**1. What is a closure in JavaScript?**

A closure is a function that remembers the variables from the place where it was defined, even after it’s executed elsewhere. This allows the function to access and manipulate those variables later.

***Example:***

*function outer() {*

*let count = 0;*

*return function() { count++; return count; }*

*}*

*const counter = outer();*

*console.log(counter()); // 1*

*console.log(counter()); // 2*

**2. What is the difference between == and === in JavaScript?**

== checks for equality after type conversion, while === checks for both value and type equality without converting types. Using === is preferred to avoid unexpected results.

**Example:**

*console.log(5 == '5'); // true*

*console.log(5 === '5'); // false*

**3. How do you create an object in JavaScript?**

You can create an object using curly braces {} or the new Object() constructor. The first method is more common and easier to read.

**Example:**

let person = { name: 'John', age: 30 };

let anotherPerson = new Object();

anotherPerson.name = 'Jane';

**4. What is an arrow function in JavaScript?**

An arrow function is a shorter syntax for writing functions. It does not have its own this context, which makes it useful for maintaining the this value in callbacks.

**Example:**

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

console.log(add(2, 3)); // 5

**5. How can you copy an object in JavaScript?**

You can copy an object using Object.assign() or the spread operator .... These methods create a shallow copy, meaning nested objects are still referenced.

**Example:**

javascript

let original = { name: 'John', age: 30 };

let copy = { ...original };

console.log(copy); // { name: 'John', age: 30 }

**6. What is NaN in JavaScript?**

NaN stands for "Not-a-Number" and represents a value that is not a legal number. It often occurs when mathematical operations fail to produce a valid result.  
**Example:**

javascript

let result = 'abc' / 2;

console.log(result); // NaN

**7. What is event bubbling in JavaScript?**

Event bubbling is a process where an event starts from the deepest target element and propagates up to the parent elements. This allows events to be handled at multiple levels in the DOM.  
**Example:**

javascript

document.body.addEventListener('click', () => console.log('Body clicked'));

document.querySelector('button').addEventListener('click', () => console.log('Button clicked'));

**8. How do you check if a variable is an array in JavaScript?**

You can check if a variable is an array using Array.isArray(). This method returns true if the variable is an array and false otherwise.  
**Example:**

javascript

let items = [1, 2, 3];

console.log(Array.isArray(items)); // true

console.log(Array.isArray('hello')); // false

**9. What is undefined in JavaScript?**

undefined is a primitive value that represents a variable that has been declared but not yet assigned a value. It is also the default return value of functions with no return statement.  
**Example:**

javascript

let x;

console.log(x); // undefined

**10. How do you declare a variable in JavaScript?**

You can declare a variable in JavaScript using var, let, or const. let and const are block-scoped, while var is function-scoped. Use const for constants and let for variables that may change.  
**Example:**

javascript

let name = 'John';

const age = 30;

These questions and answers should be helpful for your JavaScript interview preparation!

4o

More 20 IQs

ChatGPT

Here are 20 more JavaScript interview questions with answers in the format you prefer:

**11. What is the this keyword in JavaScript?**

this refers to the object from which the current code is being executed. Its value depends on how the function is called and can change in different contexts.  
**Example:**

javascript

let person = {

name: 'John',

greet: function() { console.log(this.name); }

};

person.greet(); // John

**12. What is hoisting in JavaScript?**

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution. This allows variables and functions to be used before they are declared.  
**Example:**

javascript

console.log(x); // undefined

var x = 5;

**13. How do you define a default parameter in JavaScript?**

You can define a default parameter by assigning a value to the parameter in the function definition. If no argument is passed, the default value will be used.  
**Example:**

javascript

function greet(name = 'Guest') {

console.log('Hello, ' + name);

}

greet(); // Hello, Guest

**14. What is the difference between let and var?**

let is block-scoped, meaning it is only accessible within the block it is defined. var is function-scoped and can be accessed outside the block it is declared in.  
**Example:**

javascript

if (true) {

let x = 5;

var y = 10;

}

console.log(y); // 10

console.log(x); // Error: x is not defined

**15. What is an IIFE in JavaScript?**

An IIFE (Immediately Invoked Function Expression) is a function that is executed immediately after it is defined. It is often used to create a private scope.  
**Example:**

javascript

(function() {

let message = 'Hello';

console.log(message);

})(); // Hello

**16. How do you convert a string to a number in JavaScript?**

You can convert a string to a number using the Number() function, the unary plus operator +, or parseInt()/parseFloat(). These methods parse the string and return a number.  
**Example:**

javascript

let str = '123';

let num = Number(str);

console.log(num); // 123

**17. What is the purpose of the map() method in JavaScript?**

The map() method creates a new array by applying a function to each element of an existing array. It does not modify the original array but returns a transformed array.  
**Example:**

javascript

let numbers = [1, 2, 3];

let doubled = numbers.map(n => n \* 2);

console.log(doubled); // [2, 4, 6]

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

undefined means a variable has been declared but not assigned a value, while null is an assignment value that represents "no value" or "nothing."  
**Example:**

javascript

let x;

let y = null;

console.log(x); // undefined

console.log(y); // null

**19. How do you handle asynchronous code in JavaScript?**

Asynchronous code in JavaScript can be handled using callbacks, Promises, or async/await. Promises and async/await provide more readable and maintainable code compared to callbacks.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

console.log(data);

}

fetchData();

**20. What is the purpose of the reduce() method in JavaScript?**

The reduce() method applies a function to each element of an array to reduce the array to a single value. It accumulates the result from each iteration of the function.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let sum = numbers.reduce((total, num) => total + num, 0);

console.log(sum); // 10

**21. What is the difference between forEach() and map()?**

forEach() iterates over an array and executes a function for each element without returning anything. map() also iterates but returns a new array with the results of applying the function.  
**Example:**

javascript

let numbers = [1, 2, 3];

numbers.forEach(num => console.log(num \* 2)); // Logs 2, 4, 6

let doubled = numbers.map(num => num \* 2);

console.log(doubled); // [2, 4, 6]

**22. How do you clone an array in JavaScript?**

You can clone an array using the slice() method, the spread operator ..., or Array.from(). These methods create a shallow copy of the array.  
**Example:**

javascript

let original = [1, 2, 3];

let copy = [...original];

console.log(copy); // [1, 2, 3]

**23. What is the purpose of the filter() method in JavaScript?**

The filter() method creates a new array with elements that pass a specified test. It does not modify the original array but returns a new array with the filtered elements.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let evenNumbers = numbers.filter(n => n % 2 === 0);

console.log(evenNumbers); // [2, 4]

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

Template literals are string literals enclosed in backticks (```) that allow embedding expressions using ${expression}. They also support multi-line strings without the need for concatenation.  
**Example:**

javascript

let name = 'John';

let message = `Hello, ${name}!`;

console.log(message); // Hello, John!

**25. What is the typeof operator in JavaScript?**

The typeof operator returns a string indicating the data type of a variable or expression. It can be used to check whether a variable is a number, string, object, etc.  
**Example:**

javascript

let x = 42;

console.log(typeof x); // number

**26. How do you merge two arrays in JavaScript?**

You can merge two arrays using the concat() method or the spread operator .... These methods return a new array with the combined elements.  
**Example:**

javascript

let arr1 = [1, 2];

let arr2 = [3, 4];

let merged = [...arr1, ...arr2];

console.log(merged); // [1, 2, 3, 4]

**27. What is the difference between call() and apply()?**

Both call() and apply() invoke a function with a specific this value, but call() accepts arguments individually, while apply() accepts them as an array.  
**Example:**

javascript

function greet(greeting, name) {

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

}

greet.call(null, 'Hello', 'John'); // Hello, John!

greet.apply(null, ['Hello', 'John']); // Hello, John!

**28. What is the purpose of Promise.all() in JavaScript?**

Promise.all() takes an array of Promises and returns a single Promise that resolves when all the input Promises have resolved. It is useful for running multiple asynchronous operations in parallel.  
**Example:**

javascript

Promise.all([fetch('/data1'), fetch('/data2')])

.then(responses => console.log('All data fetched'))

.catch(error => console.log('An error occurred', error));

**29. How do you handle errors in JavaScript?**

Errors in JavaScript can be handled using try...catch blocks. The try block contains code that might throw an error, and the catch block contains code to handle the error.  
**Example:**

javascript

try {

let result = riskyOperation();

} catch (error) {

console.log('An error occurred:', error.message);

}

**30. What is the difference between setTimeout() and setInterval()?**

setTimeout() executes a function once after a specified delay, while setInterval() repeatedly executes a function at a specified interval. Both are used for timing operations.  
**Example:**

javascript

setTimeout(() => console.log('Hello after 1 second'), 1000);

setInterval(() => console.log('Repeating every 2 seconds'), 2000);

These questions should provide a solid foundation for your JavaScript interview preparation.

