JS Functions Methods Interview

Here's a curated list of **50 JavaScript interview questions** on **Functions & Methods**, ranging from basic to advanced, commonly asked in interviews:

🟢 Basic-Level Questions (1–15)

1. What is the difference between a function declaration and a function expression?
2. How do you define a function in JavaScript?
3. What are first-class functions in JavaScript?
4. What is the difference between `call()`, `apply()`, and `bind()` methods?
5. What are arrow functions and how are they different from regular functions?
6. How do default parameters work in JavaScript functions?
7. What is the purpose of the `return` statement in a function?
8. What is the `arguments` object in JavaScript?
9. How do you pass arguments by reference vs by value?
10. Can a function return another function? Give an example.
11. How can you make a function parameter optional?
12. What is recursion? Provide a simple example.
13. What is function hoisting in JavaScript?
14. Can you explain anonymous functions?
15. How do IIFE (Immediately Invoked Function Expressions) work?

🟡 Intermediate-Level Questions (16–35)

16. What is a callback function? Give an example.
17. Explain closures with an example.
18. What are higher-order functions?

19. What is the use of `setTimeout()` and `setInterval()` functions?
20. How can you create a function that remembers its lexical scope?
21. What is function currying? Provide an example.
22. Explain the difference between synchronous and asynchronous functions.
23. What is the difference between rest parameters and the `arguments` object?
24. What is tail call optimization?
25. How can you memoize a function in JavaScript?
26. What is the spread operator and how is it used with functions?
27. Explain the concept of pure functions.
28. What does it mean for a function to have side effects?
29. Can functions have properties in JavaScript?
30. How does JavaScript handle function overloading?
31. What is a generator function? Provide an example.
32. How do you handle errors in functions?
33. What are async/await functions?
34. What happens when you call a function with more or fewer arguments than declared?
35. Can functions be stored in variables or passed as arguments?

🔴 Advanced-Level Questions (36–50)

36. How does the `this` keyword behave inside different types of functions?
37. How does lexical scoping work in functions?
38. Explain debounce and throttle in the context of function execution.
39. What is the role of function factories?
40. How does JavaScript handle memory for function closures?
41. What are arrow function limitations (like with `this`, `arguments`, etc)?
42. Explain the use of `Function.prototype` methods.
43. How can you implement a custom `bind()` function?

44. Explain trampolining in JavaScript.
45. What is a function composition?
46. How do you implement a polyfill for `Function.prototype.bind()`?
47. What are function decorators and how are they used?
48. How can you simulate private methods using closures?
49. Can a function be self-memoizing? Give an example.
50. How do named function expressions behave differently from anonymous ones?

Let me know if you want answers, examples, or code snippets for any of these!

Perfect! Here are **50 **technical** interview questions on **JavaScript Functions & Methods**, categorized by difficulty with a focus on **practical, code-based, and concept-heavy** content:

🟢 Basic Technical Questions (1–15)

1. Write a function to check if a number is even.
2. Explain what `typeof functionName` returns for a declared function.
3. Write a function to reverse a string.
4. What's the output of:

```js
function foo() { console.log(this); }
foo();
```

5. What is the output of:

```js
function a(x) { return x * 2; }
console.log(a(4));
```

6. How do you make a function that accepts any number of arguments and sums them?
7. What's wrong with this code?

```js
function add(a, b) {
  console.log(a + b);
}
add(1);
```

8. Write a function that returns the factorial of a number.
9. What does this return?

```js
(function() {
  return typeof arguments;
})();
```

10. Can you use a function before declaring it?
11. What's the difference between this:

```js
function f() {}
```

and this:

```js
const f = function() {};
```

12. Write an arrow function to multiply two numbers.
13. Convert a regular function to an arrow function:

```js
function double(x) { return x * 2; }
```

14. What happens when you `return` inside a function with no value?

15. Explain how default function parameters work with an example.

🟡 Intermediate Technical Questions (16–35)

16. Implement a function that takes a callback and executes it after 2 seconds.
17. How does `this` behave differently in arrow functions vs regular ones?
18. Convert the following to use rest parameters:

```js
function sum() {
  let total = 0;
  for (let i = 0; i < arguments.length; i++) {
    total += arguments[i];
  }
  return total;
}
```

19. Create a function that returns another function (closure).
20. Use `bind()` to set the context of a function.
21. Write a function that returns a promise which resolves after 3 seconds.
22. What will this log?

```js
const obj = {
  value: 10,
  logValue: function() {
    setTimeout(function() {
      console.log(this.value);
    }, 1000);
  }
};
obj.logValue();
```

23. Currify this function:

```js
function add(a, b, c) { return a + b + c; }
```

24. Implement a debounce function.
25. Write a recursive function to compute the nth Fibonacci number.
26. What's the output?

