1. Function with Default Arguments

```
function greet(name = "Guest", message = "Welcome!") {
    return `${name}, ${message}`;
}

console.log(greet("Alice"));
console.log(greet());
```

What will be the output of the above code?

2. Rest Operator in Function Arguments

```
function sum(...numbers) {
    return numbers.reduce((acc, curr) => acc + curr, 0);
}

console.log(sum(1, 2, 3));
console.log(sum(5, 10));
```

What will be the output of the above code?

3. Positional vs Named Arguments

```
function displayInfo(firstName, lastName, age) {
    return `${firstName} ${lastName} is ${age} years old.`;
}

console.log(displayInfo("John", "Doe", 30));
```

```
console.log(displayInfo("Jane", "Smith", 25));
```

What will be the output of the above code?

4. Object with Functions

```
const person = {
    name: "Alex",
    greet() {
        return `Hello, my name is ${this.name}`;
    }
};

console.log(person.greet());
```

What will be the output of the above code?

5. Closures with Function Returning Function

```
function outerFunc(outerValue) {
    return function innerFunc(innerValue) {
        return outerValue + innerValue;
    };
}

const addFive = outerFunc(5);
console.log(addFive(10));
console.log(addFive(3));
```

What will be the output of the above code?

6. Scope with Nested Functions

```
let outerVar = "I am outside";

function outer() {
    let outerVar = "I am inside";
    function inner() {
        return outerVar;
    }
    return inner();
}

console.log(outer());
```

What will be the output of the above code?

7. Default Arguments with Rest Operator

```
function multiply(factor = 2, ...numbers) {
    return numbers.map(num => num * factor);
}

console.log(multiply(3, 1, 2, 3));
console.log(multiply(undefined, 4, 5));
```

What will be the output of the above code?

8. Rest Parameters with Positional Arguments

```
function combine(first, second, ...rest) {
    return [first, second, ...rest];
}

console.log(combine(1, 2, 3, 4, 5));
console.log(combine("a", "b", "c"));
```

What will be the output of the above code?

9. Object with Functions and Closures

```
const counter = {
    count: 0,
    increment() {
        return ++this.count;
    },
    reset() {
        return this.count = 0;
    }
};

console.log(counter.increment());
console.log(counter.increment());
console.log(counter.reset());
console.log(counter.increment());
```

What will be the output of the above code?

10. Scope and Default Arguments

```
let x = 10;

function testScope(a, b = x) {
    let x = 20;
    return a + b;
}

console.log(testScope(5));
```

What will be the output of the above code?

11. Rest Operator with Arrow Functions

```
const joinStrings = (...strings) => strings.join(" ");

console.log(joinStrings("Hello", "World"));
console.log(joinStrings("", "is", "awesome"));
```

What will be the output of the above code?

12. Function with Multiple Default Arguments

```
function calculateArea(length = 5, width = 10) {
    return length * width;
}
```

```
console.log(calculateArea(7));
console.log(calculateArea());
```

What will be the output of the above code?

13. Rest Operator with No Arguments

```
function logAll(...args) {
    return args.length;
}

console.log(logAll());
console.log(logAll(1, 2, 3));
```

What will be the output of the above code?

14. Function Expression with Default Parameters

```
const multiply = function(a = 1, b = 2) {
    return a * b;
};

console.log(multiply(3, 4));
console.log(multiply(5));
```

What will be the output of the above code?

15. Closure with Counter

```
function createCounter() {
    let count = 0;
    return function() {
        return ++count;
    };
}

const counter = createCounter();
console.log(counter());
console.log(counter());
console.log(counter());
```

What will be the output of the above code?

16. Scope of Variables in Nested Functions

```
let name = "Outside";

function outerFunction() {
    let name = "Inside";
    function innerFunction() {
        return name;
    }
    return innerFunction();
}

console.log(outerFunction());
```

What will be the output of the above code?

17. Object Method and `this` Keyword

```
const car = {
    brand: "Toyota",
    getBrand() {
        return this.brand;
    }
};

const getBrand = car.getBrand;
console.log(getBrand());
```

What will be the output of the above code?