4o

More 20 IQs

ChatGPT

Here are 20 more JavaScript interview questions with answers:

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

Synchronous code is executed sequentially, blocking the next operation until the current one is completed. Asynchronous code allows multiple operations to run concurrently, not waiting for others to finish.  
**Example:**

javascript

console.log('Start');

setTimeout(() => console.log('Async operation'), 1000);

console.log('End');

// Output: Start, End, Async operation

**32. What are JavaScript promises?**

A promise is an object representing the eventual completion or failure of an asynchronous operation. It has three states: pending, fulfilled, and rejected.  
**Example:**

javascript

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

setTimeout(() => resolve('Done'), 1000);

});

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

**33. What is the purpose of the bind() method in JavaScript?**

The bind() method creates a new function that, when called, has its this keyword set to the specified value. It is useful for setting this in callback functions.  
**Example:**

javascript

let person = { name: 'John' };

let greet = function() { console.log('Hello, ' + this.name); };

let greetPerson = greet.bind(person);

greetPerson(); // Hello, John

**34. What is the difference between slice() and splice() in JavaScript?**

slice() returns a new array with selected elements from the original array, without modifying it. splice() modifies the original array by adding or removing elements.  
**Example:**

javascript

let arr = [1, 2, 3, 4];

let sliced = arr.slice(1, 3); // [2, 3]

let spliced = arr.splice(1, 2); // arr becomes [1, 4]

**35. What is the event loop in JavaScript?**

The event loop is a mechanism in JavaScript that handles asynchronous callbacks by processing events and executing them when the call stack is empty. It enables non-blocking behavior in JavaScript.  
**Example:**

javascript

console.log('Start');

setTimeout(() => console.log('Delayed'), 0);

console.log('End');

// Output: Start, End, Delayed

**36. How do you prevent event bubbling in JavaScript?**

You can prevent event bubbling by using the stopPropagation() method on the event object. This stops the event from propagating to parent elements.  
**Example:**

javascript

document.querySelector('button').addEventListener('click', function(event) {

event.stopPropagation();

console.log('Button clicked');

});

document.body.addEventListener('click', () => console.log('Body clicked'));

// Only "Button clicked" will be logged when the button is clicked.

**37. What is the purpose of the find() method in JavaScript?**

The find() method returns the first element in an array that satisfies a given condition. If no element satisfies the condition, it returns undefined.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let result = numbers.find(n => n > 2);

console.log(result); // 3

**38. How do you check if an object has a certain property in JavaScript?**

You can check if an object has a certain property using the hasOwnProperty() method or the in operator. Both methods return true if the property exists.  
**Example:**

javascript

let person = { name: 'John', age: 30 };

console.log(person.hasOwnProperty('name')); // true

console.log('age' in person); // true

**39. What is the difference between Object.freeze() and Object.seal()?**

Object.freeze() prevents any changes to an object, making it immutable. Object.seal() allows modifying existing properties but prevents adding or removing properties.  
**Example:**

javascript

let obj = { name: 'John' };

Object.freeze(obj);

obj.name = 'Jane'; // No effect

Object.seal(obj);

obj.age = 30; // Cannot add new property

**40. What is debouncing in JavaScript?**

Debouncing is a technique to limit the rate at which a function is executed. It ensures the function runs only once after a specified delay, even if the event is triggered multiple times.  
**Example:**

javascript

function debounce(func, delay) {

let timeout;

return function(...args) {

clearTimeout(timeout);

timeout = setTimeout(() => func.apply(this, args), delay);

};

}

**41. What is throttling in JavaScript?**

Throttling is a technique to ensure that a function is executed no more than once within a specified time interval, regardless of how often the event is triggered.  
**Example:**

javascript

function throttle(func, limit) {

let inThrottle;

return function(...args) {

if (!inThrottle) {

func.apply(this, args);

inThrottle = true;

setTimeout(() => inThrottle = false, limit);

}

};

}

**42. How do you check if a number is an integer in JavaScript?**

You can check if a number is an integer using the Number.isInteger() method. It returns true if the number is an integer and false otherwise.  
**Example:**

javascript

let num1 = 5;

let num2 = 5.5;

console.log(Number.isInteger(num1)); // true

console.log(Number.isInteger(num2)); // false

**43. What is the async function in JavaScript?**

An async function is a function that returns a Promise and allows the use of await inside it. It simplifies working with Promises and asynchronous code.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

console.log(data);

}

fetchData();

**44. What is the purpose of Object.assign() in JavaScript?**

Object.assign() copies the values of all enumerable properties from one or more source objects to a target object. It returns the modified target object.  
**Example:**

javascript

let target = { a: 1 };

let source = { b: 2, c: 3 };

Object.assign(target, source);

console.log(target); // { a: 1, b: 2, c: 3 }

**45. How do you iterate over an object’s properties in JavaScript?**

You can iterate over an object’s properties using a for...in loop or Object.keys(), Object.values(), or Object.entries() methods, depending on the desired output.  
**Example:**

javascript

let person = { name: 'John', age: 30 };

for (let key in person) {

console.log(key + ': ' + person[key]);

}

// Output: name: John, age: 30

**46. What is the difference between == and === in JavaScript?**

== checks for equality after type conversion, while === checks for both value and type equality without converting types. Using === avoids unexpected results.  
**Example:**

javascript

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

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

**47. What is the difference between push() and unshift() methods in JavaScript?**

push() adds elements to the end of an array, while unshift() adds elements to the beginning of an array. Both methods modify the original array.  
**Example:**

javascript

let arr = [1, 2, 3];

arr.push(4); // arr becomes [1, 2, 3, 4]

arr.unshift(0); // arr becomes [0, 1, 2, 3, 4]

**48. How do you remove duplicate items from an array in JavaScript?**

You can remove duplicate items from an array using Set or by filtering the array with indexOf() to create a new array with unique elements.  
**Example:**

javascript

let numbers = [1, 2, 2, 3, 3, 4];

let uniqueNumbers = [...new Set(numbers)];

console.log(uniqueNumbers); // [1, 2, 3, 4]

**49. What is currying in JavaScript?**

Currying is a technique where a function is transformed into a sequence of functions, each taking a single argument. It allows partial application of arguments.  
**Example:**

javascript

function add(a) {

return function(b) {

return a + b;

};

}

let add5 = add(5);

console.log(add5(3)); // 8

**50. What is the purpose of Object.keys() in JavaScript?**

Object.keys() returns an array of a given object’s own enumerable property names. It is useful for iterating over an object's properties.  
**Example:**

javascript

let person = { name: 'John', age: 30 };

let keys = Object.keys(person);

console.log(keys); // ['name', 'age']

These questions should help you further strengthen your JavaScript interview preparation!

4o

More 30 Iqs

ChatGPT

Here are 30 more JavaScript interview questions with answers:

**51. What is hoisting in JavaScript?**

Hoisting is JavaScript's behavior of moving variable and function declarations to the top of their containing scope during the compilation phase. It allows using functions and variables before they are declared.  
**Example:**

javascript

console.log(hoistedVar); // undefined

var hoistedVar = 'Hello';

**52. What is the difference between null and undefined in JavaScript?**

null is an assigned value representing "no value" or "nothing," while undefined means a variable has been declared but not assigned a value.  
**Example:**

javascript

let a = null;

let b;

console.log(a); // null

console.log(b); // undefined

**53. What is a closure in JavaScript?**

A closure is a function that retains access to its outer scope, even after the outer function has returned. This allows the inner function to access variables from the outer function.  
**Example:**

javascript

function outer() {

let outerVar = 'I am outside!';

return function inner() {

console.log(outerVar);

};

}

let closureFunc = outer();

closureFunc(); // I am outside!

