100 JavaScript

Questions *with Explanations*

🚀 Test your skills how many can you answer 🚀

---

1. What is the output of the following code?

```
console.log(typeof typeof 1);
```

- A) "number"
- B) "string"
- C) "object"
- D) "undefined"

Correct Answer: B) "string"

Explanation: The inner typeof 1 returns "number", which is a string. Then, the outer typeof "number" returns "string", because typeof any string is "string".

2. Which statement about JavaScript closures is true?

- A) A closure is a function that is evaluated in an execution context that has multiple bindings.
- B) Closures are only used for creating modules.
- C) A closure gives you access to an outer function’s scope from an inner function.
- D) Closures cannot access variables defined outside of their scope.

Correct Answer: C) A closure gives you access to an outer function’s scope from an inner function.

Explanation: Closures are functions that refer to independent (free) variables. In other words, the function defined in the closure 'remembers' the environment in which it was created.

3. What does the Promise.all method do?

- A) Takes an iterable of promises and returns a single Promise that resolves when all of the promises resolve.
- B) Takes an iterable of promises and returns a single Promise that rejects as soon as one of the promises rejects.
- C) Takes two promises and returns a new promise that resolves when both input promises resolve.
- D) Executes all promises in a synchronous manner, one after the other.

Correct Answer: A) Takes an iterable of promises and returns a single Promise that resolves when all of the promises resolve.

Explanation: Promise.all is used for executing multiple promises in parallel and waits for all of them to complete. It returns a single Promise that resolves when all of the promises in the iterable have resolved or when the iterable contains no promises.

4. How does the async/await syntax help with asynchronous code in JavaScript?

- A) It automatically optimizes the code to run synchronously.
- B) It pauses the execution of async functions until the awaited Promise is resolved or rejected.
- C) It prevents any form of asynchronous operation.
- D) It forces all functions to return a Promise.

Correct Answer: B) It pauses the execution of async functions until the awaited Promise is resolved or rejected.

Explanation: The async/await syntax makes asynchronous code look and behave a little more like synchronous code, which helps in writing clearer style asynchronous code.

5. What will the following code output to the console?

```
async function asyncFunc() {
 return 123;
}

asyncFunc().then(console.log);
```

- A) 123
- B) undefined
- C) A Promise object
- D) An error message

Correct Answer: A) 123

Explanation: When an async function is called, it returns a Promise. If the function returns a value, the Promise will be resolved with the value. In this case, asyncFunc returns 123, so the Promise resolves with 123, and that is what is logged.

6. Which of the following is true about the JavaScript event loop?

- A) The event loop allows JavaScript to perform non-blocking I/O operations, despite being single-threaded.
- B) The event loop can execute multiple JavaScript chunks simultaneously.
- C) The event loop is exclusive to Node.js and does not exist in browser environments.
- D) JavaScript code is executed in a random order, decided by the event loop.

Correct Answer: A) The event loop allows JavaScript to perform non-blocking I/O operations, despite being single-threaded.

Explanation: The event loop is a mechanism that allows JavaScript to perform non-blocking I/O operations by offloading operations to the system kernel whenever possible.

7. What is prototype inheritance in JavaScript?

- A) A method for creating a new class from an existing class.
- B) A process where objects can directly inherit properties and methods from other objects.
- C) A feature that allows creating private variables.
- D) A way to attach new properties to the Function prototype.

Correct Answer: B) A process where objects can directly inherit properties and methods from other objects.

Explanation: Prototype inheritance allows objects to inherit features from other objects. Every object in JavaScript has a built-in property, which is a link to its prototype.

8. Consider the following code snippet. What will it print?

```
let x = {}, y = {name: "Ronny"}, z = {name: "John"};
x[y] = {name: "Vivek"};
x[z] = {name: "Akki"};
console.log(x[y]);
```

- A) {name: "Vivek"}
- B) {name: "Akki"}
- C) {name: "Ronny"}
- D) undefined

Correct Answer: B) {name: "Akki"}

Explanation: When you use objects as keys, they are converted to strings. Both y and z objects will be converted to "[object Object]" string. Therefore, x[z] overwrites x[y], and the final output is {name: "Akki"}.

