JavaScript Theory Assignment

1. JavaScript Introduction

Q1: What is JavaScript? Explain the role of JavaScript in web development.

Answer: JavaScript is a high-level, interpreted programming language used to make web pages interactive. It allows developers to add dynamic behavior, validate forms, manipulate content, and enhance user experience.

Q2: How is JavaScript different from other programming languages like Python or Java?

Answer: JavaScript is interpreted, dynamically typed, and mainly used in browsers (and with Node.js on servers). Python is general-purpose, beginner-friendly, and widely used in AI, data science, and backend development. Java is compiled, strongly typed, and commonly used in enterprise applications and Android development.

Q3: Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

Answer: The <script> tag is used to add JavaScript code to an HTML page. External files are linked with <script src="script.js"></script>.

---

2. Variables and Data Types

Q1: What are variables in JavaScript? How do you declare a variable using var, let, and const?

Answer: Variables are containers for storing data values. var is function-scoped and older. let is block-scoped and modern. const is block-scoped and used for constant values.

Q2: Explain the different data types in JavaScript. Provide examples for each.

Answer: String ("Hello"), Number (42), Boolean (true), Null (null), Undefined (let x;), Object ({name: "John"}), Array ([1,2,3]).

Q3: What is the difference between undefined and null in JavaScript?

Answer: Undefined means a variable is declared but not assigned. Null represents an intentional empty value.

---

3. JavaScript Operators

Q1: What are the different types of operators in JavaScript? Explain with examples.

Answer: Arithmetic (+, -, *, /, %), Assignment (=, +=, -=), Comparison (==, ===, !=, >, <), Logical (&&, ||, !).

Q2: What is the difference between == and === in JavaScript?

Answer: == compares values only, while === compares both value and type.

---

4. Control Flow (If-Else, Switch)

Q1: What is control flow in JavaScript? Explain how if-else statements work with an example.

Answer: Control flow decides which code block runs based on conditions. Example: if(x > 0) console.log("Positive"); else console.log("Negative");

Q2: Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

Answer: A switch checks a value against multiple cases and executes the matching one. It is better than multiple if-else statements when testing many fixed values.

---

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

Q1: Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

Answer: For loop runs a set number of times. While loop runs while condition is true. Do-while loop runs at least once, then checks condition.

Q2: What is the difference between a while loop and a do-while loop?

Answer: In while loop, the condition is checked first. In do-while loop, the code runs once before checking the condition.

---

6. Functions

Q1: What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Answer: Functions are reusable blocks of code. Example: function greet() { console.log("Hello"); } greet();

Q2: What is the difference between a function declaration and a function expression?

Answer: A declaration is a named function that is hoisted. An expression is stored in a variable and not hoisted.

Q3: Discuss the concept of parameters and return values in functions.

Answer: Parameters are inputs to a function. A return value is the output that a function sends back.

---

7. Arrays

Q1: What is an array in JavaScript? How do you declare and initialize an array?

Answer: An array is a collection of values stored in a single variable. Example: let arr = ["apple", "banana", "cherry"].

Q2: Explain the methods push(), pop(), shift(), and unshift() used in arrays.

Answer: push adds an element at the end. pop removes the last element. shift removes the first element. unshift adds an element at the beginning.

---

8. Objects

Q1: What is an object in JavaScript? How are objects different from arrays?

Answer: An object stores data as key-value pairs. Objects store properties, while arrays store indexed values.

Q2: Explain how to access and update object properties using dot notation and bracket notation.

Answer: Dot notation: obj.name. Bracket notation: obj["name"]. Update: obj.age = 25.

---

9. JavaScript Events

Q1: What are JavaScript events? Explain the role of event listeners.

Answer: Events are user actions such as clicks or keypresses. Event listeners detect these actions and execute code in response.

Q2: How does the addEventListener() method work in JavaScript? Provide an example.

Answer: addEventListener attaches a function to an event. Example: btn.addEventListener("click", function(){ alert("Clicked"); });

---

10. DOM Manipulation

Q1: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

Answer: The DOM is a structured representation of HTML elements. JavaScript interacts with it to modify content, style, and structure dynamically.

Q2: Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.

Answer : getElementById("id") selects a single element. getElementsByClassName("class") selects a collection of elements. querySelector("selector") selects the first matching element.

---

11. JavaScript Timing Events (setTimeout, setInterval)

Q1: Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?

Answer: setTimeout runs a function once after a delay. setInterval repeats a function at regular intervals.

Q2: Provide an example of how to use setTimeout() to delay an action by 2 seconds.

Answer: Example: setTimeout(() => console.log("Hello"), 2000);

---

12. JavaScript Error Handling

Q1: What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.

Answer: Error handling prevents program crashes. Try contains code that may fail. Catch handles the error. Finally always runs. Example: try { let x = y + 1; } catch(e) { console.log("Error:", e); } finally { console.log("Done"); }

Q2: Why is error handling important in JavaScript applications?

Answer: Error handling ensures smooth execution, prevents crashes, and provides meaningful error messages to users.