**54. How do you create a class in JavaScript?**

You can create a class using the class keyword, defining properties in the constructor, and methods directly within the class.  
**Example:**

javascript

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log('Hello, ' + this.name);

}

}

let person = new Person('John');

person.greet(); // Hello, John

**55. What is the purpose of JSON.stringify() in JavaScript?**

JSON.stringify() converts a JavaScript object or array into a JSON string, making it easier to send data over the network or store it as a string.  
**Example:**

javascript

let obj = { name: 'John', age: 30 };

let jsonString = JSON.stringify(obj);

console.log(jsonString); // {"name":"John","age":30}

**56. What is the purpose of JSON.parse() in JavaScript?**

JSON.parse() converts a JSON string back into a JavaScript object, allowing you to work with the data as an object or array.  
**Example:**

javascript

let jsonString = '{"name":"John","age":30}';

let obj = JSON.parse(jsonString);

console.log(obj.name); // John

**57. What is the difference between forEach() and map() in JavaScript?**

forEach() executes a provided function once for each array element but does not return anything. map() also executes a function for each element but returns a new array with the results.  
**Example:**

javascript

let arr = [1, 2, 3];

arr.forEach(num => console.log(num \* 2)); // 2, 4, 6

let newArr = arr.map(num => num \* 2);

console.log(newArr); // [2, 4, 6]

**58. How do you create a Promise in JavaScript?**

You create a Promise by using the Promise constructor, passing in a function that takes resolve and reject as arguments. The promise is resolved or rejected based on conditions.  
**Example:**

javascript

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

let success = true;

if (success) {

resolve('Operation successful');

} else {

reject('Operation failed');

}

});

promise.then(result => console.log(result)); // Operation successful

**59. What is the purpose of Array.filter() in JavaScript?**

Array.filter() creates a new array containing all elements of the original array that pass a specified test (provided as a function).  
**Example:**

javascript

let numbers = [1, 2, 3, 4, 5];

let evenNumbers = numbers.filter(num => num % 2 === 0);

console.log(evenNumbers); // [2, 4]

**60. How do you create a shallow copy of an object in JavaScript?**

You can create a shallow copy of an object using Object.assign() or the spread operator .... A shallow copy only copies the object's first level of properties.  
**Example:**

javascript

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

let shallowCopy = { ...obj };

console.log(shallowCopy); // { a: 1, b: { c: 2 } }

**61. What is the difference between slice() and substring() in JavaScript?**

slice() extracts a section of a string or array and returns it as a new string or array. substring() returns the characters between two indices without accepting negative values.  
**Example:**

javascript

let str = 'Hello, World!';

console.log(str.slice(0, 5)); // Hello

console.log(str.substring(0, 5)); // Hello

**62. What is the purpose of the reduce() method in JavaScript?**

reduce() applies a function to each element of an array to reduce the array to a single value, such as a sum or a product.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let sum = numbers.reduce((accumulator, currentValue) => accumulator + currentValue, 0);

console.log(sum); // 10

**63. What are JavaScript generators?**

Generators are special functions that can pause and resume their execution, allowing you to generate values on the fly. They are defined using the function\* syntax.  
**Example:**

javascript

function\* generateNumbers() {

yield 1;

yield 2;

yield 3;

}

let gen = generateNumbers();

console.log(gen.next().value); // 1

**64. What is the this keyword in JavaScript?**

The this keyword refers to the object that is executing the current function. Its value depends on how the function is called (e.g., in an object method, it refers to the object).  
**Example:**

javascript

let person = {

name: 'John',

greet: function() {

console.log('Hello, ' + this.name);

}

};

person.greet(); // Hello, John

**65. What is NaN in JavaScript?**

NaN stands for "Not-a-Number" and is a special value that results from an operation that cannot produce a valid number, such as dividing a string by a number.  
**Example:**

javascript

let result = 'hello' / 2;

console.log(result); // NaN

**66. How do you check if a variable is an array in JavaScript?**

You can check if a variable is an array using the Array.isArray() method, which returns true if the variable is an array and false otherwise.  
**Example:**

javascript

let arr = [1, 2, 3];

console.log(Array.isArray(arr)); // true

console.log(Array.isArray('hello')); // false

**67. What is the difference between apply() and call() in JavaScript?**

Both apply() and call() are methods used to call a function with a specified this value. call() takes arguments separately, while apply() takes arguments as an array.  
**Example:**

javascript

function greet(greeting, punctuation) {

console.log(greeting + ', ' + this.name + punctuation);

}

let person = { name: 'John' };

greet.call(person, 'Hello', '!'); // Hello, John!

greet.apply(person, ['Hello', '!']); // Hello, John!

**68. What is a pure function in JavaScript?**

A pure function is a function that always produces the same output for the same input and has no side effects (does not modify external state).  
**Example:**

javascript

function add(a, b) {

return a + b;

}

console.log(add(2, 3)); // 5

**69. How do you remove the last element from an array in JavaScript?**

You can remove the last element from an array using the pop() method, which modifies the original array and returns the removed element.  
**Example:**

javascript

let arr = [1, 2, 3];

let lastElement = arr.pop();

console.log(arr); // [1, 2]

console.log(lastElement); // 3

**70. What is the typeof operator in JavaScript?**

The typeof operator is used to determine the type of a variable or expression, returning a string that indicates the type (e.g., "string," "number," "object").  
**Example:**

javascript

let num = 5;

console.log(typeof num); // number

console.log(typeof 'hello'); // string

**71. What is the difference between == and === in JavaScript?**

== compares values for equality with type coercion, meaning it converts types if necessary. === compares values for strict equality without converting types.  
**Example:**

javascript

console.log(5

4o

ChatGPT

4o mini

continue please

ChatGPT

Sure! Here are 30 more JavaScript interview questions with answers:

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

An IIFE is a function that runs as soon as it is defined. It helps create a new scope to avoid polluting the global scope.  
**Example:**

javascript

(function() {

let x = 'I am private';

console.log(x);

})();

console.log(typeof x); // undefined

**82. What is event delegation in JavaScript?**

Event delegation is a technique where you attach a single event listener to a parent element instead of multiple listeners on individual child elements. It leverages event bubbling to handle events at a higher level.  
**Example:**

javascript

document.getElementById('parent').addEventListener('click', function(event) {

if (event.target && event.target.matches('button.child')) {

console.log('Child button clicked!');

}

});

**83. What is debouncing in JavaScript?**

Debouncing is a technique used to limit the rate at which a function is executed. It ensures that the function is only called after a specified delay when no additional events occur.  
**Example:**

javascript

function debounce(func, delay) {

let timeout;

return function(...args) {

clearTimeout(timeout);

timeout = setTimeout(() => func.apply(this, args), delay);

};

}

window.addEventListener('resize', debounce(() => console.log('Resized!'), 500));

**84. What is throttling in JavaScript?**

Throttling is a technique to limit the rate at which a function is executed by ensuring it runs only once within a specified time interval.  
**Example:**

javascript

function throttle(func, limit) {

let lastCall = 0;

return function(...args) {

const now = Date.now();

if (now - lastCall >= limit) {

lastCall = now;

func.apply(this, args);

}

};

}

window.addEventListener('scroll', throttle(() => console.log('Scrolled!'), 1000));

**85. What is the bind() method in JavaScript?**

The bind() method creates a new function that, when called, has its this keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is invoked.  
**Example:**

javascript

function greet(greeting) {

console.log(greeting + ', ' + this.name);

}

let person = { name: 'John' };

let boundGreet = greet.bind(person, 'Hello');

boundGreet(); // Hello, John

**86. What is async/await in JavaScript?**

async/await is a syntax for working with asynchronous code. async functions return a promise, and await pauses the function execution until the promise is resolved.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

console.log(data);

}

fetchData();

**87. What is a Set in JavaScript?**

A Set is a collection of unique values where each value can only occur once. It is useful for storing non-duplicate values.  
**Example:**

javascript

let mySet = new Set([1, 2, 3, 4, 2]);

console.log(mySet); // Set {1, 2, 3, 4}

**88. What is a WeakSet in JavaScript?**

A WeakSet is similar to a Set but only accepts objects as values and does not prevent garbage collection of those objects.  
**Example:**

javascript

let weakSet = new WeakSet();

let obj = {};

weakSet.add(obj);

console.log(weakSet.has(obj)); // true

**89. How do you use Array.prototype.reduce() in JavaScript?**

The reduce() method executes a reducer function (that you provide) on each element of the array, resulting in a single output value.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let product = numbers.reduce((acc, val) => acc \* val, 1);

console.log(product); // 24

**90. What is Event Loop in JavaScript?**

The event loop is a mechanism that handles asynchronous operations by executing code from the call stack and then handling queued events or callbacks.  
**Example:**

javascript

console.log('Start');

setTimeout(() => console.log('Middle'), 0);

console.log('End');

// Output: Start, End, Middle

**91. What is a callback function in JavaScript?**

A callback function is a function passed as an argument to another function, to be executed after the completion of some operation.  
**Example:**

javascript

function fetchData(callback) {

setTimeout(() => callback('Data fetched'), 1000);

}

fetchData(data => console.log(data)); // Data fetched

**92. What is Promise.all() in JavaScript?**

Promise.all() is a method that takes an iterable of promises and returns a single promise that resolves when all of the promises in the iterable have resolved.  
**Example:**

javascript

let promise1 = Promise.resolve(1);

let promise2 = Promise.resolve(2);

Promise.all([promise1, promise2]).then(values => console.log(values)); // [1, 2]

**93. What is Promise.race() in JavaScript?**

Promise.race() returns a promise that resolves or rejects as soon as one of the promises in the iterable resolves or rejects, with the value or reason from that promise.  
**Example:**

javascript

let promise1 = new Promise((resolve, reject) => setTimeout(resolve, 500, 'one'));

let promise2 = new Promise((resolve, reject) => setTimeout(resolve, 100, 'two'));