9. Which one of the following correctly shows how to create a deep copy of an object in JavaScript?

- A) let objCopy = Object.assign({}, obj);
- B) let objCopy = {...obj};
- C) let objCopy = JSON.parse(JSON.stringify(obj));
- D) let objCopy = obj.slice();

Correct Answer: C) let objCopy = JSON.parse(JSON.stringify(obj));

Explanation: JSON.parse(JSON.stringify(obj)) creates a deep copy of the object, but it does not work with functions, undefined, or circular references within the object. Object.assign and the spread operator create shallow copies.

10. What is the result of the following expression?

0.1 + 0.2 === 0.3

- A) true
- B) false
- C) It throws a TypeError.
- D) It depends on the JavaScript engine.

Correct Answer: B) false

Explanation: Due to floating-point arithmetic in JavaScript (and most other programming languages), adding 0.1 and 0.2 does not exactly equal 0.3 because of precision errors in the way numbers are stored in memory.

11. What is the purpose of the Set object in JavaScript?

- A) To store a collection of any type of values, in a linear structure similar to an array.
- B) To store unique values of any type, whether primitive values or object references.
- C) To store key-value pairs for any type of values.
- D) To execute a function on each item in the array.

Correct Answer: B) To store unique values of any type, whether primitive values or object references.

Explanation: A Set is a collection of values where each value may occur only once. It's a perfect structure for ensuring uniqueness of elements and supports both primitive values and object references.

12. Which of the following is a true statement about JavaScript modules?

- A) JavaScript modules are executed in the global scope.
- B) JavaScript modules allow you to create private and public enclosures easily.
- C) In a module, variables declared with var are scoped to the module.
- D) Modules can only export one function or variable.

Correct Answer: B) JavaScript modules allow you to create private and public enclosures easily.

Explanation: JavaScript modules allow for easier management of dependencies and can contain both public and private code. Unlike scripts, modules are executed in their own scope, not the global scope, allowing for encapsulation.

13. What will the following code snippet output?

```
function* generatorFunction() {
 yield 1;
 yield 2;
```

```
 yield 3;
}

const generator = generatorFunction();
console.log(generator.next().value);
console.log(generator.next().value);
console.log(generator.next().value);
console.log(generator.next().value);
```

- A) 1 2 3 undefined
- B) 1 2 3 4
- C) 1 1 1 1
- D) Error

Correct Answer: A) 1 2 3 undefined

Explanation: Generator functions allow you to define an iterative algorithm by writing a single function whose execution is not continuous. generator.next().value returns the next value yielded by the generator. After yielding all values, calling next() again will return undefined.

14. How do you ensure that a function does not get called more often than a specified period, even if it is invoked multiple times?

- A) Callback
- B) Throttling
- C) Debouncing
- D) Promises

Correct Answer: B) Throttling

Explanation: Throttling is a technique used to ensure that a function is called at most once in a specified period. It's useful for controlling the rate at which a function is executed.

15. Which statement about async functions is NOT true?

- A) They always return a promise.
- B) They can use the await keyword inside.
- C) They can be used with the new operator to create instances.
- D) They make handling asynchronous operations more synchronous in style.

Correct Answer: C) They can be used with the new operator to create instances.

Explanation: async functions are designed to handle asynchronous operations. They cannot be used with the new operator because they are not constructors and are meant to return promises, not instantiate objects.

16. What is the default behavior of the import statement in JavaScript if the specified module does not export a default?

- A) It will return undefined.
- B) It will throw an error.
- C) It will import all exported members as an object.
- D) It will automatically create a default export.

Correct Answer: B) It will throw an error.

Explanation: If you try to import a default export from a module that doesn't export one, JavaScript will throw an error because the expected export is not found.

17. In the context of the this keyword in JavaScript, what will the following code output?

```
const obj = {
 name: 'JavaScript',
 getName: function() {
 return this.name;
```

```
 }
};

const getName = obj.getName;
console.log(getName());
```

- A) 'JavaScript'
- B) undefined
- C) null
- D) ''

Correct Answer: B) undefined

Explanation: When getName is called, it is no longer part of obj, so this does not refer to obj anymore. In a non-strict mode, this refers to the global object (window in browsers), where name is not defined, hence undefined. In strict mode, this would be undefined, leading to an error when trying to access name.

18. What is the output of this code?

```
let num = 0;
async function increment() {
 num += await 2;
 console.log(num);
}
increment();
num += 1;
console.log(num);
```

- A) 1 3
- B) 3 1
- C) 2 3
- D) 1 2

Correct Answer: A) 1 3

Explanation: The await expression causes increment to pause until a Promise is resolved. During this pause, the synchronous code continues to execute, incrementing num by 1, making it 1 before printing. After the await resolves, num is incremented by 2, leading to 3 when logged inside the async function.

