JavaScript 50 Q&A: let/const, Arrow Functions, Destructuring, Classes, Promises

Q1. [Interview] What is the difference between let and var?

Ans: let is block-scoped; var is function-scoped.

Q2. [Interview] Can const variables be reassigned?

Ans: No, but their objects can be mutated.

Q3. [Interview] What are arrow functions?

Ans: Concise syntax for functions that dont bind their own this.

Q4. [Interview] Can you use 'this' in arrow functions?

Ans: Yes, it takes the value from its lexical scope.

Q5. [Interview] What is destructuring in JavaScript?

Ans: Syntax to extract values from arrays or objects.

Q6. [Interview] What is a class in JavaScript?

Ans: ES6 syntax for constructor-based object creation.

Q7. [Interview] How to define a class method?

Ans: Inside class using method syntax.

Q8. [Interview] What is a promise?

Ans: An object representing the eventual completion of an async task.

Q9. [Interview] How to resolve a promise?

Ans: Using resolve() in executor function.

Q10. [Interview] What is .then() used for?

Ans: To handle resolved value of promise.

Q11. [Interview] What is .catch() used for?

Ans: To handle rejected promises.

Q12. [Interview] What does async/await do?

Ans: Syntax for working with promises in a cleaner way.

Q13. [Interview] Can classes extend other classes?

Ans: Yes, using extends.

Q14. [Interview] What is the constructor in a class?

Ans: Special method that initializes class instances.

Q15. [Interview] Can arrow functions be used as constructors?

Ans: No, they cannot be used with new keyword.

Q16. [Practical] Use destructuring to extract name and age from an object.

Ans:

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

Q17. [Practical] Create a class Car with method drive().

Ans:

```
class Car {
  drive() { console.log('Driving'); }
}
```

Q18. [Practical] Return first element using array destructuring.

Ans:

```
const [first] = [1,2,3];
```

Q19. [Practical] Write a promise that resolves after 1 second.

Ans:

```
new Promise(res => setTimeout(() => res('done'), 1000));
```

Q20. [Practical] Arrow function to square a number.

Ans:

```
const square = n => n * n;
```

Q21. [Practical] Destructure nested object.

Ans:

```
const user = { profile: { name: 'A' } };
const { profile: { name } } = user;
```

Q22. [Practical] Fetch data using async/await.

Ans:

Q23. [Practical] Write a method inside a class to log 'Hello'.

Ans:

Q24. [Practical] Create a constant array and modify one value.

Ans:

Q25. [Practical] Use let inside for loop and observe scope.

Ans:

Q26. [Practical] Chain .then() to a fetch() call.

Ans:

Q27. [Practical] Use catch() to log fetch error.

Ans:

Q28. [Practical] Create arrow function with multiple parameters.

Ans:

Q29. [Practical] Use default values in destructuring.

Ans:

Q30. [Practical] Return max number using rest and spread.

Ans:

Q31. [LeetCode] typeof Promise

Ans: 'function'

Q32. [LeetCode] What does const x = {}; x = 5 do?

Ans: Throws error.

Q33. [LeetCode] Arrow function 'this' binds to?

Ans: Lexical scope.

Q34. [LeetCode] Can you reassign a let variable?

Ans: Yes.

Q35. [LeetCode] [a,b] = [1,2]; value of b?

Ans: 2

Q36. [LeetCode] new Car() calls which method?

Ans: constructor

Q37. [LeetCode] Promise.resolve(1).then(x => x)

Ans: 1

Q38. [LeetCode] typeof class MyClass {}

Ans: 'function'

Q39. [LeetCode] Can const array be mutated?

Ans: Yes.

Q40. [LeetCode] let x = 10; { let x = 20; } x = ?

Ans: 10

Q41. [Conceptual] Why can't const be reassigned?

Ans: Const is immutable binding.

Q42. [Conceptual] Do classes hoist like functions?

Ans: No.

Q43. [Conceptual] Is destructuring mandatory?

Ans: No, it's syntactic sugar.

Q44. [Conceptual] Are arrow functions always better?

Ans: Not when you need 'this'.

Q45. [Conceptual] Can you mix var and let?

Ans: Yes, but avoid it.

Q46. [Conceptual] What does await pause?

Ans: Execution of async function.

Q47. [Conceptual] Are promises synchronous?

Ans: No.

Q48. [Conceptual] Can you use await outside async?

Ans: No.

Q49. [Conceptual] Why prefer arrow for callbacks?

Ans: Short and no own 'this'.

Q50. [Conceptual] Can methods be defined outside class?

Ans: Yes, with prototype.