TOPS Technology

MODULE – 17 JavaScript for Full Stack Assignment

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

Answer:

JavaScript is a high-level, interpreted, and object-oriented programming language used to make web pages interactive, dynamic, and functional.
It is one of the core technologies of the web, along with HTML and CSS.

- **HTML** → Defines the structure/content of a webpage
- **CSS** → Defines the style and layout
- **JavaScript** → Defines the behavior and interactivity

JavaScript is supported by all modern browsers and can be executed on both client-side and server-side.

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

Answer:

While JavaScript, Python, and Java are all high-level programming languages,
they are designed for different purposes and run in different environments.
JavaScript mainly focuses on web interactivity, while Python and Java are used for general-purpose programming.

- JavaScript runs directly in the browser.
- Python and Java require separate interpreters or compilers.

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

Answer:

The <script> tag in HTML is used to embed or reference JavaScript code within a web page.
It allows you to make your page interactive, dynamic, and responsive by running scripts in the browser.

Syntax.

```
 <script>
```

document.write("<br> This is JavaScript…")

</script>

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

Answer:

In JavaScript, a variable is a named container that stores data values.
It allows you to store, update, and reuse information throughout your program.

- Think of a variable like a box with a label — you can put data inside and refer to it later by its name.

var x = 10;

let y = 20;

const z = 30;

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

Answer:

In JavaScript, a data type defines the kind of value a variable can hold and how that value can be used.
JavaScript is a dynamically typed language, meaning you don't need to declare the data type explicitly —
the type is automatically determined at runtime.

JavaScript has two main categories of data types:

1. Primitive Data Types (Immutable, simple values)
2. Non-Primitive Data Types (Complex structures like objects)

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

Answer:

In JavaScript, both undefined and null represent the absence of a value, but they are used in different situations and have different meanings.

1. undefined

- A variable is automatically assigned `undefined` when it is declared but not initialized.
- It means "no value has been assigned yet."

2. null

- null is a special value that represents "no value" or "empty."
- It is explicitly assigned by the programmer to indicate no object or no data.

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

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

Answer:

Operators in JavaScript are symbols that perform operations on operands (variables or values). They are used to manipulate data, compare values, and make logical decisions.

- Arithmetic operators are used to perform mathematical calculations.
- Assignment operators are used to assign values to variables.
- Comparison operators are used to compare two values and return a Boolean (true or false).
- Logical operators are used to combine multiple conditions or make decisions.

Question 8: What is the difference between == and === in JavaScript?

Answer:

Both == and === are comparison operators in JavaScript,
but they differ in how they compare values — particularly regarding data types.

- The == operator compares only the values of two variables.
- The === operator compares both value and data type.

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

Answer:

Control flow in JavaScript refers to the order in which statements are executed in a program.

By default, JavaScript code runs from top to bottom**,** line by line**.**
However, we can change this flow using conditional statements, loops, and functions —
to make decisions and control how the program behaves.

The most common ones are:

- if
- if...else
- if...else if...else

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

Answer:

A switch statement in JavaScript is used to perform different actions based on different conditions.
It is an alternative to multiple if**...**else if statements when you need to compare one value against many possible cases.

Syntax of switch:

```
switch(expression) {
  case value1:
    // Code block
    break;

  case value2:
    // Code block
    break;

  default:
    // Code block if no case matches
}
```

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

Answer:

A loop in JavaScript is used to execute a block of code repeatedly until a specified condition is false.
They help in avoiding repetitive code and make programs efficient and concise.

- for → Best for fixed repetitions.

- while → Best when looping depends on a condition.
- do...while → Ensures code runs at least once, even if the condition is false.

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

Answer:

Both while and do...while loops are used to execute a block of code repeatedly as long as a condition is true.

However, the main difference lies in when the condition is checked — before or after executing the loop body.

Question 13: What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Answer:

A function in JavaScript is a block of reusable code designed to perform a specific task. Instead of writing the same code multiple times, you can define it once inside a function and call it whenever needed.

Functions help make code **modular**, **readable**, and **maintainable**.

```
function functionName(parameter1, parameter2, ...) {

  // code to be executed

}
```

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

Answer:

In JavaScript, there are two main ways to define a function:

1. Function Declaration → Defines a function with a name using the function keyword.
2. Function Expression → Stores a function inside a variable (can be named or anonymous).

Both are used to create functions, but they behave **differently**, especially in terms of **hoisting** and **when** they can be used.

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

Answer:

In JavaScript, **functions** are used to perform specific tasks.
They can **accept inputs** (called **parameters**) and **send back outputs** (called **return values**).