19. What is the purpose of the Symbol type in JavaScript?

- A) To create completely unique identifiers.
- B) To add symbolic properties to objects.
- C) To symbolize numeric constants.
- D) To provide symbols for iterator methods.

Correct Answer: A) To create completely unique identifiers.

Explanation: The Symbol type allows for the creation of unique identifiers. Symbols are immutable and can be used as object properties to avoid name collisions, among other uses.

20. What does the finally block in a try-catch-finally statement do?

- A) It is executed if an error occurs in the try block.
- B) It is executed after the try and catch blocks, regardless of whether an exception was thrown or caught.
- C) It replaces the catch block if no errors are thrown.
- D) It executes only if no errors occur in the try block.

Correct Answer: B) It is executed after the try and catch blocks, regardless of whether an exception was thrown or caught.

Explanation: The finally block is executed after the try and catch blocks complete, regardless of whether an exception was thrown or not. It is typically used for cleanup actions.

21. What does the following code snippet demonstrate?

```
const arr = [10, 12, 15, 21];
for (let i = 0; i < arr.length; i++) {
 setTimeout(function() {
 console.log('Index: ' + i + ', element: ' + arr[i]);
 }, 3000);
}
```

- A) Closure
- B) Event Loop
- C) Hoisting
- D) Callback Queue

Correct Answer: A) Closure

Explanation: This code snippet demonstrates a closure, where the anonymous function passed to setTimeout captures the environment in which it was created, including the current value of i for each iteration of the loop. Thanks to the use of let (which is block-scoped), each loop iteration has its own instance of i, ensuring the output reflects the actual iteration index.

22. What is the outcome of using Object.freeze on an object?

- A) It makes all properties of the object read-only.
- B) It prevents the addition of new properties to the object.
- C) It prevents the modification of existing properties and the addition of new properties.
- D) It deletes all properties of the object over time.

Correct Answer: C) It prevents the modification of existing properties and the addition of new properties.

Explanation: Object.freeze makes an object immutable. Once an object is frozen, you cannot change its properties, add new properties, or remove

existing properties. However, it only works on the object's immediate properties; nested objects are not frozen.

23. How does JavaScript handle integer overflow?

- A) By throwing an error.
- B) By wrapping around to the minimum value.
- C) By using special overflow values.
- D) JavaScript numbers are floating point, so they do not overflow in the traditional sense.

Correct Answer: D) JavaScript numbers are floating point, so they do not overflow in the traditional sense.

Explanation: JavaScript uses double-precision 64-bit binary format IEEE 754 values for numbers, which means it does not have traditional integer overflow. Instead, when numbers exceed the representable range, they become Infinity or -Infinity.

24. What feature does the following code snippet illustrate?

```
const obj = {
 async getData() {
 const data = await fetch('https://api.example.com/data');
 return data.json();
 }
};
```

- A) Promises
- B) Callbacks
- C) Async/Await in object methods
- D) Event delegation

Correct Answer: C) Async/Await in object methods

Explanation: This code snippet demonstrates the use of async/await within an object method to handle asynchronous operations, making the syntax for working with promises more readable and expressive.

25. What is a potential drawback of using the == operator for equality in JavaScript?

- A) It can lead to type coercion.
- B) It is slower than the === operator.
- C) It cannot compare string values.
- D) It always returns false when comparing different types.

Correct Answer: A) It can lead to type coercion.

Explanation: The == operator performs type coercion if the types on either side are different before comparing, which can lead to unexpected results. The === operator, in contrast, does not perform type coercion and checks for both type and value equality.

26. Which statement accurately describes the behavior of the null and undefined values in JavaScript?

- A) null is an object, and undefined is a type of its own.
- B) Both null and undefined are types of objects.
- C) null and undefined are equivalent in type and value.
- D) null is an assignment value, indicating that a variable does not point to any object, while undefined indicates absence of value.

Correct Answer: D) null is an assignment value, indicating that a variable does not point to any object, while undefined indicates absence of value.

Explanation: null is used to represent the intentional absence of any object value, while undefined is used when a variable has been declared but has not yet been assigned a value. typeof null returns "object", which is a legacy bug in JavaScript, but null and undefined are distinct types.