Promise.race([promise1, promise2]).then(value => console.log(value)); // two

**94. What is Promise.allSettled() in JavaScript?**

Promise.allSettled() returns a promise that resolves after all of the given promises have either resolved or rejected, with an array of objects that each describe the outcome.  
**Example:**

javascript

let promise1 = Promise.resolve(1);

let promise2 = Promise.reject('Error');

Promise.allSettled([promise1, promise2]).then(results => console.log(results));

// [{status: 'fulfilled', value: 1}, {status: 'rejected', reason: 'Error'}]

**95. What is Function.prototype.apply() in JavaScript?**

apply() calls a function with a given this value and arguments provided as an array (or an array-like object).  
**Example:**

javascript

function sum(a, b) {

return a + b;

}

console.log(sum.apply(null, [1, 2])); // 3

**96. What is the Object.create() method in JavaScript?**

Object.create() creates a new object with the specified prototype object and properties. It allows for inheritance and prototype-based object creation.  
**Example:**

javascript

let person = { name: 'John' };

let employee = Object.create(person);

employee.job = 'Developer';

console.log(employee.name); // John

**97. What is Array.prototype.find() in JavaScript?**

Array.prototype.find() returns the first element in the array that satisfies the provided testing function. If no elements satisfy the testing function, it returns undefined.  
**Example:**

javascript

let numbers = [4, 9, 16];

let found = numbers.find(num => num > 10);

console.log(found); // 16

**98. What is Array.prototype.some() in JavaScript?**

Array.prototype.some() tests whether at least one element in the array passes the provided function's test. It returns true if it finds such an element, otherwise false.  
**Example:**

javascript

let numbers = [1, 2, 3];

let hasNegative = numbers.some(num => num < 0);

console.log(hasNegative); // false

**99. What is the Object.freeze() method in JavaScript?**

Object.freeze() prevents modifications to an object’s properties and values, making the object immutable.  
**Example:**

javascript

let obj = { name: 'John' };

Object.freeze(obj);

obj.name = 'Jane';

console.log(obj.name); // John

**100. What is Object.seal() in JavaScript?**

Object.seal() prevents new properties from being added to an object and marks all existing properties as non-configurable. It does not prevent modifications to the values of existing properties.  
**Example:**

javascript

let obj = { name: 'John' };

Object.seal(obj);

obj.name = 'Jane';

console.log(obj.name); // Jane

These should cover a broad range of topics for JavaScript interviews!

4o mini

More

ChatGPT

Certainly! Here are 30 more JavaScript interview questions with answers:

**101. What is Array.prototype.map() in JavaScript?**

Array.prototype.map() creates a new array populated with the results of calling a provided function on every element in the calling array.  
**Example:**

javascript

let numbers = [1, 2, 3];

let doubled = numbers.map(x => x \* 2);

console.log(doubled); // [2, 4, 6]

**102. What is the Array.prototype.filter() method in JavaScript?**

Array.prototype.filter() creates a new array with all elements that pass the test implemented by the provided function.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let even = numbers.filter(x => x % 2 === 0);

console.log(even); // [2, 4]

**103. What is the Array.prototype.reduceRight() method in JavaScript?**

Array.prototype.reduceRight() applies a function against an accumulator and each element (from right to left) in the array, to reduce it to a single value.  
**Example:**

javascript

let numbers = [1, 2, 3, 4];

let result = numbers.reduceRight((acc, val) => acc - val);

console.log(result); // -4

**104. What is Object.assign() in JavaScript?**

Object.assign() copies all enumerable own properties from one or more source objects to a target object. It returns the modified target object.  
**Example:**

javascript

let target = { a: 1 };

let source = { b: 2 };

Object.assign(target, source);

console.log(target); // { a: 1, b: 2 }

**105. What is Object.keys() in JavaScript?**

Object.keys() returns an array of a given object's own enumerable property names.  
**Example:**

javascript

let obj = { name: 'John', age: 30 };

let keys = Object.keys(obj);

console.log(keys); // ['name', 'age']

**106. What is Object.values() in JavaScript?**

Object.values() returns an array of a given object's own enumerable property values.  
**Example:**

javascript

let obj = { name: 'John', age: 30 };

let values = Object.values(obj);

console.log(values); // ['John', 30]

**107. What is Object.entries() in JavaScript?**

Object.entries() returns an array of a given object's own enumerable string-keyed property [key, value] pairs.  
**Example:**

javascript

let obj = { name: 'John', age: 30 };

let entries = Object.entries(obj);

console.log(entries); // [['name', 'John'], ['age', 30]]

**108. What does Array.prototype.concat() do in JavaScript?**

Array.prototype.concat() merges two or more arrays into a single array without modifying the original arrays.  
**Example:**

javascript

let arr1 = [1, 2];

let arr2 = [3, 4];

let merged = arr1.concat(arr2);

console.log(merged); // [1, 2, 3, 4]

**109. What is Array.prototype.sort() in JavaScript?**

Array.prototype.sort() sorts the elements of an array in place and returns the sorted array. By default, it sorts elements as strings.  
**Example:**

javascript

let numbers = [4, 2, 5, 1];

numbers.sort((a, b) => a - b);

console.log(numbers); // [1, 2, 4, 5]

**110. What is the Array.prototype.reverse() method in JavaScript?**

Array.prototype.reverse() reverses the elements of an array in place and returns the array.  
**Example:**

javascript

let arr = [1, 2, 3];

arr.reverse();

console.log(arr); // [3, 2, 1]

**111. What is the Array.prototype.findIndex() method in JavaScript?**

Array.prototype.findIndex() returns the index of the first element in the array that satisfies the provided testing function. If no elements satisfy the testing function, it returns -1.  
**Example:**

javascript

let numbers = [4, 9, 16];

let index = numbers.findIndex(num => num > 10);

console.log(index); // 2

**112. What is the Array.prototype.includes() method in JavaScript?**

Array.prototype.includes() checks if an array contains a certain element and returns true if it does, otherwise false.  
**Example:**

javascript

let fruits = ['apple', 'banana', 'mango'];

console.log(fruits.includes('banana')); // true

**113. What is the Array.prototype.splice() method in JavaScript?**

Array.prototype.splice() changes the contents of an array by removing, replacing, or adding elements in place.  
**Example:**

javascript

let arr = [1, 2, 3, 4];

arr.splice(2, 1, 'a', 'b');

console.log(arr); // [1, 2, 'a', 'b', 4]

**114. What is the Array.prototype.slice() method in JavaScript?**

Array.prototype.slice() returns a shallow copy of a portion of an array into a new array object, selected from start to end (end not included).  
**Example:**

javascript

let arr = [1, 2, 3, 4];

let sliced = arr.slice(1, 3);

console.log(sliced); // [2, 3]

**115. What is Array.prototype.flat() in JavaScript?**

Array.prototype.flat() creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.  
**Example:**

javascript

let arr = [1, [2, [3, [4]]]];

let flat = arr.flat(2);

console.log(flat); // [1, 2, 3, [4]]

**116. What is Array.prototype.flatMap() in JavaScript?**

Array.prototype.flatMap() first maps each element using a mapping function, then flattens the result into a new array.  
**Example:**

javascript

let arr = [1, 2, 3];

let flatMap = arr.flatMap(x => [x, x \* 2]);

console.log(flatMap); // [1, 2, 2, 4, 3, 6]

**117. What does String.prototype.split() do in JavaScript?**

String.prototype.split() divides a string into an array of substrings based on a specified separator.  
**Example:**

javascript

let str = 'a,b,c';

let arr = str.split(',');

console.log(arr); // ['a', 'b', 'c']

**118. What is the String.prototype.trim() method in JavaScript?**

String.prototype.trim() removes whitespace from both ends of a string and returns a new string.  
**Example:**

javascript

let str = ' Hello World ';

console.log(str.trim()); // 'Hello World'

**119. What is the String.prototype.replace() method in JavaScript?**

String.prototype.replace() returns a new string with some matches of a pattern replaced by a replacement.  
**Example:**

javascript

let str = 'Hello World';

let newStr = str.replace('World', 'JavaScript');

console.log(newStr); // Hello JavaScript

**120. What is the String.prototype.concat() method in JavaScript?**

String.prototype.concat() joins two or more strings and returns a new string.  
**Example:**

javascript

let str1 = 'Hello';

let str2 = 'World';

let result = str1.concat(' ', str2);

console.log(result); // Hello World

**121. What is JSON.stringify() in JavaScript?**

JSON.stringify() converts a JavaScript object or value to a JSON string.  
**Example:**

javascript

let obj = { name: 'John', age: 30 };

let jsonString = JSON.stringify(obj);

console.log(jsonString); // '{"name":"John","age":30}'

