**JavaScript**

**1. What is JavaScript?**

- Answer: JavaScript is a high-level, interpreted programming language that is widely used for front-end web development.

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

- Answer: `let` and `const` are block-scoped, while `var` is function-scoped. `const` is used for constants, and `let` is for variables with reassignment.

- Example:

let variable1 = 10;

const constantValue = 20;

var oldVariable = 5;

**3. What is the purpose of the `===` operator in JavaScript?**

- Answer: The `===` operator checks for strict equality without type coercion.

- Example:

javascript

if (x === 5) {

// Code block executes only if x is exactly equal to 5

}

**3.1 Difference between “==” and “===”?**

“==” checks only for equality in value, whereas “===” is a stricter equality test and returns false if either the value or the type of the two variables are different.

**4. Explain the concept of closures in JavaScript.**

- Answer: Closures occur when a function is defined inside another function, allowing access to the outer function's variables even after the outer function has finished execution.

- Example:

javascript

function outerFunction() {

let outerVariable = 10;

function innerFunction() {

console.log(outerVariable);

}

return innerFunction;

}

const closure = outerFunction();

closure(); // Outputs 10

**5. What is the purpose of the `this` keyword in JavaScript?**

- Answer: The `this` keyword refers to the current execution context and can vary depending on how a function is called.

- Example:

javascript

const obj = {

property: 'value',

logProperty: function() {

console.log(this.property);

}

};

obj.logProperty(); // Outputs 'value'

**6. Explain the concept of prototypal inheritance in JavaScript.**

- Answer: In JavaScript, objects can inherit properties and methods from other objects through their prototype chain.

- Example:

javascript

function Animal(name) {

this.name = name;

}

Animal.prototype.sayName = function() {

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

};

const cat = new Animal('Whiskers');

cat.sayName(); // Outputs 'My name is Whiskers'

**7. What is the purpose of the `async` and `await` keywords in JavaScript?**

- Answer: `async` and `await` are used to work with asynchronous code more easily, making it appear more synchronous.

- Example:

async function fetchData() {

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

const data = await response.json();

console.log(data);

}

fetchData();

**8. How can you handle errors in JavaScript?**

- Answer: Use `try...catch` to handle exceptions.

- Example

try {

// Code that might throw an exception

} catch (error) {

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

}

**9. Explain the concept of event delegation in JavaScript.**

- Answer: Event delegation involves attaching a single event listener to a parent element rather than individual child elements, allowing the handling of events for multiple elements efficiently.

- Example:

javascript

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

if (event.target.tagName === 'BUTTON') {

console.log('Button clicked:', event.target.textContent);

}

});

**10. What is the purpose of the `map` function in JavaScript?**

- Answer: The `map` function is used to transform elements of an array and create a new array with the results.

- Example:

javascript

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

const squaredNumbers = numbers.map(function(num) {

return num \* num;

});

console.log(squaredNumbers); // Outputs [1, 4, 9, 16, 25]

**11. How can you add an element to the beginning and end of an array in JavaScript?**

- Answer: Use `unshift` to add to the beginning and `push` to add to the end.

- Example:

javascript

const array = [2, 3, 4];

array.unshift(1); // Adds 1 to the beginning

array.push(5); // Adds 5 to the end

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

**12. Explain the purpose of the `JSON.stringify` and `JSON.parse` functions.**

- Answer: `JSON.stringify` converts a JavaScript object to a JSON string, and `JSON.parse` converts a JSON string back to a JavaScript object.

- Example:

javascript

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

const jsonString = JSON.stringify(person);

const parsedObject = JSON.parse(jsonString);

**13. What is the purpose of the `localStorage` and `sessionStorage` objects in JavaScript?**

- Answer: `localStorage` and `sessionStorage` are used to store key-value pairs in the web browser. `localStorage` persists even after the browser is closed, while `sessionStorage` is cleared when the session ends.

- Example:

javascript

localStorage.setItem('key', 'value');

const storedValue = localStorage.getItem('key');

**14. How do you make an AJAX call in JavaScript?**

- Answer: Use the `XMLHttpRequest` object or the more modern `fetch` API.

- Example using `fetch`:

javascript

fetch('https://api.example.com/data')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

**15. Explain the concept of promises in JavaScript.**

- Answer: Promises represent the eventual completion or failure of an asynchronous operation and allow chaining of actions that should be taken after the operation is complete.

