JAVASCRIPT INTERVIEW QUESTIONS WITH ANSWERS

BASIC :

1. What is JavaScript?

JavaScript is a high-level, dynamic programming language used to create interactive and dynamic content on the web. It is a core technology of the web, along with HTML and CSS.

2. **Explain the difference between let, var, and const.**

var is function-scoped, let and const are block-scoped. let allows variable reassignment, while const does not allow reassignment once the value is set.

3. **What are the data types in JavaScript?**

The main data types are: String, Number, Boolean, Null, Undefined, Symbol, BigInt, and Object (which includes arrays, functions, etc.).

4. **How do you create a function in JavaScript?**

A function can be created using the function keyword:

Example :

```
function myFunction() {
   // code to be executed
}
```

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

== checks for value equality with type coercion, while === checks for both value and type equality without type coercion.

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

Template literals are a way to include variables and expressions within strings, using backticks () and ${}`

syntax:

```
const name = "John";

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

7. **Explain how to use Array.map() method.**

Array.map() creates a new array by applying a function to each element of the original array:

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

const doubled = numbers.map(n => n * 2);
```

8. **What is an anonymous function in JavaScript?**

An anonymous function is a function without a name. It is often used as a callback function:

```
setTimeout(function() {
   console.log("This is an anonymous function");
```

}, 1000);

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

undefined means a variable has been declared but not assigned a value. null is an assignment value representing no value.

10. **What is the purpose of the this keyword?**

- this refers to the object it belongs to. In a method, this refers to the owner object. In the global context, this refers to the global object.

INTERMEDIATE :

11. **Explain closures in JavaScript.**

A closure is a function that retains access to its outer scope even after the outer function has returned. This allows functions to maintain access to variables defined in their lexical scope:

```
function outerFunction() {
   let outerVar = 'I am outside!';
   return function innerFunction() {
      console.log(outerVar); // 'I am outside!'
   };
}
```

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

Event delegation is a technique of handling events by using a single event listener to manage all events of a particular type within a parent element, instead of adding multiple event listeners to individual child elements.

13. **How does the prototype chain work in JavaScript?**

Every JavaScript object has a prototype. When trying to access a property or method on an object, JavaScript will first look on the object itself, then up the prototype chain until it finds the property or reaches the top of the chain (Object.prototype).

14. **What is the difference between call(), apply(), and bind()?**

call() and apply() invoke a function with a specified this context, where call() accepts arguments individually, and apply() accepts them as an array. bind() returns a new function, where this is bound to the specified context.

15. **How does asynchronous JavaScript work? Explain promises.**

Asynchronous JavaScript allows code to run without blocking the main thread. Promises are a way to handle asynchronous operations, representing a value that may be available now, in the future, or never:

```
const promise = new Promise((resolve, reject) => {
  // async operation
  if (success) resolve(result);
  else reject(error);
```

```
});
```

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

The event loop is a mechanism that handles the execution of code, collects and processes events, and executes queued sub-tasks (like callbacks). It continuously checks the call stack and the task queue to manage asynchronous operations.

17. **Explain how async and await work in JavaScript.**

async functions return a promise. await pauses the execution of an async function until the promise resolves:

```
async function fetchData() {
  const response = await fetch('url');
  const data = await response.json();
  return data;
}
```

18. **What is the difference between forEach and map methods?**

forEach() executes a function on each element of an array but does not return a new array. map() also executes a function on each element but returns a new array with the transformed elements.

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

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

Example :

```
try {
   // code that may throw an error
} catch (error) {
   // code to handle the error
} finally {
   // code to run regardless of the outcome
}
```

20. **What is a higher-order function in JavaScript?**

A higher-order function is a function that takes one or more functions as arguments or returns a function as its result:

Example :

```
function higherOrder(fn) {
   return function() {
      return fn() + 1;
   };
}
```

ADVANCED :

21. **What is the purpose of Object.create() method?**

Object.create() creates a new object with the specified prototype object and properties. It allows for more precise control over the inheritance model:

Example :

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

const obj = Object.create(proto);
```

22. **How does JavaScript's garbage collection work?**

JavaScript uses automatic garbage collection to manage memory. It identifies and removes objects that are no longer reachable from the root (global object), typically using algorithms like mark-and-sweep.

23. **What are generators in JavaScript, and how do they work?**

Generators are functions that can be paused and resumed. They are declared with an asterisk (*) and use the yield keyword to pause:

```
function* generatorFunction() {
   yield 'First';
   yield 'Second';
}
```

24. **Explain the concept of currying in JavaScript.**

Currying is a technique where a function is transformed into a sequence of functions, each taking a single argument:

```
function curry(a) {
   return function(b) {
      return function(c) {
         return a + b + c;
      };
   };
}
```

25. **What is memoization, and how can it be implemented in JavaScript?**

Memoization is an optimization technique where the results of expensive function calls are cached and returned when the same inputs occur again:

Example :

```
function memoize(fn) {
   const cache = {};
   return function(...args) {
      const key = JSON.stringify(args);
      if (cache[key]) return cache[key];
      const result = fn(...args);
      cache[key] = result;
      return result;
   };
```

}

26. **How do you optimize the performance of a JavaScript application?**

Techniques include minimizing DOM access, using debounce and throttle for event handlers, optimizing loops, lazy loading, and using Web Workers for CPU-intensive tasks.

27. **Explain the difference between deep and shallow copy.**

A shallow copy duplicates an object's top-level properties, while a deep copy recursively copies all nested objects, ensuring complete duplication:

Example :

```
const shallowCopy = Object.assign({}, obj); // shallow

const deepCopy = JSON.parse(JSON.stringify(obj)); // deep
```

28. **What are WeakMap and WeakSet in JavaScript?**

WeakMap and WeakSet are collections that hold weak references to their keys or elements, meaning they do not prevent garbage collection. They are useful for memory-sensitive applications.

29. **How do modules work in JavaScript? Explain the difference between require and import.**

Modules allow code to be divided into reusable pieces. require is used in CommonJS (Node.js), while import is used in ES6 modules. import is static and must be used at the top of the file, while require is dynamic.

30. **What is the significance of the Symbol type in JavaScript?**

Symbol is a primitive data type that creates a unique, immutable identifier, often used as keys in objects to avoid property name collisions.