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

27. How do you preserve the context of `this` inside a nested function?
28. Write a function that logs arguments using `...rest` and `forEach`.
29. Implement a function that memoizes another function.
30. Write a function that uses a callback and also returns a value.
31. What is a function expression vs a function declaration? Which is hoisted?
32. Modify this code so it logs `Hello` only once every 1 second even if called rapidly:

```js
function greet() { console.log("Hello"); }
```

33. Write a polyfill for `Function.prototype.bind`.
34. Explain why arrow functions don't have their own `arguments` object.
35. What will this log and why?

```js
const a = () => { return { name: "JS" }; };
console.log(a());
```

🔴 Advanced Technical Questions (36–50)

36. Implement a function composition utility.
37. Use `apply()` to invoke a function with an array of arguments.
38. How does JavaScript handle internal scopes during recursion?
39. Simulate private variables using closures.
40. Implement a throttle function.
41. Explain lexical scoping with code.
42. How does tail call optimization benefit recursion?
43. Modify this function so that it only executes once:

```js
function init() { console.log("Init called"); }
```

44. Write a function that delays execution of another function using `setTimeout`.
45. Implement a retry logic that tries a function 3 times before failing.
46. Write a generator function that yields numbers 1 to 5.
47. How do you create a self-executing function that returns a counter?
48. What does this log and why?

```js
const obj = {
  value: 10,
  getValue: () => this.value
```

```
};
console.log(obj.getValue());
```

49. Explain how closures can lead to memory leaks and how to avoid them.
50. Implement a deep clone function using recursion.

Let me know if you want code solutions or explanations for any of these questions!

Here are **50 technical interview questions on the DOM (Document Object Model)** in JavaScript, covering **selectors, events, traversal, manipulation, performance, and advanced concepts**:

🟢 Basic DOM Questions (1–15)

1. What is the DOM in JavaScript?
2. How do you select an element by ID in the DOM?
3. What's the difference between `getElementById` and `querySelector`?
4. How do you change the text of an HTML element using JavaScript?
5. How do you change the style of an element dynamically?
6. How can you get the value from an input field using JavaScript?
7. How do you create a new HTML element in JavaScript?
8. How do you add a new child element to a parent element?
9. What is the difference between `innerHTML` and `textContent`?
10. How do you remove an element from the DOM?
11. What's the difference between `appendChild()` and `append()`?
12. How do you handle a button click using JavaScript?
13. What are DOM events? Give a few examples.
14. How do you prevent a form from submitting using JavaScript?

15. How can you check if an element has a certain class?

🟡 Intermediate DOM Questions (16–35)

16. How do you toggle a class on an element using JavaScript?
17. What's the difference between `node` and `element`?
18. How do you attach an event listener to multiple elements?
19. How do you delegate an event from a parent to child elements?
20. What is the purpose of `event.preventDefault()`?
21. What's the difference between `event.stopPropagation()` and `event.stopImmediatePropagation()`?
22. How do you get all elements with a specific class?
23. How do you get the parent of a DOM element?
24. What is the difference between `children` and `childNodes`?
25. How do you get the next sibling of an element?
26. How do you clone a DOM element?
27. What does `document.readyState` represent?
28. What is the DOMContentLoaded event?
29. How do you detect when the user scrolls the page?
30. How do you access custom data attributes (data-*)?
31. How do you programmatically trigger a click event?
32. What's the difference between `setAttribute()` and dot notation (`element.src`)?
33. What are inline, internal, and external event handlers?
34. How do you check if an element is visible in the viewport?
35. What are the pros/cons of using `innerHTML` to inject content?

🔴 Advanced DOM Questions (36–50)

36. What is the difference between `appendChild()` and `insertBefore()`?
37. Explain how event bubbling and capturing work in the DOM.
38. How does `document.createDocumentFragment()` improve performance?
39. What is reflow and repaint in the context of the DOM?
40. How can you minimize DOM reflows for better performance?
41. How do you use `MutationObserver` to watch DOM changes?
42. Explain the purpose of the `shadow DOM`.
43. How can you debounce scroll or input events for performance?
44. How do you dynamically load external scripts using DOM manipulation?
45. How do you create and dispatch custom events?
46. How does `closest()` differ from `parentElement`?
47. What is the difference between `window.onload` and `DOMContentLoaded`?
48. How do you safely insert user-generated content into the DOM?
49. What is the difference between static and live NodeLists?
50. How can you detect when an element's size or position changes?