- Example:

javascript

const fetchData = new Promise((resolve, reject) => {

// Asynchronous operation

if (success) {

resolve(data);

} else {

reject('Error occurred');

}

});

fetchData.then(data => console.log(data)).catch(error => console.error(error));

**16. What is the purpose of the `addEventListener` method in JavaScript?**

- Answer: `addEventListener` is used to attach an event handler function to an HTML element, allowing you to respond to events like clicks, keypresses, etc.

- Example:

javascript

const button = document.getElementById('

myButton');

button.addEventListener('click', function() {

console.log('Button clicked!');

});

**17. How do you create and manipulate elements in the DOM using JavaScript?**

- Answer: Use methods like `createElement`, `appendChild`, and `setAttribute` to create and modify HTML elements.

- Example:

javascript

const newDiv = document.createElement('div');

newDiv.textContent = 'Hello, World!';

document.body.appendChild(newDiv);

**18. What is the purpose of the `setInterval` function in JavaScript?**

- Answer: `setInterval` is used to repeatedly execute a function at specified intervals.

- Example:

javascript

setInterval(function() {

console.log('Executing at regular intervals');

}, 1000); // Executes every 1000 milliseconds (1 second)

**19. Explain the concept of the Event Loop in JavaScript.**

- Answer: The Event Loop is a mechanism in JavaScript that handles asynchronous operations by managing the execution of the call stack, callback queue, and microtask queue.

- Example:

javascript

console.log('Start');

setTimeout(function() {

console.log('Timeout');

}, 0);

console.log('End');

**20. What is the purpose of the `typeof` operator in JavaScript?**

- Answer: The `typeof` operator is used to determine the data type of a variable.

- Example:

javascript

const x = 5;

console.log(typeof x); // Outputs 'number'

**21. How can you prevent the default behavior of an event in JavaScript?**

- Answer: Use the `event.preventDefault()` method.

- Example:

javascript

const link = document.getElementById('myLink');

link.addEventListener('click', function(event) {

event.preventDefault();

console.log('Link clicked, but default behavior prevented.');

});

**22. What is the purpose of the `try...catch` statement in JavaScript?**

- Answer: The `try...catch` statement is used to handle errors by executing code in a `try` block and catching and handling any exceptions in the `catch` block.

- Example:

javascript

try {

// Code that might throw an exception

} catch (error) {

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

}

**23. How can you dynamically change the style of an element in JavaScript?**

- Answer: Use the `style` property of the element.

- Example:

javascript

const myElement = document.getElementById('myElement');

myElement.style.color = 'red';

**24. What is the purpose of the `Array.isArray` method in JavaScript?**

- Answer: `Array.isArray` checks if a value is an array.

- Example:

javascript

const arr = [1, 2, 3];

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

**25. How can you check if an object has a specific property in JavaScript?**

- Answer: Use the `hasOwnProperty` method or the `in` operator.

- Example:

javascript

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

console.log(myObject.hasOwnProperty('name')); // Outputs true

**26. Explain the concept of arrow functions in JavaScript.**

- Answer: Arrow functions provide a concise syntax for writing function expressions and lexically bind the `this` value.

- Example:

javascript

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

**27. What is the purpose of the `localStorage` object in JavaScript?**

- Answer: The `localStorage` object is used to store key-value pairs in the web browser persistently.

- Example:

javascript

localStorage.setItem('key', 'value');

const storedValue = localStorage.getItem('key');

**28. How can you iterate over the properties of an object in JavaScript?**

- Answer: Use a `for...in` loop or `Object.keys()` to iterate over an object's properties.

- Example:

javascript

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

for (const key in myObject) {

console.log(`${key}: ${myObject[key]}`);

}

**29. What is the purpose of the `splice` method in JavaScript?**

- Answer: The `splice` method is used to change the contents of an array by removing or replacing existing elements and/or adding new elements in place.

- Example:

javascript

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

numbers.splice(2, 1); // Removes 1 element starting at index 2

**30. Explain the concept of the Single Responsibility Principle in the context of JavaScript.**

- Answer: The Single Responsibility Principle (SRP) states that a function or module should have only one reason to change, meaning it should only have one responsibility or job.

-