1. Parameters (Inputs to a Function)

Parameters act like **placeholders** or **variables** that receive values when the function is called.
They allow data to be passed **into** a function.

2. Return Values (Output from a Function)

A **return value** is the **result** that a function sends back after processing.

The return statement stops the function execution and **returns a value** to the caller.

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

Answer:

An **array** in JavaScript is a **collection of elements (values)** stored in a **single variable**.
Each element in an array has a **numeric index**, starting from **0**.

Arrays are used to store **multiple values** (like numbers, strings, objects, etc.) in a single variable.

Example: let fruits = ["Apple", "Banana", "Cherry"];

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

Answer:

In JavaScript, arrays come with several **built-in methods** that make it easy to **add** or **remove** elements.

The four commonly used ones are:

- push() → Add to the end
- pop() → Remove from the end
- shift() → Remove from the beginning
- unshift() → Add to the beginning

These methods directly **modify** the original array.

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

Answer:

In JavaScript, an object is a collection of key–value pairs.
Each key (also called a property) is a string or symbol, and each value can be of any data type — string, number, array, another object, or even a function.

- Objects are used to represent real-world entities with properties and behaviors.

Example of an Object

```
let person = {
  name: "Aman",
  age: 22,
  city: "Ahmedabad"
};
```

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

Answer:

In JavaScript, objects store data as key–value pairs.
You can **access**, add, modify, or delete these properties using:

1. **Dot Notation (.)**
2. **Bracket Notation ([ ])**

Both notations are used to work with object properties, but they differ slightly in usage and flexibility.

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

Answer:

In JavaScript, events are actions or occurrences that happen in the browser, often as a result of user interaction or the browser itself.
JavaScript can detect and respond to these events using event handling.

Event handling means writing code that executes in response to a specific event.

For example:
When a user clicks a button → run a function that displays a message.

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

Answer:

The addEventListener() method in JavaScript allows you to attach an event handler to a specific HTML element without overwriting existing event handlers.
It makes web pages more interactive by responding to user actions like clicks, typing, mouse movement, etc.

Syntax:

```
element.addEventListener(event, function, useCapture);
```

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

Answer:

The DOM **(**Document Object Model**)** is **a** programming interface provided by the browser that represents the structure of an HTML or XML document as a tree of objects**.**

Each element (like <p>, <div>, <img>, etc.) in the webpage becomes a node (object) in this tree.
JavaScript can access, modify, add, or remove these nodes dynamically — allowing web pages to become interactive and dynamic.

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

Answer:

In JavaScript, to **access and manipulate HTML elements**, we first need to **select** them from the **DOM (Document Object Model)**.
The most common methods for selecting elements are:

1. document.getElementById()
2. document.getElementsByClassName()
3. document.querySelector()

Each method serves a different purpose and returns elements in different formats.

- The getElementById() method selects a **single HTML element** based on its **unique id attribute**.
- The getElementsByClassName() method selects **all elements** that have a specific **class name**.
- The querySelector() method returns **the first element** that matches a given **CSS selector**.

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

Answer:

In JavaScript, both **setTimeout()** and **setInterval()** are **built-in timing functions** that allow you to **control when code executes** — useful for animations, notifications, and delayed actions.

These functions are part of the **Window** object and are commonly used for **timing events** in web applications.

- The **setTimeout()** function is used to **execute a piece of code once** after a specified **delay (in milliseconds)**.
- The **setInterval()** function repeatedly executes a piece of code **at a fixed time interval** until stopped.

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

Answer:

The setTimeout() function in JavaScript is used to **execute a function once after a specified delay** (in milliseconds).
1 second = 1000 milliseconds, so a **2-second delay = 2000 milliseconds**.

Syntax:

```
setTimeout(function, delay);
```

- **function** → The code or function to be executed.
- **delay** → Time in milliseconds before running the function.

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

Answer:

Error handling in JavaScript is the process of managing and responding to runtime errors — so the script doesn't crash unexpectedly.

JavaScript provides the **try...catch...finally** construct to handle such errors gracefully.

Syntax:

```
try {
  // Code that might cause an error
}
catch (error) {
  // Code to handle the error
}
finally {
  // Code that runs no matter what
}
```

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

Answer:

Error handling in JavaScript is the process of detecting and managing errors that occur during program execution — without stopping the entire application.

Since JavaScript runs in the **browser (client-side)** or **server (Node.js)**, unhandled errors can **break functionality**, **confuse users**, and **affect performance**.

Example:

```
try {
  console.log(unknownVar);
} catch (error) {
  console.log("Handled error: " + error.message);
}
console.log("App continues running...");
```

Submitted By Ansari Amanhusain K.