27. In JavaScript, what is the main use case for the bind method?

- A) To fix the context of this within a function.
- B) To combine two or more functions.
- C) To delay the execution of a function.
- D) To repeat the execution of a function multiple times.

Correct Answer: A) To fix the context of this within a function.

Explanation: The bind method creates a new function that, when called, has its this keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is called.

28. What does the following expression evaluate to?

typeof NaN

- A) "number"
- B) "NaN"
- C) "undefined"
- D) "object"

Correct Answer: A) "number"

Explanation: NaN stands for "Not-a-Number," but paradoxically, its type is "number". This is because NaN is used to represent a computational error or an undefined result in numerical calculations.

29. What will the following code output?

```
console.log(0.1 + 0.2 === 0.3);
console.log(0.1 + 0.2);
```

- A) false, 0.30000000000000004
- B) true, 0.3
- C) false, 0.3

- D) true, 0.30000000000000004

Correct Answer: A) false, 0.30000000000000004

Explanation: Due to floating-point arithmetic errors, 0.1 + 0.2 does not exactly equal 0.3 in JavaScript. Instead, it results in a slightly off value like 0.30000000000000004, making the strict equality to 0.3 false.

30. What is the output of the following JavaScript code?

```
const set = new Set([1, 2, 3, 4, 5, 1, 2]);
console.log(set.size);
```

- A) 7
- B) 5
- C) Undefined
- D) Error

Correct Answer: B) 5

Explanation: A Set object in JavaScript stores unique values. When initializing a set with an array that contains duplicate values (1 and 2 in this case), those duplicates are removed. Thus, the size of the set is the count of unique values, which is 5.

31. Which option best describes the event delegation pattern in JavaScript?

- A) Attaching a single event listener to a parent element to handle events for multiple child elements.
- B) Assigning an event listener to each individual element you want to handle events for.
- C) Using multiple event listeners on a single element to handle different types of events.
- D) Delegating JavaScript execution to web workers for asynchronous event handling.

Correct Answer: A) Attaching a single event listener to a parent element to handle events for multiple child elements.

Explanation: Event delegation is a technique where you use a single event listener on a parent element to manage all of its child elements' events, taking advantage of the event bubbling phase. This approach is efficient and reduces the memory footprint by minimizing the number of event handlers needed.

32. What is the result of trying to extend a JavaScript object that has been frozen with Object.freeze()?

- A) The new properties will be added successfully.
- B) The operation will silently fail without adding new properties.
- C) An error will be thrown in strict mode.
- D) The object will be unfrozen, and new properties will be added.

Correct Answer: C) An error will be thrown in strict mode.

Explanation: When you attempt to extend (add new properties to) a frozen object in strict mode, JavaScript will throw a TypeError to indicate the operation is not allowed. In non-strict mode, the operation will fail silently without adding new properties.

33. How does the ?? operator function in JavaScript?

- A) It returns the left-hand operand if it is not null or undefined, otherwise it returns the right-hand operand.
- B) It merges two objects or arrays into a single object or array.
- C) It checks whether two values are both null or undefined.
- D) It performs a bitwise comparison of two values.

Correct Answer: A) It returns the left-hand operand if it is not null or undefined, otherwise it returns the right-hand operand.

Explanation: The nullish coalescing operator ?? is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand. It serves as a way to provide default values.

34. What does the following JavaScript code output?

console.log(`2` + `2` - `2`);

- A) 20
- B) 22
- C) 2
- D) "222"

Correct Answer: B) 22

Explanation: This code demonstrates how JavaScript handles type coercion with different operators. The plus (+) operator concatenates the two string operands into "22", then the minus (-) operator converts the string "22" into a number and subtracts 2, resulting in 20.

35. Which of these statements about WeakMaps in JavaScript is true?

- A) WeakMaps allow any type of value (both objects and primitive values) as keys.
- B) WeakMaps can have their keys enumerated.
- C) A WeakMap automatically removes entries when their keys are no longer referenced.
- D) WeakMaps have a clear() method that removes all entries.

Correct Answer: C) A WeakMap automatically removes entries when their keys are no longer referenced.

Explanation: WeakMaps hold "weak" references to key objects, which means that if there are no other references to an object used as a key in a WeakMap, that key-value pair will be eligible for garbage collection. This

feature is useful for managing memory in large applications. WeakMaps do not allow enumeration of their keys and do not have a clear() method.