**122. What is JSON.parse() in JavaScript?**

JSON.parse() parses a JSON string, constructing the JavaScript value or object described by the string.  
**Example:**

javascript

let jsonString = '{"name":"John","age":30}';

let obj = JSON.parse(jsonString);

console.log(obj); // { name: 'John', age: 30 }

**123. What is the Function.prototype.call() method in JavaScript?**

Function.prototype.call() calls a function with a given this value and arguments provided individually.  
**Example:**

javascript

function greet(greeting, name) {

console.log(greeting + ', ' + name);

}

greet.call(null, 'Hello', 'John'); // Hello, John

**124. What is the Function.prototype.apply() method in JavaScript?**

Function.prototype.apply() calls a function with a given this value and arguments provided as an array (or an array-like object).  
**Example:**

javascript

function greet(greeting, name) {

console.log(greeting + ', ' + name);

}

greet.apply(null, ['Hello', 'John']); // Hello, John

**125. What is Function.prototype.bind() in JavaScript?**

Function.prototype.bind() creates a new function that, when called, has its this keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is invoked.  
**Example:**

javascript

function greet(greeting, name) {

console.log(greeting + ', ' + name);

}

let boundGreet = greet.bind(null, 'Hello');

boundGreet('John'); // Hello, John

**126. What is RegExp in JavaScript?**

RegExp is an object that represents a regular expression, which is used for pattern matching within strings.  
**Example:**

javascript

let regex = /hello/i;

console.log(regex.test('Hello World')); // true

**127. What is the RegExp.prototype.test() method in JavaScript?**

RegExp.prototype.test() tests if a pattern exists in a string, returning true if it does and false otherwise.  
**Example:**

javascript

let regex = /world/;

console.log(regex.test('Hello World')); // true

**128. What is the RegExp.prototype.exec() method in JavaScript?**

RegExp.prototype.exec() executes a search for a match in a string and returns an array of information or null if no match is found.  
**Example:**

javascript

let regex = /world/;

let result = regex.exec('Hello World');

console.log(result); // ['World']

**129. What is Set.prototype.has() in JavaScript?**

Set.prototype.has() returns a boolean indicating whether an element exists in the Set or not.  
**Example:**

javascript

let mySet = new Set([1, 2, 3]);

console.log(mySet.has(2)); // true

**130. What is WeakMap in JavaScript?**

WeakMap is a collection of key-value pairs where keys are objects and values can be any value. It allows for garbage collection of keys when they are no longer referenced elsewhere.  
**Example:**

javascript

let weakMap = new WeakMap();

let obj = {};

weakMap.set(obj, 'value');

console.log(weakMap.get(obj)); // 'value'

These questions and answers should further enhance your preparation for JavaScript interviews.

4o mini

More

ChatGPT

Certainly! Here are 30 more JavaScript interview questions with answers:

**131. What is the for...of loop in JavaScript?**

The for...of loop iterates over iterable objects like arrays, strings, or sets, executing a block of code for each element.  
**Example:**

javascript

let array = [1, 2, 3];

for (let value of array) {

console.log(value);

}

// Output: 1 2 3

**132. What is the for...in loop in JavaScript?**

The for...in loop iterates over the enumerable properties of an object, including properties inherited from its prototype chain.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

for (let key in obj) {

console.log(key, obj[key]);

}

// Output: a 1, b 2

**133. What is Symbol in JavaScript?**

Symbol is a primitive data type that represents a unique and immutable value, often used as property keys to avoid name collisions.  
**Example:**

javascript

let sym = Symbol('description');

let obj = { [sym]: 'value' };

console.log(obj[sym]); // value

**134. What is NaN in JavaScript?**

NaN stands for "Not-a-Number" and represents a value that is not a legal number. It is a special value of the number type.  
**Example:**

javascript

let result = 0 / 0;

console.log(result); // NaN

**135. What is the Number.isNaN() method in JavaScript?**

Number.isNaN() determines whether the value is NaN and its type is number. It does not coerce the value to a number.  
**Example:**

javascript

console.log(Number.isNaN(NaN)); // true

console.log(Number.isNaN('string')); // false

**136. What is the isNaN() function in JavaScript?**

The isNaN() function determines whether a value is NaN. It coerces the value to a number before checking.  
**Example:**

javascript

console.log(isNaN(NaN)); // true

console.log(isNaN('string')); // true

**137. What is Infinity in JavaScript?**

Infinity is a global property that represents positive infinity, which is a numeric value greater than any other number.  
**Example:**

javascript

let largeNumber = 1 / 0;

console.log(largeNumber); // Infinity

**138. What is -Infinity in JavaScript?**

-Infinity is a global property that represents negative infinity, which is a numeric value less than any other number.  
**Example:**

javascript

let smallNumber = -1 / 0;

console.log(smallNumber); // -Infinity

**139. What is the Math object in JavaScript?**

The Math object provides properties and methods for mathematical constants and functions, such as Math.max() or Math.sqrt().  
**Example:**

javascript

console.log(Math.max(1, 2, 3)); // 3

console.log(Math.sqrt(16)); // 4

**140. What is the Date object in JavaScript?**

The Date object represents a single moment in time and provides methods for date and time manipulation.  
**Example:**

javascript

let now = new Date();

console.log(now.toString()); // Current date and time

**141. What is the JSON object in JavaScript?**

The JSON object provides methods for parsing JSON strings into JavaScript objects and stringifying JavaScript objects into JSON strings.  
**Example:**

javascript

let jsonString = '{"name": "John"}';

let obj = JSON.parse(jsonString);

console.log(obj.name); // John

**142. What is Object.getPrototypeOf() in JavaScript?**

Object.getPrototypeOf() returns the prototype (i.e., the internal [[Prototype]] property) of a specified object.  
**Example:**

javascript

let obj = {};

console.log(Object.getPrototypeOf(obj) === Object.prototype); // true

**143. What is Object.setPrototypeOf() in JavaScript?**

Object.setPrototypeOf() sets the prototype (i.e., the internal [[Prototype]] property) of a specified object to another object or null.  
**Example:**

javascript

let obj = {};

let proto = { a: 1 };

Object.setPrototypeOf(obj, proto);

console.log(obj.a); // 1

**144. What is Reflect in JavaScript?**

The Reflect object provides methods for interceptable JavaScript operations and works similarly to Object but with additional capabilities.  
**Example:**

javascript

let obj = {};

Reflect.set(obj, 'key', 'value');

console.log(obj.key); // value

**145. What is Proxy in JavaScript?**

Proxy is a built-in object that enables you to create a proxy for another object, allowing you to intercept and customize operations performed on that object.  
**Example:**

javascript

let target = {};

let handler = {

get: (obj, prop) => 'Hello ' + prop

};

let proxy = new Proxy(target, handler);

console.log(proxy.name); // Hello name

**146. What is WeakSet in JavaScript?**

WeakSet is a collection of objects where the keys are objects and the values are not directly stored. Objects are weakly held, allowing them to be garbage collected.  
**Example:**

javascript

let weakSet = new WeakSet();

let obj = {};

weakSet.add(obj);

console.log(weakSet.has(obj)); // true

**147. What is WeakMap in JavaScript?**

WeakMap is a collection of key-value pairs where keys are objects and values can be any value. It allows for garbage collection of keys when they are no longer referenced.  
**Example:**

javascript

let weakMap = new WeakMap();

let obj = {};

weakMap.set(obj, 'value');

console.log(weakMap.get(obj)); // value

**148. What is Promise in JavaScript?**

Promise represents the eventual completion (or failure) of an asynchronous operation and its resulting value.  
**Example:**

javascript

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

setTimeout(() => resolve('Done'), 1000);

});

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

**149. What is Promise.prototype.then() in JavaScript?**

Promise.prototype.then() adds a fulfillment and rejection handler to the promise, and returns a new promise resolving to the return value of the called handler.  
**Example:**

javascript

let promise = Promise.resolve('Hello');

promise.then(value => console.log(value)); // Hello

**150. What is Promise.prototype.catch() in JavaScript?**

Promise.prototype.catch() adds a rejection handler to the promise and returns a new promise resolving to the final settled value.  
**Example:**

javascript

let promise = Promise.reject('Error');

promise.catch(error => console.log(error)); // Error

**151. What is Promise.prototype.finally() in JavaScript?**

Promise.prototype.finally() adds a handler to be executed when the promise is settled, regardless of its outcome.  
**Example:**

javascript

let promise = Promise.resolve('Done');

promise.finally(() => console.log('Promise settled')).then(result => console.log(result));