18. Rest Operator with Mixed Parameters

```
function listColors(color1, color2, ...otherColors) {
    return otherColors;
}

console.log(listColors("Red", "Blue", "Green", "Yellow"));
```

What will be the output of the above code?

19. Default Arguments and Undefined

```
function displayMessage(message = "Hello") {
    return message;
}
```

```
console.log(displayMessage(undefined));
console.log(displayMessage("Hi"));
```

What will be the output of the above code?

20. Closure with Outer Function Arguments

```
function createMultiplier(multiplier) {
    return function(number) {
        return number * multiplier;
    };
}

const multiplyBy3 = createMultiplier(3);
console.log(multiplyBy3(5));
```

What will be the output of the above code?

21. Scope of Variables in Function Expressions

```
let value = 10;

const calculate = function(a) {
    let value = 20;
    return a + value;
};

console.log(calculate(5));
console.log(value);
```

What will be the output of the above code?

22. Rest Parameters with Spread Operator

```
function sumNumbers(...nums) {
    return nums.reduce((a, b) => a + b, 0);
}

const numbers = [1, 2, 3];
console.log(sumNumbers(...numbers));
```

What will be the output of the above code?

23. Positional and Named Parameters with Default Values

```
function displayUser(name = "Unknown", age = 18) {
    return `${name} is ${age} years old`;
}

console.log(displayUser("John", 25));
console.log(displayUser(undefined, 30));
```

What will be the output of the above code?

24. Scope Chain with Function Inside Function

```
let number = 100;
```

```
function outerFunc() {
    let number = 50;
    function innerFunc() {
        return number;
    }
    return innerFunc();
}

console.log(outerFunc());
```

What will be the output of the above code?

25. Object Methods and Rest Parameters

```
const calculator = {
    add(...nums) {
        return nums.reduce((a, b) => a + b, 0);
    }
};

console.log(calculator.add(5, 10, 15));
```

What will be the output of the above code?

26. Function with Closure and Return

```
function createAdder(x) {
    return function(y) {
        return x + y;
    };
}

const add10 = createAdder(10);
console.log(add10(5));
```

```
console.log(add10(20));
```

What will be the output of the above code?

27. Default Arguments and Null Values

```
function displayInfo(name = "Anonymous", age = 18) {
    return `${name} is ${age} years old.`;
}

console.log(displayInfo(null, 25));
console.log(displayInfo(undefined, null));
```

What will be the output of the above code?

28. Rest Parameters and Returning Arrays

```
function createArray(...items) {
    return items;
}

console.log(createArray(1, 2, 3, 4));
console.log(createArray("a", "b", "c"));
```

What will be the output of the above code?

29. Function with Outer and Inner Variables

```
let a = 1;

function outer() {
    let a = 2;
    function inner() {
        return a;
    }
    return inner();
}

console.log(outer());
```

What will be the output of the above code?

30. Default Parameters with Expressions

```
function calculateTotal(price, tax = price * 0.05) {
    return price + tax;
}

console.log(calculateTotal(100));
console.log(calculateTotal(200, 30));
```

What will be the output of the above code?

31. Closure with Multiple Levels

```
function outer() {
```

```
    let outerVar = "Outer";
    return function middle() {
        let middleVar = "Middle";
        return function inner() {
            return `${outerVar} ${middleVar}`;
        };
    };
}

console.log(outer()()());
```

What will be the output of the above code?

32. Rest Operator with Default Arguments

```
function listNumbers(first, second = 2, ...rest) {
    return rest;
}

console.log(listNumbers(1, 3, 4, 5, 6));
console.log(listNumbers(1));
```

What will be the output of the above code?

33. Function with Closure Capturing Arguments

```
function createCounter(start) {
    let count = start;
    return function() {
        return ++count;
    };
}

const counter1 = createCounter(0);
```

```
const counter2 = createCounter(5);

console.log(counter1());
console.log(counter2());
```

What will be the output of the above code?

34. Object Method with `this` and Rest Parameters

```
const robot = {
    name: "Robo",
    greet(...messages) {
        return `${this.name}: ${messages.join(" ")}`;
    }
};

console.log(robot.greet("Hello", "World!"));
```

What will be the output of the above code?

