1. Function Expression Hoisting

```
console.log(typeof test);
var test = function() { return 5; };
```

2. Lexical Scoping with Let and Var

```
let x = 5;
function scopeTest() {
  console.log(x);
  var x = 10;
}
scopeTest();
```

3. Closure Inside a Loop

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

let increment = createIncrementer();
console.log(increment());
console.log(increment());
```

4. Variable Hoisting

```
console.log(foo);
```

```
var foo = 10;
```

5. Block Scope with Let

```
function blockScope() {
  let x = 10;
  if (true) {
    let x = 20;
    console.log(x);
  }
  console.log(x);
}
blockScope();
```

6. Function Scope and Closures

```
function outer() {
  var outerVar = "outer";
  return function inner() {
    console.log(outerVar);
  };
}
var innerFunc = outer();
innerFunc();
```

7. Variable Shadowing

```
let a = 10;
function shadow() {
  let a = 5;
  console.log(a);
}
```

```
shadow();
console.log(a);
```

8. Closure and Multiple Functions

```
function outerFunc() {
  let count = 0;
  return {
    increment: function() {
      count++;
      console.log(count);
    },
    decrement: function() {
      count--;
      console.log(count);
    }
  };
}

let counter = outerFunc();
counter.increment();
counter.decrement();
```

9. Lexical Scoping with Functions

```
function outerFunc() {
  let x = 10;
  function innerFunc() {
    console.log(x);
  }
  return innerFunc;
}
let result = outerFunc();
result();
```

10. Hoisting with Function Declarations

```
function hoistingTest() {
  console.log(x);
  var x = 10;
}
hoistingTest();
```

11. Functions and Default Parameters

```
function add(a = 5, b = a * 2) {
  return a + b;
}
console.log(add(3));
console.log(add());
```

12. Closure with State

```
function createCounter() {
  let count = 0;
  return function() {
    return ++count;
  };
}
const counter = createCounter();
console.log(counter());
console.log(counter());
```

13. Hoisting in Functions

```
function testHoist() {
  console.log(foo);
  var foo = 10;
}
testHoist();
```

14. Self-Invoking Function with Closure

```
(function() {
  var x = 5;
})();
console.log(x);
```

15. Closure and Returning Functions

```
function multiplierFactory(factor) {
  return function(num) {
    return num * factor;
  };
}
let double = multiplierFactory(2);
console.log(double(4));
```

16. Block Scope and Let

```
let x = 5;
function checkScope() {
  let x = 10;
  console.log(x);
}
checkScope();
console.log(x);
```

17. Functions and Parameter Hoisting

```
function hoistParam(param) {
  console.log(param);
  var param = 20;
}
hoistParam(10);
```

18. Nested Functions and Scope Chain

```
let x = 1;
function outer() {
  let x = 2;
  function inner() {
    console.log(x);
  }
  inner();
}
outer();
```

19. Arguments Object

```
function testArgs(a, b, c) {
  console.log(arguments[0], arguments[1], arguments[2]);
}
testArgs(1, 2);
```

20. Variable Shadowing in Blocks

```
let x = 5;
{
  let x = 10;
  console.log(x);
}
console.log(x);
```

21. Closures and Lexical Scope

```
function outer() {
  let a = 1;
  function inner() {
    console.log(a);
  }
  return inner;
}
let result = outer();
result();
```

22. Hoisting in Function Expressions

```
console.log(typeof foo);
var foo = function() {
  return 5;
};
```

23. Function Declaration Hoisting

```
function hoistFunc() {
  console.log(foo());
  function foo() {
    return 10;
  }
}
hoistFunc();
```

24. Closures and Arguments

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

const increment = createFunction();
console.log(increment());
console.log(increment());
```

25. Scope and Variable Shadowing

```
let x = 10;
function checkScope() {
  let x = 20;
  console.log(x);
}
checkScope();
console.log(x);
```

26. Immediate Invocation and Scope

```
(function() {
  var x = 5;
})();
console.log(x);
```

27. Closures with Nested Functions

```
function outerFunction() {
  var outerVar = "I'm outer";
  return function innerFunction() {
    console.log(outerVar);
  };
}
outerFunction()();
```

