JAVA SCRIPT ASSIGNMENT

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

- JavaScript is a client-side scripting language used in web development.
- Adds interactivity, dynamic updates, and control to HTML/CSS content.
- Enables features like form validation, animations, and API interactions.

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

- JavaScript is browser-based; Python/Java are mostly server-side.
- JS is dynamically typed; Java is statically typed.
- Python emphasizes readability; JS focuses on event-driven logic.
- JS runs in browsers without compilation; Java needs JVM; Python needs an interpreter.

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

- `<script>` embeds or references JavaScript in HTML.
- Inline script: `<script>console.log('Hi');</script>`
- External file: `<script src="main.js"></script>` placed before `</body>` or in `<head>` with `defer`.

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

- Variables store data values.

- `var`: function-scoped, hoisted.
- `let`: block-scoped, modern.
- `const`: block-scoped, read-only (cannot reassign).

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

- String: `"Hello"`
- Number: `123`
- Boolean: `true`, `false`
- Undefined: variable declared but not assigned
- Null: explicitly empty (`let x = null`)
- Object: `{name: 'John'}`
- Array: `[1, 2, 3]`
- Symbol: `Symbol('id')`
- BigInt: `1234567890123456789On`

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

- `undefined`: variable declared but not assigned a value.
- `null`: intentional absence of value; manually set.
- `typeof undefined` → `'undefined'`, `typeof null` → `'object'`

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

a. Arithmetic Operators

- `+`, `-`, `*`, `/`, `%`, `**`
- Example: `5 + 2 = 7`

b. Assignment Operators

- `=`, `+=`, `-=`, `*=`, `/=`
- Example: `x += 5` is `x = x + 5`

c. Comparison Operators

- `==`, `===`, `!=`, `!==`, `<`, `>`, `<=`, `>=`
- Example: `5 === '5'` → false

d. Logical Operators

- `&&`, `||`, `!`
- Example: `true && false` → false

8. What is the difference between == and === in JavaScript?

- `==`: loose equality (type conversion allowed).
- `===`: strict equality (no type conversion).
- Example: `5 == '5'` → true, `5 === '5'` → false

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

- Control flow directs the order of execution.
- `if-else` checks conditions:

```
let age = 18;
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

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

- `switch` checks multiple cases for a variable.
- Better than many `if-else` when checking exact values.

```
let color = "red";
switch(color) {
  case "red":
    console.log("Stop");
    break;
  case "green":
    console.log("Go");
    break;
  default:
    console.log("Wait");
}
```

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

a. for loop

```
for (let i = 0; i < 3; i++) {
  console.log(i);
}
```

b. while loop

```
let i = 0;
while (i < 3) {
  console.log(i);
  i++;
}
```

c. do-while loop

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 3);
```

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

- `while`: condition checked before execution.
- `do-while`: condition checked after execution (executes at least once).

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

- A function is a block of code that performs a task.

Syntax:

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

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

- **Function Declaration:**

```
function sayHi() { }
```

- Hoisted (can be used before declared).
- **Function Expression:**

```
let sayHi = function() { };
```

- Not hoisted.

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

- Parameters: inputs passed into a function.
- Return value: output returned by function.

```
function add(a, b) {
  return a + b;
}
let result = add(2, 3); // 5
```

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

- Array: ordered list of values.

- Syntax: `let arr = [1, 2, 3];`

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

- `push()`: adds at end → `arr.push(4)`
- `pop()`: removes from end → `arr.pop()`
- `shift()`: removes from start → `arr.shift()`
- `unshift()`: adds at start → `arr.unshift(0)`

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

- Object: collection of key-value pairs.
- Array: ordered list; Object: unordered properties.

let person = { name: "John", age: 30 };

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

- **Dot notation:** `person.name = "Doe"`
- **Bracket notation:** `person["age"] = 25`

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

- Events: actions like clicks, mouse moves, form submissions.

- Event listeners: functions that respond to events.

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

- Adds a handler to an event without overwriting others.

```
document.getElementById("btn").addEventListener("click", function() {
  alert("Clicked!");
});
```

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

- DOM is a tree structure of the web page.
- JavaScript can manipulate DOM to change content, style, structure.

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

- `getElementById("id")`: returns single element by ID.
- `getElementsByClassName("class")`: returns HTMLCollection.
- `querySelector("selector")`: returns first matching element (CSS selector).

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

- `setTimeout(fn, ms)`: runs `fn` once after `ms` milliseconds.

- `setInterval(fn, ms)`: runs `fn` repeatedly every `ms` milliseconds.

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

```
setTimeout(function() {
  console.log("Executed after 2 seconds");
}, 2000);
```

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

- Handles runtime errors to prevent crashes.

```
try {
  let x = y + 1; // y is not defined
} catch (err) {
  console.log("Error:", err.message);
} finally {
  console.log("Cleanup");
}
```

27. Why is error handling important in JavaScript applications?

- Prevents app from breaking unexpectedly.
- Provides better user experience.
- Helps in debugging and maintaining stable apps.