// Output: Promise settled

// Done

**152. What is Array.prototype.some() in JavaScript?**

Array.prototype.some() tests whether at least one element in the array passes the provided function. It returns true if any element passes, otherwise false.  
**Example:**

javascript

let numbers = [1, 2, 3];

let hasEven = numbers.some(num => num % 2 === 0);

console.log(hasEven); // true

**153. What is Array.prototype.every() in JavaScript?**

Array.prototype.every() tests whether all elements in the array pass the provided function. It returns true if all elements pass, otherwise false.  
**Example:**

javascript

let numbers = [2, 4, 6];

let allEven = numbers.every(num => num % 2 === 0);

console.log(allEven); // true

**154. What is String.prototype.startsWith() in JavaScript?**

String.prototype.startsWith() checks if a string starts with the specified substring. It returns true if it does, otherwise false.  
**Example:**

javascript

let str = 'Hello World';

console.log(str.startsWith('Hello')); // true

**155. What is String.prototype.endsWith() in JavaScript?**

String.prototype.endsWith() checks if a string ends with the specified substring. It returns true if it does, otherwise false.  
**Example:**

javascript

let str = 'Hello World';

console.log(str.endsWith('World')); // true

**156. What is String.prototype.includes() in JavaScript?**

String.prototype.includes() checks if a string contains the specified substring. It returns true if it does, otherwise false.  
**Example:**

javascript

let str = 'Hello World';

console.log(str.includes('World')); // true

**157. What is String.prototype.repeat() in JavaScript?**

String.prototype.repeat() returns a new string with the specified number of copies of the original string concatenated.  
**Example:**

javascript

let str = 'Hello';

console.log(str.repeat(3)); // HelloHelloHello

**158. What is Array.prototype.reduce() in JavaScript?**

Array.prototype.reduce() applies a function against an accumulator and each element in the array to reduce it to a single value.  
**Example:**

javascript

let numbers = [1, 2, 3];

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

console.log(sum); // 6

**159. What is Array.prototype.filter() in JavaScript?**

Array.prototype.filter() creates a new array with all elements that pass the test implemented by the provided function.  
**Example:**

javascript

let numbers = [1, 2, 3];

let evenNumbers = numbers.filter(num => num % 2 === 0);

console.log(evenNumbers); // [2]

**160. What is Array.prototype.map() in JavaScript?**

Array.prototype.map() creates a new array with the results of calling a provided function on every element in the calling array.  
**Example:**

javascript

let numbers = [1, 2, 3];

let doubled = numbers.map(num => num \* 2);

console.log(doubled); // [2, 4, 6]

Feel free to use these questions and answers to further prepare for your interviews!

4o mini

More even you can cover the ecma script right ? More IQs . why you did't oop concept ? why you did't cover ECMA?

ChatGPT

Memory updated

Absolutely, I'll include more questions covering ECMAScript features and object-oriented programming concepts. Here are additional JavaScript interview questions:

**ECMAScript and OOP Concepts**

**161. What is the ECMAScript specification?**

ECMAScript is a scripting language specification standardized by ECMA International, which JavaScript is based on. It defines the language’s syntax, semantics, and features.  
**Example:**  
ES6 (ECMAScript 2015) introduced features like classes and arrow functions.

**162. What are the main features introduced in ECMAScript 6 (ES6)?**

ES6 introduced several new features including arrow functions, classes, template literals, destructuring assignments, and let and const keywords.  
**Example:**

javascript

// Arrow function

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

// Class

class Person {

constructor(name) {

this.name = name;

}

}

**163. What is the purpose of let and const in ECMAScript 6?**

let allows block-scoped variable declaration, while const is used for declaring constants whose values cannot be reassigned.  
**Example:**

javascript

let variable = 1;

variable = 2; // Valid

const constant = 1;

constant = 2; // Error: Assignment to constant variable.

**164. What are template literals in ECMAScript 6?**

Template literals allow for embedded expressions and multi-line strings using backticks (). They can include placeholders with ${expression}` syntax.  
**Example:**

javascript

let name = 'John';

let greeting = `Hello, ${name}!`;

console.log(greeting); // Hello, John!

**165. What is destructuring in ECMAScript 6?**

Destructuring allows unpacking values from arrays or properties from objects into distinct variables.  
**Example:**

javascript

// Array destructuring

let [a, b] = [1, 2];

console.log(a, b); // 1 2

// Object destructuring

let { x, y } = { x: 1, y: 2 };

console.log(x, y); // 1 2

**166. What are spread and rest operators in ECMAScript 6?**

The spread operator (...) expands elements of an iterable into individual elements, while the rest operator collects multiple elements into a single array.  
**Example:**

javascript

// Spread operator

let arr = [1, 2, 3];

let newArr = [...arr, 4];

console.log(newArr); // [1, 2, 3, 4]

// Rest operator

function sum(...numbers) {

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

}

console.log(sum(1, 2, 3)); // 6

**167. What are classes in ECMAScript 6?**

Classes provide a more syntactically clear way to create objects and handle inheritance, using class and extends keywords.  
**Example:**

javascript

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise.`);

}

}

class Dog extends Animal {

speak() {

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

}

}

let dog = new Dog('Rex');

dog.speak(); // Rex barks.

**168. What is the async/await syntax in ECMAScript 7?**

The async/await syntax allows for writing asynchronous code in a synchronous manner, improving readability and handling of asynchronous operations.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

return data;

}

**169. What is Object.assign() in ECMAScript 6?**

Object.assign() copies the values of all enumerable own properties from one or more source objects to a target object.  
**Example:**

javascript

let target = { a: 1 };

let source = { b: 2 };

Object.assign(target, source);

console.log(target); // { a: 1, b: 2 }

**170. What is Symbol in ECMAScript 6?**

Symbol is a primitive data type that is used to create unique identifiers for object properties, which helps in avoiding property name collisions.  
**Example:**

javascript

let sym = Symbol('description');

let obj = { [sym]: 'value' };

console.log(obj[sym]); // value

**171. What is Proxy in ECMAScript 6?**

Proxy is an object that enables you to define custom behavior for fundamental operations (e.g., property lookup, assignment).  
**Example:**

javascript

let target = {};

let handler = {

get: (obj, prop) => `Property ${prop} accessed`

};

let proxy = new Proxy(target, handler);

console.log(proxy.someProperty); // Property someProperty accessed

**172. What are WeakSet and WeakMap in ECMAScript 6?**

WeakSet and WeakMap are collections where keys are weakly held, meaning they do not prevent garbage collection of objects.  
**Example:**

javascript

// WeakSet

let weakSet = new WeakSet();

let obj = {};

weakSet.add(obj);

console.log(weakSet.has(obj)); // true

// WeakMap

let weakMap = new WeakMap();

let key = {};

weakMap.set(key, 'value');

console.log(weakMap.get(key)); // value

**173. What is Array.prototype.find() in ECMAScript 6?**

Array.prototype.find() returns the first element in the array that satisfies the provided testing function, or undefined if no element is found.  
**Example:**

javascript

let numbers = [4, 9, 16];

let result = numbers.find(num => num > 5);

console.log(result); // 9

**174. What is Array.prototype.findIndex() in ECMAScript 6?**

Array.prototype.findIndex() returns the index of the first element in the array that satisfies the provided testing function, or -1 if no element is found.  
**Example:**

javascript

let numbers = [4, 9, 16];

let index = numbers.findIndex(num => num > 5);

console.log(index); // 1

**175. What is Array.prototype.includes() in ECMAScript 7?**

Array.prototype.includes() determines whether an array contains a certain element, returning true if it does, otherwise false.  
**Example:**

javascript

let numbers = [1, 2, 3];

console.log(numbers.includes(2)); // true

**176. What is the Object.entries() method in ECMAScript 7?**

Object.entries() returns an array of a given object's own enumerable string-keyed property [key, value] pairs.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

let entries = Object.entries(obj);

console.log(entries); // [['a', 1], ['b', 2]]

**177. What is the Object.values() method in ECMAScript 7?**

Object.values() returns an array of a given object's own enumerable string-keyed property values.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

let values = Object.values(obj);

console.log(values); // [1, 2]

**178. What are class fields in ECMAScript 2022?**

class fields are a way to declare properties directly within a class definition, including private fields prefixed with #.  
**Example:**

javascript

class MyClass {

#privateField = 'secret';

publicField = 'public';

getPrivate() {

return this.#privateField;

}

}

let instance = new MyClass();

console.log(instance.publicField); // public

console.log(instance.getPrivate()); // secret

**179. What is Array.prototype.at() in ECMAScript 2022?**

Array.prototype.at() returns the element at the given position, allowing for negative indexing.  
**Example:**

javascript

let array = [1, 2, 3];

console.log(array.at(-1)); // 3

**180. What is WeakRef in ECMAScript 2022?**

WeakRef provides a way to hold a weak reference to an object, allowing the object to be garbage collected if there are no other references to it.  
**Example:**

javascript

let obj = { name: 'example' };

let weakRef = new WeakRef(obj);

console.log(weakRef.deref()); // { name: 'example' }

**181. What is the class keyword in JavaScript?**

The class keyword introduces a way to create objects and handle inheritance more syntactically than using function constructors.  
**Example:**

javascript

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise.`);

}

}

**182. What is inheritance in JavaScript classes?**

Inheritance allows a class to use properties and methods from another class. This is achieved using the extends

4o mini

please continue

ChatGPT

**182. What is inheritance in JavaScript classes?**

Inheritance allows a class to use properties and methods from another class. This is achieved using the extends keyword to create a subclass.  
**Example:**

javascript

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise.`);

}

}

class Dog extends Animal {

speak() {

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

}

}

let dog = new Dog('Rex');

dog.speak(); // Rex barks.