28. Hoisting of Function Expressions

```
console.log(typeof func);
var func = function() {
  return 5;
};
```

29. Block Scope with Let

```
let x = 10;
if (true) {
  let x = 20;
  console.log(x);
}
console.log(x);
```

30. Closures and State Management

```
function counter() {
  let count = 0;
  return function() {
    return ++count;
  };
}
const increment = counter();
console.log(increment());
console.log(increment());
```

31. Hoisting and Function Parameters

```
function testHoisting(param) {
  console.log(param);
  var param = 20;
}
testHoisting(10);
```

32. Variable Shadowing in Nested Functions

```
let a = 1;
function outer() {
  let a = 2;
  function inner() {
    console.log(a);
  }
  inner();
}
outer();
```

33. Closures with Multiple Functions

```
function createCounter() {
  let count = 0;
  return {
    increment: function() {
      count++;
      console.log(count);
    },
    decrement: function() {
      count--;
      console.log(count);
    }
  };
}

const counter = createCounter();
counter.increment();
counter.decrement();
```

34. Lexical Scope and Closures

```
let a = 1;
function outerFunc() {
  let a = 2;
  return function() {
    console.log(a);
  };
}
let result = outerFunc();
result();
```

35. Hoisting and Function Declarations

```
console.log(test());
function test() {
  return 5;
}
```

36. Closures with Functions Returning Functions

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

let double = createMultiplier(2);
console.log(double(4));
```

37. Block Scope with Let and Var

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

38. Closures and State Encapsulation

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

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

39. Function Expressions and Hoisting

```
console.log(typeof func);
var func = function() {
  return 10;
};
```

40. Function Scope and Hoisting

```
function scopeTest() {
  console.log(x);
  var x = 10;
}
scopeTest();
```

41. Closure and Returning Functions

```
function multiplierFactory(factor) {
  return function(num) {
    return num * factor;
  };
}
let double = multiplierFactory(2);
console.log(double(5));
```

42. Lexical Scoping and Function Variables

```
let x = 10;
function outerFunc() {
  console.log(x);
}
function test() {
  let x = 20

;
  outerFunc();
}
test();
```

43. Scoping and Temporal Dead Zone

```
console.log(a);
let a = 10;
```

Question: What is the output and why? Explain the concept of the "Temporal Dead Zone."

44. Hoisting in Nested Functions

```
function outer() {
  console.log(a);
  var a = 10;
  function inner() {
    console.log(a);
    var a = 20;
  }
  inner();
}
outer();
```

Question: What will be printed and why? Discuss how hoisting affects variables in nested functions.

45. Closure with Function Re-Assignment

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

let c1 = counter;
let c2 = counter;

console.log(c1());
console.log(c2());
console.log(c1());
```

Question: What will be the output and why? Analyze the behavior of closures when functions are assigned to multiple variables.

46. Variable Scope in Loops (Var vs. Let)

```
for (var i = 0; i < 3; i++) {
  for (let i = 0; i < 2; i++) {
    console.log(i);
  }
}
console.log(i);
```

Question: What will be printed and why? Explain the difference between `var` and `let` in loop scoping.

47. Lexical Scope and Closures

```
function outer() {
  let a = 1;
  return function inner() {
    return a++;
  };
}

const x = outer();
console.log(x());
console.log(x());
console.log(a);
```

Question: What will be the output and why? Analyze the closure formed by `inner()` and how it impacts variable access.

48. Re-declaration with Var and Let

```
var x = 10;
let x = 20;
console.log(x);
```

Question: What will happen here, and why? Discuss why this causes an error and the difference between `var` and `let` in terms of redeclaration.

49. Function and Block Scope

```
function outer() {
  var x = 10;
  if (true) {
    let x = 20;
    console.log(x);
  }
  console.log(x);
}
outer();
```

Question: What will be printed and why? Explain how block scope works with `let` and `var`.

50. Const with Objects

```
const obj = { a: 10 };
obj.a = 20;
obj = { a: 30 };
console.log(obj.a);
```

Question: What is the output and why? Discuss how `const` works with objects, particularly in terms of re-assignment versus mutation.

51. Immediate Invocation with Var and Let

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

Question: What will be printed and why? Analyze the behavior of IIFE (Immediately Invoked Function Expression) with `let` in loops.

52. Function Hoisting vs. Variable Hoisting

```
function test() {
  console.log(foo);
  var foo = 10;
  function foo() {}
  console.log(foo);
}
test();
```

Question: What will be the output and why? Explain the order in which function declarations and variable declarations are hoisted.

