Name: Sandip Kumar Shah
Roll No: 23081029
Csit-5th sem

# Questions and Answers

1. What is Scripting Language? Differentiate between Scripting language and Programming Language.
   Answer: A scripting language is a programming language that runs tasks automatically without needing to be compiled first. It's used to control apps or webpages.
   Differences:

- Scripting languages (like JavaScript, PHP) are interpreted at runtime; programming languages (like C++, Java) are usually compiled first.
- Scripting languages automate small tasks, like updating a webpage; programming languages build big apps, like games or software.
- Scripting languages often work inside another program (like a browser); programming languages can run on their own.

2. Describe how web application works.

   Answer: A web application runs on a client-server system. The user (client) uses a browser to visit a webpage. The browser sends a request to the server. The server processes it, often using scripts like PHP or JavaScript, and sends back data (like HTML or images). The browser then shows the webpage, and JavaScript can add interactivity, like updating a form without reloading.
   Example: When you submit a form, the browser sends it to the server, which checks the data and sends back a response, like "Success".

3. Why do we need Scripting Language? Describe and differentiate between different types of scripting language.
   Answer: Scripting languages are needed to automate tasks, add

interactivity (like buttons that respond), and manage webpage content easily.
Types:

- Client-side scripting (like JavaScript): Runs in the browser to update pages fast, e.g., showing a popup.
- Server-side scripting (like PHP, Python): Runs on the server to handle data or user logins, e.g., saving a form.
  Differences:
- Client-side affects what users see instantly; server-side manages behind-the-scenes tasks.
- Client-side needs a browser; server-side needs a server.

4. Write briefly about JavaScript explaining features and limitations of JavaScript.

   Answer: JavaScript is a language that makes webpages interactive, like responding to clicks.
   Features:

- Fast: Runs quickly in browsers.
- Event-driven: Reacts to user actions, like mouse clicks.
- Works everywhere: Runs on any browser or device.
  Limitations:
- Security: Can be risky if not coded carefully, as it runs in browsers.
- User control: Users can turn it off, stopping some features.
- Limited power: Can’t access files directly like other languages.

5. What is a variable in JavaScript? Explain how can we create variables in JavaScript. Differentiate between let and var syntax along with an example.
   Answer: A variable is a container for data, like numbers or text, used in JavaScript code.
   Create variables with:

- var: Older way, less strict.
- let: Modern, limits where it’s used.
- const: For data that won’t change.
  var vs let:
- var works in the whole function or globally; let only works inside its block (like inside {}).
- var can be redeclared; let can’t.

```
var x = 5;
let y = 10;
if (true) {
var x = 15;
let y = 20;
console.log(x, y);
}
console.log(x, y);
```

6. Explain different methods to embed a JavaScript code in HTML code with example.

   Answer: JavaScript can be added to HTML in three ways:

- Inline: Code in an HTML tag’s event, like a button click.
  `<button onclick="alert('Hi!')">Click Me</button>`
- Internal: Code inside a <script> tag in HTML.

```
<script> function sayHi() { alert("Hello!"); } </script>

<button onclick="sayHi()">Greet</button>
```

- External: Code in a separate .js file linked with <script>.

```
<script src="mycode.js"></script>

<button onclick="sayHi()">Greet</button>
```

mycode.js:

Name: Sandip Kumar Shah
Roll No: 23081029
Csit-5th sem

```
function sayHi() {
alert("Hello from file!");
}
```

7. Discuss the different types of loops used in JavaScript. What is the use of continue and break?

   Answer: JavaScript loops run code multiple times:

- for: Repeats a set number of times.
  ```
  for (let i = 0; i < 3; i++) {
  console.log(i);
  }
  ```
- while: Repeats while a condition is true.
  ```
  let i = 0;
  while (i < 3) {
  console.log(i);
  i++;
  }
  ```
- do...while: Runs at least once, then checks condition.
  ```
  let i = 0;
  do {
  console.log(i);
  i++;
  } while (i < 3);
  ```
- for...of: Loops through arrays or lists.
  ```
  let colors = ["red", "blue"];
  for (let color of colors) {
  console.log(color);
  }
  ```
- forEach: Runs a function for each array item.
  break: Stops the loop completely.
  ```
  for (let i = 0; i < 5; i++) {
  if (i == 3) break;
  ```

```
console.log(i);
}
```

continue: Skips one loop cycle.

```
for (let i = 0; i < 5; i++) {
if (i == 2) continue;
console.log(i);
}
```

8. What are the different data types in JavaScript? Explain the different operators you are familiar with in JavaScript.
   Answer: JavaScript data types:

- string: Text, like "hello".
- number: Numbers, like 42 or 3.14.
- boolean: true or false.
- null: Empty value.
- undefined: No value set.
- object: Complex data, like {name: "Ana"}.
- symbol: Unique values.

  Operators:
- Arithmetic: + (add), - (subtract), * (multiply), / (divide).
- Comparison: == (equal), === (strict equal), != (not equal), >, <.
- Logical: && (and), || (or), ! (not).
- Assignment: = (set), += (add and set).

```
let x = 5 + 3;
let same = (5 === "5");
let both = (x > 0 && x < 10);
```

9. What do you mean by a loop? Describe the different types of loops known in JavaScript.

   Answer: A loop runs code several times until a condition is met.
   Types:

- for: Loops a set number of times.
```
for (let i = 0; i < 2; i++) {
console.log(i);
}
```
- while: Loops while true.
```
let i = 0;
while (i < 2) {
console.log(i);
i++;
}
```
- do...while: Runs once, then loops if true.
```
let i = 0;
do {
console.log(i);
i++;
} while (i < 2);
```
- for...in: Loops through object keys.
```
let obj = {a: 1, b: 2};
for (let key in obj) {
console.log(key);
}
```
- for...of: Loops through arrays.
```
let arr = [1, 2];
for (let val of arr) {
console.log(val);
}
```

10. Explain array creation and traversal in JavaScript.

Answer: Arrays hold multiple values in one variable.
Creation:

- Using []: Simple list.
- Using new Array(): Same, but less common.
```
let nums = [1, 2, 3];
let empty = new Array(3);
```

Traversal:

- for loop: Use index numbers.
  ```
  for (let i = 0; i < nums.length; i++) {
  console.log(nums[i]);
  }
  ```
- for...of: Direct values.
  ```
  for (let num of nums) {
  console.log(num);
  }
  ```

11. What do you understand by conditional statements? Explain different conditional statements in JavaScript.

Answer: Conditional statements run code if a condition is true.
Types:

- if: Checks one condition.
  ```
  let age = 20;
  if (age >= 18) {
  console.log("Adult");
  }
  ```
- else if: Checks more conditions.
  ```
  if (age < 13) {
  console.log("Kid");
  } else if (age < 18) {
  console.log("Teen");
  }
  ```
- else: Runs if nothing else is true.
  ```
  if (age < 18) {
  console.log("Minor");
  } else {
  console.log("Adult");
  }
  ```

- switch: Picks code by value.
  ```
  let day = 1;
  switch (day) {
  case 1:
  console.log("Monday");
  break;
  default:
  console.log("Other");
  }
  ```

12. What are functions? Explain the different ways to create a function.
    Answer: Functions are code you can run anytime by calling their name.
    Ways to create:

- Declaration: Basic function.
  ```
  function add(a, b) {
  return a + b;
  }
  console.log(add(2, 3));
  ```
- Expression: Store in a variable.
  ```
  let subtract = function(a, b) {
  return a - b;
  };
  console.log(subtract(5, 2));
  ```
- Arrow: Short, modern syntax.
  ```
  let multiply = (a, b) => a * b;
  console.log(multiply(4, 2));
  ```

13. What is recursive function? Explain the significance of arrow function in JavaScript.
    Answer: A recursive function calls itself to solve smaller parts of a problem.

```
function factorial(n) {
if (n <= 1) return 1;
return n * factorial(n - 1);
}
console.log(factorial(3));
```

Arrow function significance:

- Short syntax: Less code to write.
- Keeps ‘this’ from parent: Great for callbacks.

```
let obj = {
name: "Zoe",
show: function() {
setTimeout(() => console.log(this.name), 100);
}
};
obj.show();
```

14. What do you mean by higher order function? Explain different types of higher order functions you are familiar with.

Answer: A higher-order function uses functions as inputs or outputs.