35. Default Parameters and Logical OR

```
function getValue(value = 10) {
    return value || 20;
}

console.log(getValue(0));
console.log(getValue(15));
```

What will be the output of the above code?

36. Closures with Functions and Scope

```
let globalValue = "Global";

function makeFunc() {
    let localValue = "Local";
    return function() {
        return localValue;
    };
}

const myFunc = makeFunc();
console.log(myFunc());
```

What will be the output of the above code?

37. Rest Operator and Returning Length

```
function countItems(...items) {
    return items.length;
}

console.log(countItems(1, 2, 3, 4));
console.log(countItems("a", "b"));
```

What will be the output of the above code?

38. Closures with Parameterized Outer Function

```
function createGreeter(greeting) {
    return function(name) {
        return `${greeting}, ${name}`;
    };
}

const sayHello = createGreeter("Hello");
console.log(sayHello("Alice"));
```

What will be the output of the above code?

39. Function with Default Arguments and Strings

```
function createSentence(subject = "Someone", verb = "does", object = "something") {
    return `${subject} ${verb} ${object}.`;
}

console.log(createSentence("The cat", "jumps", "high"));
console

.log(createSentence("The dog"));
```

What will be the output of the above code?

40. Returning Arrays with Rest Parameters

```
function makeArray(...args) {
    return args;
}

console.log(makeArray(1, 2, 3, 4));
```

What will be the output of the above code?

41. Closure with Variable from Outer Scope

```
let count = 10;

function incrementCounter() {
    return ++count;
}

console.log(incrementCounter());
console.log(incrementCounter());
```

What will be the output of the above code?

42. Default Arguments with Undefined Parameter

```
function greet(name = "Guest") {
    return `Hello, ${name}`;
}

console.log(greet(undefined));
console.log(greet("John"));
```

What will be the output of the above code?

43. Rest Operator and Function Arguments

```
function concatStrings(...strings) {
    return strings.join(", ");
}

console.log(concatStrings("apple", "banana", "cherry"));
console.log(concatStrings());
```

What will be the output of the above code?

44. Function with Nested Scope

```
let message = "Global";

function outer() {
    let message = "Outer";
    function inner() {
        let message = "Inner";
        return message;
    }
    return inner();
}

console.log(outer());
```

What will be the output of the above code?

45. Object Method and `this` Context

```
const user = {
    name: "Alice",
    getName: function() {
```

```
        return this.name;
    }
};

const getName = user.getName;
console.log(getName());
```

What will be the output of the above code?

46. Rest Parameters with Named Parameters

```
function showDetails(id, ...details) {
    return details;
}

console.log(showDetails(101, "Alice", "Developer", "NYC"));
```

What will be the output of the above code?

47. Default Arguments and Overriding

```
function calculateDiscount(price, discount = 10) {
    return price - (price * discount) / 100;
}

console.log(calculateDiscount(100));
console.log(calculateDiscount(200, 20));
```

What will be the output of the above code?

48. Closures with Private Variables

```
function secretHolder(secret) {
    return function() {
        return secret;
    };
```

```
}

const revealSecret = secretHolder("My Secret");
console.log(revealSecret());
```

What will be the output of the above code?

49. Scope of Variables and Functions

```
var globalVar = "I am global";

function outerFunction() {
    var localVar = "I am local";
    return function innerFunction() {
        return globalVar + " and " + localVar;
    };
}

console.log(outerFunction()());
```

What will be the output of the above code?

50. Function Expression with Default Arguments

```
const divide = function(a = 10, b = 2) {
    return a / b;
};

console.log(divide(20, 4));
console.log(divide(15));
```

What will be the output of the above code?

51. Object Method Using `this` Inside Function

```
const person = {
```

```
    firstName: "John",
    lastName: "Doe",
    fullName() {
        return this.firstName + " " + this.lastName;
    }
};

console.log(person.fullName());
```

What will be the output of the above code?

52. Rest Parameters with Arithmetic Operations

```
function sumAll(...numbers) {
    return numbers.reduce((a, b) => a + b, 0);
}

console.log(sumAll(1, 2, 3, 4));
console.log(sumAll());
```

What will be the output of the above code?