53. Scoping Inside Functions

```
var x = 5;
function scopeCheck() {
  console.log(x);
  var x = 10;
  console.log(x);
}
scopeCheck();
```

Question: What will be printed and why? Discuss how variable hoisting affects the `x` variable within the function.

54. Closures with Function Properties

```
function counter() {
  counter.count = (counter.count || 0) + 1;
  return counter.count;
}

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

Question: What will be the output? How does the function property `counter.count` affect the closure and variable persistence?

55. Block Scope and Re-declaration

```
let x = 10;
{
  let x = 20;
  var y = 30;
}
console.log(x);
console.log(y);
```

Question: What will be the output? Explain how `let` and `var` behave differently inside block scope.

56. Hoisting of Function Expression

```
console.log(func());
var func = function() {
  return 10;
};
```

Question: What will be printed and why? Discuss the difference in hoisting behavior between function declarations and function expressions.

57. Const and Re-assignment

```
const a = 5;
{
  const a = 10;
  console.log(a);
}
console.log(a);
```

Question: What will be printed? Discuss how `const` works in different block scopes and whether re-declaration is possible.

58. IIFE (Immediately Invoked Function Expression) and Hoisting

```
(function() {
  console.log(a);
  var a = 10;
})();
```

Question: What will be the output and why? Explain the role of hoisting inside an IIFE.

59. Lexical Scoping and Closure Memory

```
function outer() {
  let counter = 0;
  return function increment() {
    counter++;
    return counter;
  };
}

let increment1 = outer();
let increment2 = outer();
console.log(increment1());
console.log(increment2());
console.log(increment1());
```

Question: What will be printed? Discuss how closures preserve their lexical environment across different instances of the same function.

60. Let and Var in Loops

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

Question: What will be printed and why? Explain the difference between `var` and `let` in loop variable declaration.

61. Shadowing with Let and Var

```
let a = 5;
function test() {
  var a = 10;
  console.log(a);
}
test();
console.log(a);
```

Question: What will be printed and why? Discuss how `let` and `var` interact in terms of scope and shadowing.

62. Function Scope with Var

```
function testScope() {
  console.log(x);
  var x = 10;
  console.log(x);
}
testScope();
```

Question: What will be printed? Analyze how `var` variables are hoisted within a function and how their value is handled before initialization.

63. Re-declaration with Var Inside Function

```
var a = 10;
function testVar() {
  var a = 20;
  console.log(a);
}
testVar();
console.log(a);
```

Question: What will be the output? Explain how variable scope works when re-declaring `var` inside a function.

64. Scoping in Nested Functions

```
function outerFunc() {
  var x = 10;
  function innerFunc() {
    console.log(x);
  }
  innerFunc();
}
outerFunc();
```

Question: What will be printed and why? Discuss the concept of lexical scoping in nested functions.

65. Block Scoping with Let

```
let a = 5;
{
  let a = 10;
  console.log(a);
}
console.log(a);
```

Question: What will be printed? Discuss how block scoping with `let` affects variable visibility inside and outside the block.

66. Closure with Multiple Functions

```
function outerFunc() {
  let count = 0;
  return {
    increment: function() {
      count++;
      return count;
    },
    decrement: function() {
      count--;
      return count;
    }
  };
}

let counter = outerFunc();
console.log(counter.increment());
console.log(counter.decrement());
```

Question: What will be the output and why? Analyze how closures behave with multiple functions accessing the same lexical scope.

67. Hoisting and Function Declaration

```
console.log(test());
function test() {
  return 5;
}
```

Question: What will be printed and why? Explain how function declarations are hoisted.

68. Function Expression vs. Declaration

```
console.log(foo());
var foo = function() {
  return 5;
};
```

Question: What will happen here and why? Discuss the difference between function expressions and function declarations in terms of hoisting.

69. Scope Chain in Nested Functions

```
let a = 1;
function outer

() {
  let b = 2;
  function inner() {
    console.log(a, b);
  }
  inner();
}
outer();
```

Question: What will be printed and why? Explain how the scope chain works in nested functions.

70. Closure in Multiple Instances

```
function outer() {
  let x = 10;
  return function() {
    return x++;
  };
}

const a = outer();
const b = outer();

console.log(a());
console.log(b());
console.log(a());
console.log(b());
```

Question: What will be printed? Discuss how closures work with multiple instances and how they preserve their own state.