1. What is JavaScript?

Answer: JavaScript is a high-level, dynamic programming language commonly used for creating interactive effects within web browsers. It is a core technology alongside HTML and CSS.

2. Explain the difference between `==` and `===`.

Answer: `==` is used for abstract equality comparison (type conversion is done), while `===` is used for strict equality comparison (no type conversion).

```
console.log(5 == '5');  // true
console.log(5 === '5'); // false
```

3. What is a closure in JavaScript?

Answer: A closure is a function that retains access to its lexical scope, even when the function is executed outside that scope.

```
function outer() {
    let count = 0;
    return function() {
        count++;
        return count;
    };
}
const increment = outer();
console.log(increment()); // 1
console.log(increment()); // 2
```

4. Explain `let`, `var`, and `const`.

Answer:

- `var` is function-scoped and can be re-declared and updated.
- `let` is block-scoped and can be updated but not re-declared.
- `const` is block-scoped and cannot be updated or re-declared.

```
var x = 10;
let y = 20;
const z = 30;
```

5. What is hoisting in JavaScript?

Answer: Hoisting is JavaScript's default behavior of moving declarations to the top of the current scope (function or global).

```
console.log(a); // undefined
var a = 5;
```

6. What is the difference between `null` and `undefined`?

Answer:

- `undefined` means a variable has been declared but not assigned a value.
- `null` is an assignment value that represents no value or object.

```
let x;
console.log(x); // undefined
let y = null;
console.log(y); // null
```

7. What are arrow functions?

Answer: Arrow functions are a concise syntax for writing function expressions in ES6. They do not have their own `this` or `arguments`.

```
const add = (a, b) => a + b;
console.log(add(2, 3)); // 5
```

8. What is the event loop in JavaScript?

Answer: The event loop is a mechanism that allows JavaScript to perform non-blocking operations by offloading operations to the browser's APIs, ensuring smooth execution of asynchronous code.

9. Explain `async` and `await`.

Answer: `async` is used to declare an asynchronous function, and `await` is used to pause the execution of the function until the Promise is resolved.

```
async function fetchData() {
    let response = await fetch('https://api.example.com/data');
    let data = await response.json();
    return data;
}
```

10. What is a Promise?

Answer: A Promise is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

```
let promise = new Promise((resolve, reject) => {
    let success = true;
    if (success) {
        resolve('Success!');
    } else {
        reject('Failure!');
    }
});
promise.then(result => console.log(result)).catch(error =>
console.log(error));
```

11. What is the `this` keyword?

Answer: `this` refers to the object it belongs to, and its value depends on how the function is called.

12. Explain the prototype chain.

Answer: The prototype chain is used to build the inheritance hierarchy in JavaScript. Objects inherit properties and methods from their prototype, forming a chain.

13. What is the difference between `call`, `apply`, and `bind`?

Answer:

- `call`: Invokes a function with a given `this` value and arguments provided individually.
- `apply`: Invokes a function with a given `this` value and arguments provided as an array.
- `bind`: Returns a new function with a given `this` value and optional arguments.

```
function greet(greeting, name) {
    console.log(greeting + ', ' + name);
}
greet.call(null, 'Hello', 'Alice');
greet.apply(null, ['Hello', 'Alice']);
const greetAlice = greet.bind(null, 'Hello', 'Alice');
greetAlice();
```

14. What is a higher-order function?

Answer: A higher-order function is a function that takes another function as an argument or returns a function as a result.

```
function multiplier(factor) {
    return function(x) {
        return x * factor;
    };
}
const double = multiplier(2);
console.log(double(5)); // 10
```

15. What is the difference between synchronous and asynchronous code?

Answer: Synchronous code is executed line by line, blocking further execution until the current line completes. Asynchronous code allows other operations to proceed while waiting for the current one to complete.

16. Explain `setTimeout` and `setInterval`.

Answer:

- `setTimeout`: Executes a function after a specified delay.
- `setInterval`: Repeatedly executes a function with a fixed time delay between each call.

```
setTimeout(() => console.log('Executed after 2 seconds'), 2000);
setInterval(() => console.log('Executed every 2 seconds'), 2000);
```

17. What are template literals?

Answer: Template literals are strings enclosed by backticks ` that can contain placeholders for expressions, using `${}`.

```
let name = 'Alice';
console.log(`Hello, ${name}!`); // Hello, Alice!
```

18. What is destructuring in JavaScript?

Answer: Destructuring allows you to unpack values from arrays or properties from objects into distinct variables.

```
let [a, b] = [1, 2];
let {name, age} = {name: 'Alice', age: 25};
```

19. What is the spread operator?

Answer: The spread operator (`...`) expands an array or object into its individual elements.

```
let arr = [1, 2, 3];
let newArr = [...arr, 4, 5];
console.log(newArr); // [1, 2, 3, 4, 5]
```

20. Explain the difference between `for`, `for...in`, and `for...of`.

Answer:

- `for`: Iterates over a sequence of values.
- `for...in`: Iterates over the enumerable properties of an object.
- `for...of`: Iterates over the values of an iterable object.

```
let arr = [10, 20, 30];
for (let i = 0; i < arr.length; i++) console.log(arr[i]);
for (let index in arr) console.log(arr[index]);
for (let value of arr) console.log(value);
```

21. What is `typeof` operator?

Answer: The `typeof` operator returns the data type of a variable.

```
console.log(typeof 42); // "number"
console.log(typeof 'hello'); // "string"
```

22. What is `NaN` in JavaScript?

Answer: `NaN` stands for Not-a-Number and is the result of an invalid or undefined mathematical operation.

```
console.log(0 / 0); // NaN
```

23. Explain event delegation.

Answer: Event delegation is a technique of using a single event listener to manage all events of a particular type by taking advantage of event bubbling.