Let me know if you want **code examples**, **answers**, or **flashcards** based on these!

Here are **20 technical interview questions** focused on **Events in JavaScript**, covering basics, propagation, delegation, and advanced use:

🟢 Basic Event Questions (1–7)

1. What is an event in JavaScript?
2. How do you add an event listener to a DOM element?
3. What is the syntax for `addEventListener()`?

4. What is the difference between `onclick` and `addEventListener()`?
5. How do you remove an event listener?
6. What are some common DOM events?
7. How can you prevent the default behavior of an event (e.g., form submission)?

🟡 Intermediate Event Questions (8–15)

8. What is event bubbling in JavaScript?
9. What is event capturing (or trickling)?
10. What's the difference between `event.target` and `event.currentTarget`?
11. What does `event.stopPropagation()` do?
12. What does `event.stopImmediatePropagation()` do?
13. How do you implement event delegation?
14. Why is event delegation useful?
15. How do you handle multiple events (like both `click` and `mouseover`) on the same element?

🔴 Advanced Event Questions (16–20)

16. What is the third parameter in `addEventListener(type, listener, useCapture)` used for?
17. How do you create and dispatch a custom event?
18. What is a passive event listener and when should it be used?
19. How does debouncing or throttling relate to events like `scroll` or `resize`?
20. What are synthetic events in frameworks like React, and how do they relate to native events?

Let me know if you want **answers, explanations, or sample code** for any of these!

Here are **30 technical interview questions** on **Classes & Objects in JavaScript**, spanning **object creation, class syntax, inheritance, and advanced concepts**:

🟢 Basic Level (1–10)

1. What is an object in JavaScript?
2. How do you create an object using object literals?
3. How do you access and modify object properties?
4. What is the difference between dot notation and bracket notation?
5. How do you add a method to an object?
6. What is the `this` keyword inside an object method?
7. What are constructor functions?
8. How do you create an object using a constructor function?
9. What is a class in JavaScript?
10. How do you create an instance of a class?

🟡 Intermediate Level (11–20)

11. What is the purpose of the `constructor` method in a class?
12. How do you define methods inside a class?
13. How does inheritance work using classes in JavaScript?
14. What is the `extends` keyword used for?
15. How do you call a parent class constructor?
16. What's the difference between a method and a function in an object context?
17. How do you check if a property exists in an object?
18. What's the difference between `Object.create()` and a constructor function?

19. What are static methods in a class?
20. How do you loop through all properties of an object?

🔴 Advanced Level (21–30)

21. How do you make a property private in a class (ES6+ syntax)?
22. What is prototypal inheritance?
23. What's the difference between class-based and prototype-based inheritance?
24. How do you use `Object.assign()`?
25. What are getters and setters in JavaScript classes?
26. How do you override a method in a subclass?
27. What is the difference between `Object.keys()`, `Object.values()`, and `Object.entries()`?
28. How does JavaScript handle `this` in arrow functions inside classes?
29. What is the purpose of `super()` in class constructors?
30. Can you add properties to an object after it has been created? How?

Let me know if you want **answers**, **code examples**, or **flashcards** for quick revision!

Here are **50 technical interview questions** on **Callbacks, Promises, and Async/Await in JavaScript**, covering **fundamentals, syntax, error handling, chaining, and best practices**.

🟢 Callbacks (1–15)

1. What is a callback function in JavaScript?
2. How do you pass a function as a callback?
3. What is the use of callbacks in asynchronous programming?

4. What are the advantages of using callbacks?
5. What are the problems with callbacks?
6. What is "callback hell"?
7. How can you avoid callback hell?
8. Write a simple example of a callback-based function.
9. What is the difference between synchronous and asynchronous callbacks?
10. Can a function return and also accept a callback?
11. How do you handle errors in callbacks?
12. Explain the inversion of control in the context of callbacks.
13. How are callbacks used in event listeners?
14. What is the difference between a named and anonymous callback?
15. When would you prefer callbacks over promises?

