25 Advanced JavaScript Questions

1. Question: What is JavaScript and what are its key features?

Answer:

JavaScript is a high-level, interpreted programming language primarily used for web development. Key features include:

Dynamically typed.

Supports both procedural and object-oriented programming.

Runs on the client side (web browsers).

Asynchronous programming with callbacks and Promises.

2. Question: Explain the difference between var, let, and const in JavaScript.

Answer:

var: Function-scoped, can be redeclared, and is hoisted.

let: Block-scoped, can be reassigned, and is hoisted.

const: Block-scoped, cannot be reassigned after declaration, and is hoisted.

3. Question: What is the significance of closures in JavaScript?

Answer:

Closures allow functions to retain access to variables from their outer (enclosing) scope even after the outer function has finished executing. They are crucial for creating private variables and maintaining state.

4. Question: Explain the event delegation in JavaScript.

Answer:

Event delegation is a technique where a single event listener is attached to a common ancestor instead of individual elements. It takes advantage of event bubbling, reducing the number of event listeners and improving performance.

5. Question: What is the purpose of the this keyword in JavaScript?

Answer:

this refers to the object that is currently executing the function. Its value is determined by how a function is called, and it allows access to the object's properties and methods.

6. Question: Describe the concept of prototypal inheritance in JavaScript.

Answer:

JavaScript uses prototypal inheritance, where objects can inherit properties and methods from other objects via a prototype chain. Each object has a prototype object, and if a property is not found in the object, JavaScript looks up the chain until it finds the property or reaches the end.

7. Question: What is a closure and provide an example?

Answer:

A closure is a function that has access to variables from its outer (enclosing) scope, even after the outer function has finished executing. Example:

```
function outer() {
  let outerVar = 10;

  function inner() {
    console.log(outerVar);
  }
```

```
  return inner;
}

const closureFunc = outer();
closureFunc(); // Outputs: 10
```

In this example, inner forms a closure with access to outerVar.

8. Question: What is the difference between null and undefined in JavaScript?

Answer:

null: A deliberate assignment indicating the absence of a value.
undefined: A variable that has been declared but not assigned a value, or a non-existent property in an object.

9. Question: Explain the purpose of the bind method in JavaScript.

Answer:
The bind method creates a new function that, when called, has its this keyword set to a specific value. It is often used to create functions with a fixed this value.

10. Question: What is the difference between == and === in JavaScript?

Answer:

==: Loose equality operator, only checks for equality of values after type coercion.
===: Strict equality operator, checks for equality of values and types without type coercion.

11. Question: Explain the purpose of the map function in JavaScript.

Answer:
The map function is used to transform each element of an array and create a new array with the results. It does not modify the original array.

```
const numbers = [1, 2, 3];
const squaredNumbers = numbers.map(num => num * num);
// Result: [1, 4, 9]
```

12. Question: What is the purpose of the async and await keywords in JavaScript?

Answer:

async is used to declare an asynchronous function, and await is used to pause the execution of an async function until the promise is resolved, returning the resolved value.

```
async function fetchData() {
  const result = await fetch('https://example.com');
  const data = await result.json();
  console.log(data);
}
```

13. Question: Explain the concept of hoisting in JavaScript.

Answer:

Hoisting is a JavaScript behavior where variable and function declarations are moved to the top of their containing scope during compilation. However, only the declarations are hoisted, not the initializations.

14. Question: What is the purpose of the reduce method in JavaScript?

Answer:

The reduce method is used to accumulate values in an array and reduce it to a single value. It takes a callback function that performs the accumulation.

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((acc, num) => acc + num, 0);
// Result: 10
```

15. Question: How does event delegation work in JavaScript?

Answer:

Event delegation involves attaching a single event listener to a common ancestor, rather than attaching multiple listeners to individual elements. It leverages event bubbling, allowing the handling of events on descendant elements through a single listener.

16. Question: Explain the purpose of the localStorage in JavaScript.

Answer:

localStorage is a web storage object that allows developers to store key/value pairs in a web browser with no expiration time. The stored data persists even when the browser is closed and reopened.

```
// Storing data
localStorage.setItem('username', 'John');

// Retrieving data
```

```
const username = localStorage.getItem('username');
console.log(username); // Outputs: John
```

17. Question: How do you handle errors in JavaScript?

Answer:

Errors in JavaScript can be handled using try, catch, finally blocks.

```
try {
  // Code that might throw an error
  throw new Error('An error occurred');
} catch (error) {
  console.error(error.message);
} finally {
  // Code that always runs
}
```

18. Question: What is the purpose of the Promise object in JavaScript?

Answer:

Promise is an object representing the eventual completion or failure of an asynchronous operation and its resulting value. It allows better handling of asynchronous operations, especially with async/await.

19. Question: How does the event loop work in JavaScript?

Answer:

The event loop is a mechanism in JavaScript that allows the execution of code to be non-blocking. It consists of a call stack, callback queue, and event loop. The call stack processes synchronous code, while asynchronous code is handled through callback functions pushed to the callback queue by web APIs.

20. Question: Explain the concept of arrow functions in JavaScript.

Answer:

Arrow functions provide a concise syntax for writing function expressions. They do not have their own this and arguments and are not suitable for functions that require these features.

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

21. Question: What is the purpose of the Object.create() method in JavaScript?

Answer:

Object.create() is used to create a new object with a specified prototype object. It allows for prototypal inheritance.

```
const person = {
  greet: function() {
    console.log('Hello!');
  }
};

const john = Object.create(person);
john.greet(); // Outputs: Hello!
```

22. Question: Explain the concept of the event bubbling and capturing phases.

Answer:

Event propagation in the DOM occurs in two phases: capturing phase (top-down) and bubbling phase (bottom-up). Event listeners can be placed in either phase to handle events as they propagate through the DOM.

23. Question: What is a RESTful API, and how does it work in JavaScript?

Answer:

A RESTful API (Representational State Transfer) is an architectural style for designing networked applications. It uses standard HTTP methods (GET, POST, PUT,

DELETE) for communication and is stateless. In JavaScript, the fetch API is often used to interact with RESTful APIs.

24. Question: What is the purpose of the setTimeout function in JavaScript?

Answer:

setTimeout is used to delay the execution of a function by a specified number of milliseconds.

```
console.log('Start');
setTimeout(() => {
  console.log('Delayed');
}, 1000);
console.log('End');
```

Output:

```
Start
End
Delayed
```

25. Question: How does the typeof operator work in JavaScript?

Answer:

The typeof operator returns a string indicating the type of an operand. It is often used to check the type of a variable.

```
console.log(typeof 42); // Outputs: 'number'
console.log(typeof 'Hello'); // Outputs: 'string'
console.log(typeof true); // Outputs: 'boolean'
```