36. What is the output of this JavaScript snippet?

```
function foo() {
 return
 {
 bar: "hello"
 };
}

console.log(foo());
```

- A) undefined
- B) { bar: "hello" }
- C) SyntaxError
- D) ReferenceError

Correct Answer: A) undefined

Explanation: JavaScript's automatic semicolon insertion (ASI) mechanism inserts a semicolon after the return statement because it's followed by a newline, which makes the function return undefined instead of the object. To return the object, the opening brace { should be on the same line as the return statement.

37. What feature do dynamic imports (import()) in JavaScript enable?

- A) Synchronous module loading
- B) Loading modules based on a condition
- C) Importing CSS files into JavaScript
- D) Declaring variables without var, let, or const

Correct Answer: B) Loading modules based on a condition

Explanation: Dynamic imports allow developers to load JavaScript modules dynamically as needed, rather than loading all scripts at the beginning. This can be based on conditions, user interactions, or any other programmatic situation, enabling more efficient resource loading and potentially improving performance by splitting code.

38. Which of these is a true statement about the JavaScript Map object?

- A) A Map object can only have strings as keys.
- B) The Map object maintains the insertion order of its elements.
- C) A Map object's size can be retrieved with the .length property.
- D) Map objects are always faster than plain objects for any operation.

Correct Answer: B) The Map object maintains the insertion order of its elements.

Explanation: Unlike plain objects, a Map remembers the original insertion order of the keys. This can be particularly useful when the order of elements is important for the application logic. The size of a Map is retrieved using the .size property, not .length.

39. What is the purpose of the Array.prototype.reduce method in JavaScript?

- A) To reduce all elements of the array to a single value, from left to right.
- B) To filter out values in an array based on a test function.
- C) To execute a function on each element in the array without changing the array.
- D) To find the first element in the array that satisfies a provided testing function.

Correct Answer: A) To reduce all elements of the array to a single value, from left to right.

Explanation: The reduce() method applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value. This is useful for summing up numbers, combining arrays, or any operation that needs to derive a single result from multiple elements.

40. How does the JavaScript runtime handle asynchronous operations such as setTimeout, Promise, and async/await?

- A) By using the call stack exclusively.
- B) By blocking the main thread until the operation completes.
- C) With a single-threaded event loop and a callback queue.
- D) Through multi-threading, creating a new thread for each operation.

Correct Answer: C) With a single-threaded event loop and a callback queue.

Explanation: JavaScript is single-threaded and uses an event loop to handle asynchronous operations. Functions like setTimeout or asynchronous operations initiated by Promises and async/await are handled outside the main thread's call stack. When these operations complete, they are moved to a callback queue and then executed by the main thread when the call stack is empty, allowing JavaScript to perform non-blocking operations.

41. What is the significance of using a DocumentFragment in DOM manipulation?

- A) It automatically optimizes memory usage by clearing unused DOM elements.
- B) It represents a minimal document object that has no parent.
- C) It allows batch insertion of DOM elements, improving performance.
- D) It provides a way to asynchronously load DOM elements.

Correct Answer: C) It allows batch insertion of DOM elements, improving performance.

Explanation: A DocumentFragment is a lightweight container that can hold temporary DOM elements. Adding elements to a DocumentFragment does not cause page reflow or repaint which occurs when elements are directly inserted into the DOM. This makes using DocumentFragment an efficient way to make multiple changes to the DOM with a single reflow/repaint cycle.

42. In JavaScript, how does the Proxy object enhance functionality?

- A) By enabling the creation of objects with a specified prototype.
- B) By allowing objects to define custom behavior for fundamental operations (e.g., property lookup, assignment, enumeration, function invocation).
- C) By providing a mechanism for encapsulating private properties and methods.
- D) By facilitating the communication between web workers and the main thread.

Correct Answer: B) By allowing objects to define custom behavior for fundamental operations (e.g., property lookup, assignment, enumeration, function invocation).

Explanation: The Proxy object is used to define custom behavior for fundamental operations on objects, such as property lookup, assignment, enumeration, and even function invocation. This makes it a powerful tool for creating smart proxies, validation, observing property changes, and more.