🟡 Promises (16–35)

16. What is a Promise in JavaScript?
17. What are the three states of a Promise?
18. How do you create a Promise?
19. What does `resolve()` and `reject()` do in a Promise?
20. What is `.then()` used for?
21. What is `.catch()` used for?
22. How do you chain Promises?
23. What happens if you return a value inside a `.then()` block?
24. How do you return another Promise from a `.then()` block?
25. What is `Promise.all()` and when should it be used?
26. What happens if one Promise in `Promise.all()` fails?
27. What is `Promise.race()`?
28. What is `Promise.any()` and how does it differ from `Promise.race()`?

29. What is `Promise.allSettled()`?
30. What is the difference between `then().catch()` and `try { await } catch`?
31. How do you convert a callback-based function to return a Promise?
32. What is a microtask queue and how does it relate to Promises?
33. Can you cancel a Promise?
34. What is an unhandled promise rejection?
35. How do Promises differ from callbacks?

🔵 Async/Await (36–50)

36. What is `async` / `await` in JavaScript?
37. How do you write an async function?
38. What does `await` do?
39. Can you use `await` outside of an `async` function?
40. What happens if you `await` a non-Promise value?
41. How do you handle errors in `async/await`?
42. Can `await` be used inside loops?
43. How do you use `Promise.all()` with `async/await`?
44. What is the performance impact of `await` inside a loop vs parallel execution?
45. What is top-level `await`?
46. How do `async/await` help with readability compared to Promises?
47. How does the call stack behave with async functions?
48. How does `await` affect the event loop?
49. What's the difference between `await` and `.then()`?
50. When would you use Promises over `async/await`?

Let me know if you'd like **answers**, **code snippets**, or a **quiz format** for these!

Here are **30 technical and interview questions** on the **Fetch API in JavaScript**, covering fundamentals, usage patterns, error handling, and comparison with other tools like Axios and XMLHttpRequest.

🟢 Basic Fetch API Questions (1–10)

1. What is the Fetch API in JavaScript?
2. How do you make a simple GET request using Fetch?
3. What does the `fetch()` function return?
4. Is `fetch()` synchronous or asynchronous?
5. How do you convert the response of a `fetch()` request to JSON?
6. How do you make a POST request using `fetch()`?
7. How do you send headers (e.g., authorization) with a `fetch()` request?
8. What happens if a `fetch()` request fails?
9. How do you catch errors in a `fetch()` call?
10. What is the default method used by `fetch()`?

🟡 Intermediate Fetch API Questions (11–20)

11. How do you handle HTTP errors (like 404 or 500) with `fetch()`?
12. Why doesn't `fetch()` reject the Promise on HTTP error status codes?
13. How do you use `fetch()` with async/await?
14. What is the difference between `response.ok` and `response.status`?
15. How do you upload a file using the Fetch API?
16. What are CORS issues and how do they relate to `fetch()`?
17. How do you send data in `application/x-www-form-urlencoded` format using `fetch()`?

18. How do you abort a `fetch()` request?
19. What is the role of `AbortController` with `fetch()`?
20. How can you set a timeout on a fetch request?

🔴 Advanced & Comparison Questions (21–30)

21. Compare `fetch()` with `XMLHttpRequest`.
22. Compare `fetch()` with Axios.
23. What are the limitations of the Fetch API?
24. How does streaming work with the Fetch API (using `response.body`)?
25. How do you read plain text or blob data from a `fetch()` response?
26. Can you retry a failed fetch request automatically? How?
27. How do you handle multiple fetch requests in parallel?
28. What is the difference between `mode: 'cors'` vs `mode: 'no-cors'` in fetch options?
29. How can you debug a `fetch()` request in the browser?
30. How do you use `fetch()` in Service Workers or Progressive Web Apps (PWA)?

Would you like answers or code examples for any of these?

Here are the **50 most commonly asked JavaScript interview questions**, covering **basics to advanced concepts** that are regularly asked in technical rounds:

🟢 Basics & Fundamentals (1–15)

1. What is JavaScript?
2. How is JavaScript different from Java?
3. Explain `var`, `let`, and `const`.

4. What are data types in JavaScript?
5. What is hoisting in JavaScript?
6. What is the difference between `==` and `===`?
7. What are truthy and falsy values?
8. What is the difference between `null` and `undefined`?
9. What is a closure?
10. What is the scope of a variable?
11. What is lexical scope?
12. Explain the concept of the execution context.
13. What is the call stack?
14. What is a callback function?
15. What is event bubbling and capturing?