Types:

- map: Changes each item in an array.
```
let nums = [1, 2, 3];
let doubles = nums.map(x => x * 2);
console.log(doubles);
```
- filter: Keeps items that pass a test.
```
let evens = nums.filter(x => x % 2 === 0);
console.log(evens);
```
- reduce: Combines items into one value.
```
let sum = nums.reduce((total, x) => total + x, 0);
console.log(sum);
```

Name: Sandip Kumar Shah
Roll No: 23081029
Csit-5th sem

15. What is a forEach loop? Elaborate with an example.

Answer: forEach calls a function for each item in an array.

```
let names = ["Kim", "Lee", "Sam"];
names.forEach((name, index) => {
console.log(index + ": " + name);
});
```

16. Describe properties and methods. Enlist the properties and methods of following objects in JavaScript:

Answer: Properties are data in an object, like a number. Methods are functions an object can run.

a. Array:

- Property: length
- Methods: push(), pop(), slice()

b. Boolean:

- Methods: toString(), valueOf()

c. String:

- Property: length
- Methods: toUpperCase(), charAt()

d. Math:

- Properties: PI
- Methods: random(), floor()

e. Window:

- Properties: location, document
- Methods: alert(), setTimeout()

f. Location:

- Properties: href, hostname
- Methods: reload(), replace()

g. Screen:

- Properties: width, height

h. History:

- Methods: back(), forward()

i. Navigator:

- Properties: userAgent, language

  j. RegExp:
- Properties: source
- Methods: test(), exec()

  k. Number:
- Methods: toFixed(), toString()

17. What do you understand by DOM? What are the different properties and methods associated with it?

    Answer: The DOM (Document Object Model) is how JavaScript sees and edits HTML. It's a tree of elements, like buttons or divs.

    Properties:

- innerHTML: Element's content.
- style: Element's CSS.
- value: Form input's value.

  Methods:
- getElementById(): Finds one element.
- querySelector(): Finds by CSS selector.
- createElement(): Makes new element.

```
let div = document.createElement("div");
div.innerHTML = "Hi";
document.body.appendChild(div);
```

18. Differentiate between event and event handler.

    Answer: Event: Something a user does, like clicking a button.

    Event handler: Code that runs when the event happens.

```
<button onclick="myClick()">Click</button>

<script> function myClick() { alert("Clicked!"); } </script>
```

19. What is regular expressions? Describe quantifiers and meta character with example.

    Answer: Regular expressions are patterns to match text, like

finding emails.
Quantifiers:

- *: 0 or more times.
- +: 1 or more times.
- ?: 0 or 1 time.

Meta characters:

- \d: Any digit.
- \w: Any letter or number.

```
let text = "123abc";
let regex = /\d+/;
console.log(regex.test(text));
```

20. Show an example of client side validation using JavaScript.

Answer: Client-side validation checks data in the browser.

```
<form name="myForm">
 <input type="text" name="name">
 <button onclick="validate()">Submit</button>
 </form>
<script>
function validate()
{
let name = document.forms["myForm"]["name"].value;
if (name == "")
{
alert("Name can’t be empty");
return false;
}

 }
 </script>
```

Name: Sandip Kumar Shah
Roll No: 23081029
Csit-5th sem

21. How can we access HTML objects using JavaScript? Explain with example.
Answer: JavaScript accesses HTML elements with DOM methods:

- getElementById(): By ID.
- querySelector(): By CSS selector.
- getElementsByTagName(): By tag name.

```
<div id="box">Text</div>
<script>
let div = document.getElementById("box");
div.innerHTML = "New Text";
</script>
```