**183. What is polymorphism in JavaScript classes?**

Polymorphism in JavaScript classes allows methods in different classes to have the same name but behave differently based on the class context.  
**Example:**

javascript

class Animal {

speak() {

console.log('Animal speaks');

}

}

class Dog extends Animal {

speak() {

console.log('Dog barks');

}

}

let animal = new Animal();

let dog = new Dog();

animal.speak(); // Animal speaks

dog.speak(); // Dog barks

**184. What is encapsulation in JavaScript classes?**

Encapsulation is the concept of restricting direct access to some of an object's components, typically through private or protected members.  
**Example:**

javascript

class Person {

#name;

constructor(name) {

this.#name = name;

}

getName() {

return this.#name;

}

}

let person = new Person('Alice');

console.log(person.getName()); // Alice

**185. What are getter and setter methods in JavaScript classes?**

Getter and setter methods allow you to define how properties are accessed and mutated. They use get and set keywords respectively.  
**Example:**

javascript

class Person {

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

set name(value) {

this.\_name = value;

}

}

let person = new Person('Alice');

console.log(person.name); // Alice

person.name = 'Bob';

console.log(person.name); // Bob

**186. What is the super keyword in JavaScript classes?**

The super keyword is used to call methods or access properties from a parent class within a subclass.  
**Example:**

javascript

class Animal {

speak() {

console.log('Animal speaks');

}

}

class Dog extends Animal {

speak() {

super.speak(); // Calls Animal's speak method

console.log('Dog barks');

}

}

let dog = new Dog();

dog.speak();

// Animal speaks

// Dog barks

**187. What are static methods in JavaScript classes?**

Static methods are defined on the class itself rather than on instances of the class. They are called using the class name.  
**Example:**

javascript

class MathUtil {

static add(x, y) {

return x + y;

}

}

console.log(MathUtil.add(2, 3)); // 5

**188. What is the purpose of Object.create() in JavaScript?**

Object.create() creates a new object with the specified prototype object and properties. It is useful for implementing inheritance.  
**Example:**

javascript

let animal = { type: 'animal' };

let dog = Object.create(animal);

dog.breed = 'Labrador';

console.log(dog.type); // animal

**189. What is the Reflect API in ECMAScript 2015?**

The Reflect API provides methods to interact with and manipulate objects and their properties, and is used for meta-programming.  
**Example:**

javascript

let obj = { a: 1 };

Reflect.set(obj, 'b', 2);

console.log(obj.b); // 2

**190. What are WeakSet and WeakMap used for in JavaScript?**

WeakSet and WeakMap are collections where the keys or values are weakly referenced, meaning they do not prevent garbage collection.  
**Example:**

javascript

let weakSet = new WeakSet();

let obj = {};

weakSet.add(obj);

console.log(weakSet.has(obj)); // true

let weakMap = new WeakMap();

weakMap.set(obj, 'value');

console.log(weakMap.get(obj)); // value

**191. What is the Proxy object used for in JavaScript?**

The Proxy object allows you to define custom behavior for fundamental operations on objects, like property lookup and assignment.  
**Example:**

javascript

let handler = {

get: (target, prop, receiver) => `Property ${prop} accessed`

};

let proxy = new Proxy({}, handler);

console.log(proxy.someProperty); // Property someProperty accessed

**192. How do you create a private field in JavaScript classes?**

Private fields in JavaScript classes are created using the # syntax and are only accessible within the class that defines them.  
**Example:**

javascript

class MyClass {

#privateField = 'secret';

getPrivateField() {

return this.#privateField;

}

}

let instance = new MyClass();

console.log(instance.getPrivateField()); // secret

**193. What are Object.entries() and Object.values() methods used for?**

Object.entries() returns an array of [key, value] pairs from an object, while Object.values() returns an array of the object's values.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

console.log(Object.entries(obj)); // [['a', 1], ['b', 2]]

console.log(Object.values(obj)); // [1, 2]

**194. What is a Symbol in JavaScript and how is it used?**

Symbol is a primitive data type used to create unique identifiers for object properties, preventing accidental property name collisions.  
**Example:**

javascript

let sym = Symbol('description');

let obj = { [sym]: 'value' };

console.log(obj[sym]); // value

**195. What is the async function in JavaScript?**

The async function allows you to write asynchronous code that looks synchronous, using await for promises and simplifying error handling.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

return data;

}

**196. What is Array.prototype.flat() in ECMAScript 2019?**

Array.prototype.flat() creates a new array with all sub-array elements concatenated into it recursively up to a specified depth.  
**Example:**

javascript

let arr = [1, [2, [3, [4]]]];

console.log(arr.flat(2)); // [1, 2, 3, [4]]

**197. What is Array.prototype.flatMap() in ECMAScript 2019?**

Array.prototype.flatMap() first maps each element using a mapping function, then flattens the result into a new array.  
**Example:**

javascript

let arr = [1, 2, 3];

console.log(arr.flatMap(x => [x, x \* 2])); // [1, 2, 2, 4, 3, 6]

**198. What is the Number.isNaN() method in ECMAScript 2015?**

Number.isNaN() determines whether the passed value is NaN and its type is number. It is more reliable than the global isNaN() function.  
**Example:**

javascript

console.log(Number.isNaN(NaN)); // true

console.log(Number.isNaN('string')); // false

**199. What is Object.fromEntries() in ECMAScript 2019?**

Object.fromEntries() transforms a list of key-value pairs into an object. It is the reverse operation of Object.entries().  
**Example:**

javascript

let entries = [['a', 1], ['b', 2]];

let obj = Object.fromEntries(entries);

console.log(obj); // { a: 1, b: 2 }

**200. What is the WeakRef class in ECMAScript 2022?**

WeakRef provides a way to hold a weak reference to an object, allowing the object to be garbage collected if there are no other references.  
**Example:**

javascript

let obj = { name: 'example' };

let weakRef = new WeakRef(obj);

console.log(weakRef.deref()); // { name: 'example' }

Feel free to use these questions to cover a broad range of ECMAScript and OOP concepts!

4o mini

More

ChatGPT

Here are more JavaScript interview questions with answers:

**201. What is the Promise.allSettled() method?**

Promise.allSettled() returns a promise that resolves after all of the given promises have either resolved or rejected. It provides an array of objects describing the outcome of each promise.  
**Example:**

javascript

let p1 = Promise.resolve(1);

let p2 = Promise.reject('error');

Promise.allSettled([p1, p2]).then(results => {

console.log(results);

// [{status: 'fulfilled', value: 1}, {status: 'rejected', reason: 'error'}]

});

**202. What is the Object.hasOwn() method in ECMAScript 2022?**

Object.hasOwn() is a method to determine if an object has a specific property as its own property, similar to hasOwnProperty(), but with a simpler syntax.  
**Example:**

javascript

let obj = { a: 1 };

console.log(Object.hasOwn(obj, 'a')); // true

console.log(Object.hasOwn(obj, 'b')); // false

**203. What is the import.meta object in JavaScript?**

The import.meta object provides metadata about the current module, including the URL of the module.  
**Example:**

javascript

console.log(import.meta.url); // The URL of the current module

**204. What is the String.prototype.replaceAll() method?**

String.prototype.replaceAll() replaces all occurrences of a substring or pattern with a specified replacement. It is similar to replace() but with global replacement.  
**Example:**

javascript

let str = 'foo foo';

console.log(str.replaceAll('foo', 'bar')); // bar bar

**205. What is the Array.prototype.at() method in ECMAScript 2022?**

Array.prototype.at() returns the element at a given position from the end of the array, allowing negative indices.  
**Example:**

javascript

let arr = [1, 2, 3];

console.log(arr.at(-1)); // 3

**206. What is a WeakSet in JavaScript?**

WeakSet is a collection of objects where the objects are held weakly, meaning they can be garbage collected if there are no other references to them.  
**Example:**

javascript

let weakSet = new WeakSet();

let obj = {};

weakSet.add(obj);

console.log(weakSet.has(obj)); // true

obj = null; // The object can be garbage collected

**207. What is a WeakMap in JavaScript?**

WeakMap is a collection of key-value pairs where keys are objects and are held weakly, allowing the key-value pairs to be garbage collected if there are no other references.  
**Example:**

javascript

let weakMap = new WeakMap();

let obj = {};

weakMap.set(obj, 'value');

console.log(weakMap.get(obj)); // value

obj = null; // The key-value pair can be garbage collected

**208. What is the Array.prototype.sort() method in JavaScript?**

Array.prototype.sort() sorts the elements of an array in place and returns the sorted array. The default sort order is according to string Unicode code points.  
**Example:**

javascript

let arr = [3, 1, 2];

arr.sort();

console.log(arr); // [1, 2, 3]