🟡 Functions, Objects, and Arrays (16–30)

16. What is the difference between a function declaration and expression?
17. What is the difference between arrow functions and regular functions?
18. How does the `this` keyword work?
19. What are objects in JavaScript?
20. What are arrays in JavaScript?
21. How do you iterate over arrays?
22. What are higher-order functions?
23. What is the spread operator?
24. What is destructuring in JavaScript?
25. What is a prototype?
26. What is prototypal inheritance?
27. What are classes in JavaScript?
28. What is the difference between `map()`, `filter()`, and `reduce()`?

29. How does optional chaining (`?.`) work?
30. What is the use of the `in` and `instanceof` operators?

🔵 Asynchronous JavaScript (31–40)

31. What is the event loop in JavaScript?
32. What is the difference between synchronous and asynchronous code?
33. What are Promises?
34. What is async/await?
35. How do you handle errors in async/await?
36. What is the Fetch API?
37. What is a callback hell and how do you avoid it?
38. What is `setTimeout` and `setInterval`?
39. What is the purpose of `Promise.all()`?
40. How does JavaScript handle concurrency with single-threaded nature?

🔴 Advanced Concepts & DOM (41–50)

41. What is the DOM?
42. How do you manipulate DOM elements using JavaScript?
43. What are events and how do you handle them?
44. What is event delegation?
45. What is a closure and how is it used in practice?
46. What are IIFEs (Immediately Invoked Function Expressions)?
47. What is memory leak and how can it be prevented?
48. What is debouncing and throttling?
49. What is the difference between `localStorage`, `sessionStorage`, and `cookies`?

50. How is garbage collection handled in JavaScript?

Let me know if you'd like:

- 🔍 **Detailed answers with code examples**
- 📚 **Flashcards for revision**
- 👩‍💻 **Mock interview simulation**

Happy prepping!

Absolutely! Here are **40 basic coding-related JavaScript questions** that cover all the **foundational concepts**, such as variables, functions, arrays, loops, objects, strings, conditionals, and more.

🔹 1–10: Variables, Data Types, Operators

1. Declare a variable and assign your name to it.
2. What is the difference between `let`, `var`, and `const`?
3. What are primitive data types in JavaScript?
4. Convert a string `"123"` to a number.
5. Use the `typeof` operator to check the type of a variable.
6. What is the result of `true + 1`?
7. What does `NaN` mean? How do you check for it?
8. Write a function to swap two variables.
9. What is the output of `null == undefined`?
10. Explain the use of `===` vs `==` with examples.

🔹 11–20: Strings, Arrays, Loops

11. Write a function to reverse a string.
12. Check if a string is a palindrome.
13. Count the number of vowels in a string.
14. How do you split a string by spaces?
15. How do you join an array into a single string?
16. Write a function to find the largest number in an array.
17. Write a loop to print all even numbers from 1 to 10.
18. Use a `forEach` loop to print each element in an array.
19. Write a function to remove duplicates from an array.
20. Reverse an array without using `reverse()`.

🔹 21–30: Functions, Scope, Objects

21. What is a function? Write a function that adds two numbers.
22. What is the difference between parameters and arguments?
23. What is lexical scope?
24. Demonstrate how closures work in JavaScript.
25. What is an object? Create a `person` object with name and age.
26. How do you access and modify object properties?
27. How do you loop through an object's keys and values?
28. How do you delete a property from an object?
29. What is the difference between shallow and deep copy?
30. What is the output of this?

```js
const obj1 = { name: "Alice" };
const obj2 = obj1;
obj2.name = "Bob";
console.log(obj1.name);
```

🔹 31–40: Conditionals, Arrays, ES6 Features

31. Write a program to check if a number is odd or even.
32. Write a function that returns the factorial of a number.
33. What are arrow functions? Convert a function to an arrow function.
34. What is the spread operator? Show an example.
35. Use destructuring to extract values from an array.
36. Use `map()` to create a new array with each number squared.
37. Use `filter()` to find all numbers greater than 5 in an array.
38. Use `reduce()` to sum all numbers in an array.
39. Write a function that returns the number of occurrences of each character in a string.
40. Write a function to flatten a nested array.

These questions are great for:

- **Practice**
- **Concept checks**
- **Entry-level interviews**

Let me know if you want any of these:

- 🧠 **Quiz format**
- 📄 **PDF cheat sheet**
- ✅ **Solutions with code**