53. Closure with Functions Returning Functions

```
function createPowerFunction(power) {
    return function(base) {
        return Math.pow(base, power);
    };
}

const square = createPowerFunction(2);
console.log(square(4));
```

What will be the output of the above code?

54. Default Arguments with Logical OR

```
function displayScore(score = 100) {
    return score || 0;
}

console.log(displayScore(0));
console.log(displayScore(80));
```

What will be the output of the above code?

55. Rest Operator with Different Data Types

```
function mixAndMatch(...args) {
    return args.join("-");
}

console.log(mixAndMatch(1, "apple", true));
```

What will be the output of the above code?

56. Closure Capturing Outer Variable

```
let greeting = "Hello";

function greetUser() {
    return function(name) {
        return `${greeting}, ${name}`;
    };
}

const greet = greetUser();
console.log(greet("Alice"));
```

What will be the output of the above code?

57. Function with Scope and Shadowing

```
let x = 5;

function testScope() {
    let x = 10;
    return x;
}

console.log(testScope());
console.log(x);
```

What will be the output of the above code?

58. Object Method with Rest Parameters

```
const restaurant = {
    name: "Pizza Place",
    order(...items) {
        return `Ordered: ${items.join(", ")}`;
    }
};

console.log(restaurant.order("Pizza", "Pasta"));
```

What will be the output of the above code?

59. Default Arguments with String Concatenation

```
function welcomeMessage(name = "Guest") {
    return `Welcome, ${name}!`;
}

console.log(welcomeMessage());
console.log(welcomeMessage("John"));
```

What will be the output of the above code?

60. Closure with Count and Increment

```
function createCounter(start = 0) {
    let count = start;
    return function() {
        return ++count;
    };
}

const counter = createCounter(5);
console.log(counter());
console.log(counter());
```

What will be the output of the above code?

61. Rest Parameters with Array Operations

```
function findMax(...numbers) {
    return Math.max(...numbers);
}

console.log(findMax(10, 20, 30, 40));
console.log(findMax());
```

What will be the output of the above code?

62. Function Returning Objects

```
function createUser(name, age) {
    return {
        name,
        age
    };
}

const user = createUser("Alice", 25);
console.log(user);
```

What will be the output of the above code?

63. Default Parameters and Dynamic Values

```
function calculateTotal(price, tax = price * 0.1) {
    return price + tax;
}

console.log(calculateTotal(100));
console.log(calculateTotal(200, 30));
```

What will be the output of the above code?

64. Closure with Function Inside Function

```
function outer() {
    let outerVar = "Outer";

    function inner() {
        let innerVar = "Inner";
        return outerVar + " " + innerVar;
    }

    return inner();
}

console.log(outer());
```

What will be the output of the above code?

65. Rest Operator and Array Length

```
function countArgs(...args) {
    return args.length;
}

console.log(countArgs(1, 2, 3));
```

```
console.log(countArgs("a", "b", "c", "d"));
```

What will be the output of the above code?

66. Object Method with Default Parameters

```
const userProfile = {
    name: "John",
    greet(greeting = "Hi") {
        return `${greeting}, ${this.name}`;
    }
};

console.log(userProfile.greet());
console.log(userProfile.greet("Hello"));
```

What will be the output of the above code?

67. Rest Parameters and Sorting

```
function sortNumbers(...numbers) {
    return numbers.sort((a, b) => a - b);
}

console.log(sortNumbers(10, 5, 2, 8));
```

What will be the output of the above code?

68. Closures and Private Variables

```
function bankAccount(initialBalance) {
    let balance = initialBalance;
    return function() {
        return balance;
    };
}
```

```
const account = bankAccount(1000);
console.log(account());
```

What will be the output of the above code?

69. Default Arguments with Multiple Parameters

```
function processPayment(amount, fee = 2, discount = 0) {
    return amount + fee - discount;
}

console.log(processPayment(100, undefined, 10));
console.log(processPayment(200, 5));
```

What will be the output of the above code?

70. Function Returning Function with Parameters

```
function multiplier(factor) {
    return function(number) {
        return number * factor;
    };
}

const double = multiplier(2);
console.log(double(5));
```

What will be the output of the above code?