**1. What are the different data types in JavaScript?**

* **Primitive types:** string, number, boolean, null, undefined, bigint, symbol
* **Non-primitive types:** object, array, function

**2. What is the difference between null and undefined?**

* **null**: A value that represents an intentional absence of any object value.
* **undefined**: A variable that has been declared but not assigned a value.

**3. What is the difference between let, const, and var?**

* **var**: Function-scoped, can be redeclared and reassigned.
* **let**: Block-scoped, can be reassigned but not redeclared.
* **const**: Block-scoped, cannot be reassigned or redeclared.

**4. What is the difference between == and ===?**

* **==**: Checks only value (type coercion happens).
* **===**: Checks both value and type.

js

CopyEdit

console.log(5 == "5"); // true

console.log(5 === "5"); // false

**5. What is closure in JavaScript?**

* A closure is a function that remembers the scope in which it was created, even if it is executed outside of that scope.

js

CopyEdit

function outer() {

let count = 0;

return function inner() {

count++;

console.log(count);

};

}

const counter = outer();

counter(); // 1

counter(); // 2

**6. What are the different ways to create objects in JavaScript?**

* Object literal: { key: value }
* Constructor function: function Person() {}
* Class: class Person {}
* Object.create()

**7. What is the difference between shallow copy and deep copy?**

* **Shallow copy**: Copies reference, changes affect original.
* **Deep copy**: Creates a completely independent copy.

js

CopyEdit

const obj = { a: 1, b: { c: 2 } };

const shallowCopy = { ...obj }; // Shallow copy

const deepCopy = JSON.parse(JSON.stringify(obj)); // Deep copy

**8. Explain event bubbling and capturing.**

* **Bubbling:** Event propagates from child to parent.
* **Capturing:** Event propagates from parent to child.

js

CopyEdit

document.getElementById("child").addEventListener("click", (e) => {

console.log("Child clicked");

}, false); // Bubbling

document.getElementById("parent").addEventListener("click", (e) => {

console.log("Parent clicked");

}, true); // Capturing

**9. What is the difference between map(), forEach(), filter(), and reduce()?**

* **map()**: Returns a new array with modified values.
* **forEach()**: Iterates over an array but doesn't return a new array.
* **filter()**: Returns a new array with elements that satisfy a condition.
* **reduce()**: Reduces array to a single value.

**10. What is the difference between synchronous and asynchronous programming?**

* **Synchronous**: Code executes line by line.
* **Asynchronous**: Code executes without blocking the main thread.

js

CopyEdit

console.log("Start");

setTimeout(() => console.log("Async Code"), 1000);

console.log("End");

**11. What is the difference between setTimeout and setInterval?**

* **setTimeout**: Executes code once after a delay.
* **setInterval**: Executes code repeatedly at a specified interval.

js

CopyEdit

setTimeout(() => console.log("Runs once"), 1000);

setInterval(() => console.log("Runs repeatedly"), 1000);

**12. What is a promise in JavaScript?**

* A promise represents a value that may be available now, or in the future, or never.

js

CopyEdit

let promise = new Promise((resolve, reject) => {

setTimeout(() => resolve("Success"), 2000);

});

promise.then((result) => console.log(result));

**13. Explain async/await.**

* **async**: Declares an asynchronous function.
* **await**: Waits for a promise to resolve.

js

CopyEdit

async function fetchData() {

let response = await fetch("https://jsonplaceholder.typicode.com/todos/1");

let data = await response.json();

console.log(data);

}

fetchData();

**14. Explain call, apply, and bind.**

* **call**: Calls a function with a specific this value and arguments.

js

CopyEdit

function greet(message) {

console.log(`${message}, ${this.name}`);

}

const person = { name: "John" };

greet.call(person, "Hello");

* **apply**: Similar to call, but arguments are passed as an array.
* **bind**: Returns a new function with this bound.

**15. What is the difference between function declaration and function expression?**

* **Function declaration:** Named function that is hoisted.

js

CopyEdit

function sayHello() {

console.log("Hello");

}

* **Function expression:** Anonymous function assigned to a variable.

js

CopyEdit

const sayHello = function () {

console.log("Hello");

};

**16. What is the this keyword?**

* The value of this depends on how a function is called.
* Inside an arrow function, this refers to its lexical scope.

js

CopyEdit

const obj = {

name: "John",

greet: function () {

console.log(this.name);

},

};

obj.greet();

**17. What are template literals in JavaScript?**

* Template literals use backticks (`) and allow for variable interpolation.

js

CopyEdit

let name = "John";

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

**18. What are higher-order functions?**

* Functions that take another function as an argument or return a function.

js

CopyEdit

function higherOrderFunction(callback) {

return callback();

}

higherOrderFunction(() => console.log("Hello"));

**19. What is destructuring in JavaScript?**

* Extract values from arrays or objects easily.

js

CopyEdit

const person = { name: "John", age: 25 };

const { name, age } = person;

console.log(name, age);

**20. What is the difference between spread and rest operators?**

* **Spread (...)**: Expands an array or object.
* **Rest (...)**: Gathers arguments into an array.

js

CopyEdit

const arr = [1, 2, 3];

console.log(...arr); // Spread

function sum(...numbers) {

return numbers.reduce((acc, num) => acc + num, 0);

}

console.log(sum(1, 2, 3)); // Rest

These are just **20** key JavaScript questions. Let me know if you want the full list of **50**! 🚀