+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

🧠 1. Interview Questions and Answers on JavaScript Functions

🔹 Basic Level

Q1. What is a function in JavaScript?

A:
A function is a block of reusable code that performs a specific task. It runs when "called" using its name followed by ().

Q2. How do you declare a function in JavaScript?

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

Q3. What are function parameters and arguments?

A:

- Parameters are variables in the function definition.
- Arguments are values passed when calling the function.

```
function add(a, b) { // a and b are parameters
  return a + b;
}
add(5, 10); // 5 and 10 are arguments
```

Q4. What is the difference between function declaration and function expression?

```
// Function Declaration
function sayHi() {
  console.log("Hi");
}

// Function Expression
const sayHello = function () {
  console.log("Hello");
};
```

Function declarations are hoisted, expressions are not.

Q5. What is a return statement in JavaScript?

A:
It ends function execution and returns a value.

◆ **Intermediate Level**

Q6. What is an arrow function?

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

Q7. Can JavaScript functions be nested?

A: Yes. Inner functions can access outer function variables due to closures.

Q8. What is a callback function?

A:
A function passed as an argument to another function.

```
function greet(name, callback) {
  console.log("Hello " + name);
  callback();
}
greet("Max", () => console.log("Welcome!"));
```

Q9. What is the arguments object?

A:
An array-like object accessible inside all non-arrow functions containing passed arguments.

◆ **Advanced Level**

Q10. What are IIFE functions?

A: Immediately Invoked Function Expressions

```
(function () {
  console.log("Runs Immediately");
})();
```

Q11. What is a pure function?

A function where:

- Output depends only on input.

+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

- No side effects.

Q12. What is recursion in JavaScript?

A function calling itself to solve a problem.

🧪 2. 25 Practical Tasks for JavaScript Functions (Step-by-Step)

✅ Beginner (Tasks 1–10)

1. Create a function to add two numbers

```
function add(a, b) {
  return a + b;
}
console.log(add(5, 3));
```

---

2. Function to check if a number is even or odd

```
function isEven(n) {
  return n % 2 === 0 ? "Even" : "Odd";
}
```

---

3. Function to find the square of a number

```
function square(n) {
  return n * n;
}
```

---

4. Function to return the larger of two numbers

```
function max(a, b) {
  return a > b ? a : b;
}
```

---

5. Create a function to greet with name

```
function greet(name) {
  console.log(`Hello, ${name}`);
}
```

---

6. Function to convert Celsius to Fahrenheit

```
function toFahrenheit(c) {
  return (c * 9) / 5 + 32;
}
```

---

7. Function with default parameters

```
function welcome(name = "Guest") {
  console.log(`Welcome, ${name}`);
}
```

---

8. Function expression to multiply two numbers

```
const multiply = function (a, b) {
  return a * b;
};
```

---

9. Arrow function to divide

```
const divide = (a, b) => a / b;
```

---

10. Check if number is positive, negative, or zero

```
function checkSign(n) {
  if (n > 0) return "Positive";
  if (n < 0) return "Negative";
  return "Zero";
}
```

🔁 Intermediate (Tasks 11–20)

11. Recursive function for factorial

```
function factorial(n) {
  if (n === 0) return 1;
  return n * factorial(n - 1);
}
```

---

12. Function to reverse a string

```
function reverseStr(str) {
  return str.split("").reverse().join("");
}
```

---

13. Function to check if string is a palindrome

```
function isPalindrome(str) {
  return str === str.split("").reverse().join("");
}
```

---

14. Find sum of array elements using function

```
function sumArray(arr) {
  let sum = 0;
  for (let num of arr) sum += num;
  return sum;
}
```

---

15. Function to find min and max from array

```
function minMax(arr) {
  return { min: Math.min(...arr), max: Math.max(...arr) };
}
```

---

16. Function that accepts another function (callback)

```
function processUser(name, cb) {
  cb(name);
}
processUser("Max", (name) => console.log(`Welcome ${name}`));
```

---

17. IIFE function to print date

```
(function () {
  console.log(new Date());
})();
```

---

18. Arrow function that returns square of array

```
const squareArray = (arr) => arr.map((x) => x * x);
```

---

19. Function with rest parameters

```
function total(...nums) {
  return nums.reduce((a, b) => a + b);
}
```

---

20. Function to count vowels in a string

```
function countVowels(str) {
  return (str.match(/[aeiou]/gi) || []).length;
}
```

🔥 Advanced (Tasks 21–25)

21. Memoized Fibonacci using closure

```
function memoFib() {
  const memo = {};
  return function fib(n) {
    if (n in memo) return memo[n];
    if (n < 2) return n;
    return (memo[n] = fib(n - 1) + fib(n - 2));
  };
}
const fib = memoFib();
```

---

22. Function to debounce user input (simulation)

```
function debounce(func, delay) {
  let timeout;
  return (...args) => {
    clearTimeout(timeout);
    timeout = setTimeout(() => func(...args), delay);
  };
}
```

---

23. Function returning another function (closure)

```
function makeAdder(x) {
  return function (y) {
    return x + y;
  };
}
const add5 = makeAdder(5);
```

---

24. Create a custom map function

```
function customMap(arr, fn) {
  const result = [];
  for (let i = 0; i < arr.length; i++) {
    result.push(fn(arr[i]));
  }
  return result;
}
```

---

25. Function chaining example

```
function chain(val) {
  return {
    add(n) {
      val += n;
      return this;
    },
    sub(n) {
      val -= n;
      return this;
    },
    result() {
      return val;
    },
  };
}
console.log(chain(5).add(3).sub(2).result()); // 6
```