; }

// app.js
import { add } from './math.js';
```

Significance:

- Keeps code clean and modular
- Easy to maintain and reuse
- Helps with project scaling and team collaboration

Q. Explain the principles of functional programming as they apply to JavaScript. Discuss concepts like immutability and pure functions.

Ans : 1. Pure Functions:

- Always return the same output for the same input
- Do not change external values or variables

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

---

2. Immutability:

- Data is not changed directly
- Instead, a new copy is created and modified

```
let arr = [1, 2, 3];
let newArr = [...arr, 4]; // arr remains unchanged
```

Other FP concepts:

- Functions as values
- Higher-order functions
- Avoiding side effects