24. What is an Immediately Invoked Function Expression (IIFE)?

Answer: An IIFE is a function that is executed immediately after its creation. It helps in avoiding polluting the global scope.

```
(function() {
    console.log('IIFE executed');
})();
```

25. What is debouncing in JavaScript?

Answer: Debouncing ensures that a function is not called too frequently. It limits the rate at which a function can be executed.

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

26. What is throttling in JavaScript?

Answer: Throttling ensures a function is called at most once in a specified period.

```
function throttle(func, limit) {
    let inThrottle;
    return function(...args) {
        if (!inThrottle) {
            func.apply(this, args);
            inThrottle = true;
            setTimeout(() => inThrottle = false, limit);
        }
    };
}
```

27. What is `JSON.stringify` and `JSON.parse`?

Answer:

- `JSON.stringify`: Converts a JavaScript object into a JSON string.
- `JSON.parse`: Parses a JSON string and converts it into a JavaScript object.

```
let obj = {name: 'Alice', age: 25};
let jsonString = JSON.stringify(obj);
let newObj = JSON.parse(jsonString);
```

28. Explain the concept of `strict mode` in JavaScript.

Answer: Strict mode is a way to enforce stricter parsing and error handling in JavaScript, helping to catch common coding mistakes and unsafe actions.

```
'use strict';
```

29. What is the difference between `slice` and `splice`?

Answer:

- `slice`: Returns a shallow copy of a portion of an array into a new array.
- `splice`: Changes the contents of an array by removing or replacing existing elements and/or adding new elements.

```
let arr = [1, 2, 3, 4];
console.log(arr.slice(1, 3)); // [2, 3]
arr.splice(1, 2, 10, 20);
console.log(arr); // [1, 10, 20, 4]
```

30. What is the difference between `map`, `filter`, and `reduce`?

Answer:

- `map`: Creates a new array with the results of calling a function on every element.
- `filter`: Creates a new array with all elements that pass the test.
- `reduce`: Applies a function against an accumulator and each element to reduce it to a single value.

```
let arr = [1, 2, 3, 4];
let mapResult = arr.map(x => x * 2);       // [2, 4, 6, 8]
let filterResult = arr.filter(x => x > 2); // [3, 4]
let reduceResult = arr.reduce((sum, x) => sum + x, 0); // 10
```

31. What is a callback function?

Answer: A callback is a function passed into another function as an argument, which is then invoked inside the outer function to complete some action.

32. What are promises?

Answer: A promise represents the eventual completion (or failure) of an asynchronous operation. Promises allow chaining operations using `then`, `catch`, and `finally`.

33. What is the difference between `undefined` and `not defined`?

Answer:

- `undefined`: A variable that has been declared but has not been assigned a value.
- `not defined`: A variable that has not been declared.

34. Explain the concept of scope in JavaScript.

Answer: Scope determines the accessibility of variables and functions in different parts of the code. JavaScript has function scope, block scope, and global scope.

35. What is an object in JavaScript?

Answer: An object is a collection of properties, where each property is defined by a key-value pair.

36. What is a prototype in JavaScript?

Answer: Every JavaScript object has a prototype, an object from which it inherits properties and methods.

37. What is event bubbling?

Answer: Event bubbling is the process where an event starts from the target element and bubbles up to the root element, triggering handlers along the way.

38. What is the `fetch` API?

Answer: The `fetch` API is a modern interface for making HTTP requests in JavaScript, replacing older technologies like `XMLHttpRequest`.

39. Explain the concept of a module in JavaScript.

Answer: A module is a piece of code that is independent and reusable, usually exporting specific parts of the code (variables, functions, classes) to be imported into other modules.

40. What is an arrow function?

Answer: Arrow functions are a shorthand syntax for writing functions in ES6. They are lexically bound, meaning they inherit the `this` context from their parent scope.

41. Explain the concept of promises in JavaScript.

Answer: Promises represent a value that may be available now, in the future, or never. They allow for asynchronous code to be handled more cleanly than traditional callback methods.

42. What is the difference between `let` and `var`?

Answer: `let` is block-scoped and cannot be re-declared in the same scope, while `var` is function-scoped and can be re-declared.

43. What is the difference between `==` and `===` in JavaScript?

Answer: `==` compares values for equality, converting the data types if needed, whereas `===` checks for both value and type equality without conversion.

44. What is the difference between synchronous and asynchronous code in JavaScript?

Answer: Synchronous code is executed sequentially, blocking further execution until the current task completes, while asynchronous code can execute multiple tasks concurrently, allowing other operations to run before completion.

45. What is the purpose of `use strict` in JavaScript?

Answer: `use strict` enforces stricter parsing and error handling in your JavaScript code, helping to avoid common mistakes and bugs.

46. What is an event loop in JavaScript?

Answer: The event loop is a mechanism that allows JavaScript to handle asynchronous events. It processes tasks from the callback queue and executes them one by one, ensuring that the main thread remains non-blocking.

47. What is a higher-order function?

Answer: A higher-order function is a function that takes another function as an argument, returns a function, or both. It allows for more abstract and flexible code.

48. What is `call`, `apply`, and `bind` in JavaScript?

Answer:

- `call`: Invokes a function with a given `this` value and arguments provided individually.
- `apply`: Invokes a function with a given `this` value and arguments provided as an array.
- `bind`: Returns a new function with a given `this` value and optional arguments.

49. What is a polyfill in JavaScript?

Answer: A polyfill is a piece of code used to provide modern functionality on older browsers that do not natively support it.

50. What is event delegation in JavaScript?

Answer: Event delegation allows you to use a single event handler to manage events for multiple child elements by leveraging event bubbling.