**209. What is the Array.prototype.reduce() method in JavaScript?**

Array.prototype.reduce() executes a reducer function on each element of the array, resulting in a single output value.  
**Example:**

javascript

let sum = [1, 2, 3].reduce((accumulator, currentValue) => accumulator + currentValue, 0);

console.log(sum); // 6

**210. What is the Array.prototype.find() method in JavaScript?**

Array.prototype.find() returns the first element in the array that satisfies the provided testing function.  
**Example:**

javascript

let arr = [5, 12, 8, 130, 44];

let found = arr.find(element => element > 10);

console.log(found); // 12

**211. What is the Array.prototype.includes() method in JavaScript?**

Array.prototype.includes() determines whether an array contains a certain element, returning true or false.  
**Example:**

javascript

let arr = [1, 2, 3];

console.log(arr.includes(2)); // true

console.log(arr.includes(4)); // false

**212. What is the Set object in JavaScript?**

Set is a collection of unique values where each value must be unique and can be of any type.  
**Example:**

javascript

let set = new Set([1, 2, 2, 3]);

console.log([...set]); // [1, 2, 3]

**213. What is the Map object in JavaScript?**

Map is a collection of key-value pairs where keys and values can be of any type. Keys are unique, and the order of elements is maintained.  
**Example:**

javascript

let map = new Map();

map.set('key1', 'value1');

console.log(map.get('key1')); // value1

**214. What is the JSON.stringify() method used for?**

JSON.stringify() converts a JavaScript object or value to a JSON string. It is useful for serialization.  
**Example:**

javascript

let obj = { name: 'Alice', age: 25 };

let jsonStr = JSON.stringify(obj);

console.log(jsonStr); // '{"name":"Alice","age":25}'

**215. What is the JSON.parse() method used for?**

JSON.parse() parses a JSON string and constructs the JavaScript value or object described by the string.  
**Example:**

javascript

let jsonStr = '{"name":"Alice","age":25}';

let obj = JSON.parse(jsonStr);

console.log(obj); // { name: 'Alice', age: 25 }

**216. What is the Object.keys() method in JavaScript?**

Object.keys() returns an array of a given object's own enumerable property names.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

console.log(Object.keys(obj)); // ['a', 'b']

**217. What is the Object.values() method in JavaScript?**

Object.values() returns an array of a given object's own enumerable property values.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

console.log(Object.values(obj)); // [1, 2]

**218. What is the Object.entries() method in JavaScript?**

Object.entries() returns an array of a given object's own enumerable string-keyed property [key, value] pairs.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

console.log(Object.entries(obj)); // [['a', 1], ['b', 2]]

**219. What is the for...in loop used for in JavaScript?**

The for...in loop iterates over the enumerable properties of an object, including inherited properties.  
**Example:**

javascript

let obj = { a: 1, b: 2 };

for (let key in obj) {

console.log(key, obj[key]);

}

// Output: a 1, b 2

**220. What is the for...of loop used for in JavaScript?**

The for...of loop iterates over iterable objects, such as arrays, strings, or other collections. It does not work with objects.  
**Example:**

javascript

let arr = [1, 2, 3];

for (let value of arr) {

console.log(value);

}

// Output: 1, 2, 3

**221. What is async/await in JavaScript?**

async/await simplifies working with asynchronous code. async declares a function as asynchronous, and await pauses the function execution until a promise resolves.  
**Example:**

javascript

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

return data;

}

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

Template literals allow you to embed expressions inside strings using ${expression} syntax and support multi-line strings.  
**Example:**

javascript

let name = 'Alice';

let greeting = `Hello, ${name}!`;

console.log(greeting); // Hello, Alice!

**223. What is destructuring assignment in JavaScript?**

Destructuring assignment extracts values from arrays or properties from objects into distinct variables.  
**Example:**

javascript

let [a, b] = [1, 2];

let { x, y } = { x: 10, y: 20 };

console.log(a, b); // 1 2

console.log(x, y); // 10 20

**224. What is the spread operator in JavaScript?**

The spread operator (...) expands elements of an array or object into individual elements or properties.  
**Example:**

javascript

let arr = [1, 2, 3];

let newArr = [...arr, 4, 5];

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

**225. What is the rest parameter in JavaScript functions?**

The rest parameter (...) allows a function to accept an indefinite number of arguments as an array.  
**Example:**

javascript

function sum(...numbers) {

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

}

console.log(sum(1, 2, 3)); // 6

**226. What is the Map object in JavaScript?**

Map is a collection of keyed data items, similar to an object, but with keys that can be of any type and maintain insertion order.  
**Example:**

javascript

let map = new Map();

map.set('key1', 'value1');

console.log(map.get('key1')); // value1

**227. What is the Set object in JavaScript?**

Set is a collection of unique values, where each value can only occur once, and the insertion order is preserved.  
**Example:**

javascript

let set = new Set([1, 2, 2, 3]);

console.log(set); // Set {1, 2, 3}

**228. What is the Array.prototype.concat() method?**

Array.prototype.concat() merges two or more arrays into a single array.  
**Example:**

javascript

let arr1 = [1, 2];

let arr2 = [3, 4];

let arr3 = arr1.concat(arr2);

console.log(arr3); // [1, 2, 3, 4]

**229. What is the Array.prototype.map() method?**

Array.prototype.map() creates a new array with the results of calling a provided function on every element in the array.  
**Example:**

javascript

let arr = [1, 2, 3];

let doubled = arr.map(x => x \* 2);

console.log(doubled); // [2, 4, 6]

**230. What is the Array.prototype.filter() method?**

Array.prototype.filter() creates a new array with all elements that pass the test implemented by the provided function.  
**Example:**

javascript

let arr = [1, 2, 3, 4];

let even = arr.filter(x => x % 2 === 0);

console.log(even); // [2, 4]

**231. What is the Array.prototype.findIndex() method?**

Array.prototype.findIndex() returns the index of the first element in the array that satisfies the provided testing function.  
**Example:**

javascript

let arr = [5, 12, 8, 130, 44];

let index = arr.findIndex(element => element > 10);

console.log(index); // 1

**232. What is the Array.prototype.some() method?**

Array.prototype.some() tests whether at least one element in the array passes the test implemented by the provided function.  
**Example:**

javascript

let arr = [1, 2, 3, 4];

let hasEven = arr.some(x => x % 2 === 0);

console.log(hasEven); // true

**233. What is the Array.prototype.every() method?**

Array.prototype.every() tests whether all elements in the array pass the test implemented by the provided function.  
**Example:**

javascript

let arr = [2, 4, 6];

let allEven = arr.every(x => x % 2 === 0);

console.log(allEven); // true

**234. What is the String.prototype.trim() method?**

String.prototype.trim() removes whitespace from both ends of a string.  
**Example:**

javascript

let str = ' Hello ';

console.log(str.trim()); // 'Hello'

**235. What is the String.prototype.charAt() method?**

String.prototype.charAt() returns the character at a specified index in a string.  
**Example:**

javascript

let str = 'Hello';

console.log(str.charAt(1)); // 'e'

**236. What is the String.prototype.slice() method?**

String.prototype.slice() extracts a section of a string and returns it as a new string.  
**Example:**

javascript

let str = 'Hello, world!';

console.log(str.slice(0, 5)); // 'Hello'

**237. What is the String.prototype.toUpperCase() method?**

String.prototype.toUpperCase() returns a new string with all characters converted to uppercase.  
**Example:**

javascript

let str = 'hello';

console.log(str.toUpperCase()); // 'HELLO'

**238. What is the String.prototype.toLowerCase() method?**

String.prototype.toLowerCase() returns a new string with all characters converted to lowercase.  
**Example:**

javascript

let str = 'HELLO';

console.log(str.toLowerCase()); // 'hello'

**239. What is the String.prototype.includes() method?**

String.prototype.includes() determines whether a string contains a specified substring.  
**Example:**

javascript

let str = 'Hello, world!';

console.log(str.includes('world')); // true

**240. What is the Array.prototype.sort() method?**

Array.prototype.sort() sorts the elements of an array in place and returns the sorted array. The sort order is determined by converting elements to strings and comparing their sequences of UTF-16 code units values.  
**Example:**

javascript

let arr = [4, 2, 5, 1, 3];

arr.sort((a, b) => a - b);

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

These questions and answers cover a range of ECMAScript features and JavaScript concepts, including newer ECMAScript versions and common language constructs.