43. What is the output of the following code snippet?

```
let x = 10;
const promise = new Promise((resolve, reject) => {
 x = x + 5;
 resolve(x);
});
```

```
promise.then((val) => {
 console.log(val);
});
console.log(x);
```

- A) 10 followed by 15
- B) 15 followed by 15
- C) 15 followed by 10
- D) 10 followed by 10

Correct Answer: B) 15 followed by 15

Explanation: The promise executor function runs immediately when a new Promise is instantiated, which modifies x to 15 before the console.log(x) outside the promise chain is executed. Since JavaScript promises are asynchronous, console.log(x) outside then() is reached before the promise's then() method, but after x has been updated. Both logs print 15, showing the synchronous code's immediate effect and the asynchronous resolution of the promise.

44. What will the following code output to the console?

```
console.log(1);
setTimeout(() => { console.log(2); }, 0);
Promise.resolve().then(() => console.log(3));
console.log(4);
```

- A) 1, 2, 3, 4
- B) 1, 3, 4, 2
- C) 1, 4, 3, 2
- D) 1, 4, 2, 3

Correct Answer: C) 1, 4, 3, 2

Explanation: The code first logs 1 synchronously. setTimeout is asynchronous and is placed in the Web APIs, so console.log(2); is queued for future execution. Promise.resolve().then() queues console.log(3); in the microtask queue, which has higher priority than the callback queue used by setTimeout, making it execute right after the current call stack is clear but before any timer or I/O events. console.log(4); is executed next synchronously. Therefore, the correct execution order is 1, 4, 3, 2.

45. How can you ensure a function fetchData that returns a promise will retry up to 3 times upon failure before giving up?

- A) Use a loop to call fetchData inside a try/catch block up to 3 times.
- B) Use recursion and a counter variable to call fetchData again in the .catch() method if the counter is less than 3.
- C) Use the Promise.all method to call fetchData three times in parallel and return the first successful attempt.
- D) Modify fetchData to include a loop that attempts the operation up to 3 times before rejecting.

Correct Answer: B) Use recursion and a counter variable to call fetchData again in the .catch() method if the counter is less than 3.

Explanation: A recursive approach with a counter allows for the function to retry upon failure. Each time fetchData is called and fails, the .catch() handler can call it again, incrementing the counter each time. This process can repeat until the operation succeeds or the counter reaches 3, at which point the promise is either resolved (upon success) or rejected (after 3 failures), providing controlled retry logic.

46. What is the effect of using Object.seal on an object in JavaScript?

- A) It prevents any new properties from being added to the object and marks all existing properties as non-configurable.
- B) It completely freezes the object, preventing any changes to its properties.

- C) It allows the addition of new properties but prevents deletion of existing properties.
- D) It encrypts the object's properties to secure its data.

Correct Answer: A) It prevents any new properties from being added to the object and marks all existing properties as non-configurable.

Explanation: Object.seal prevents new properties from being added to an object and sets all existing properties to non-configurable, which means you cannot delete them. However, it still allows modification of existing property values (unless they are individually defined as read-only), unlike Object.freeze which prevents any changes to the object.

47. Which statement about template literals in JavaScript is true?

- A) They can only be used for string concatenation.
- B) They do not support embedded expressions.
- C) They can contain placeholders for embedding expressions.
- D) They are evaluated at compile-time.

Correct Answer: C) They can contain placeholders for embedding expressions.

Explanation: Template literals are enclosed by backticks (`) and can contain placeholders indicated by the dollar sign and curly braces (${expression}). These placeholders can include any JavaScript expression, including variables and operations, which makes template literals powerful for creating dynamic strings.

48. What is the purpose of the Array.prototype.flatMap method in JavaScript?

- A) To flatten an array up to a specified depth and map it using a provided function.
- B) To apply a function to each element of the array without altering the structure of the array.

- C) To map each element to a new array and concatenate the sub-arrays into one array.
- D) To flatten an array of arrays into a single array without applying any mapping function.

Correct Answer: C) To map each element to a new array and concatenate the sub-arrays into one array.

Explanation: The flatMap method first maps each element using a mapping function, then flattens the result into a new array. It is essentially equivalent to a map followed by a flat of depth 1, but flatMap is more efficient than running map and flat separately.

49. In the context of JavaScript's execution context, what is the global execution context?

- A) The context created for each function call.
- B) The environment in which JavaScript code is executed for the first time.
- C) A special execution context for event callbacks.
- D) The execution context for JavaScript running inside Web Workers.

Correct Answer: B) The environment in which JavaScript code is executed for the first time.

Explanation: The global execution context is the base level context in which JavaScript code starts its execution when loaded by the browser or a JavaScript engine. It is where top-level variables and functions reside.

50. What will the following JavaScript code output?

```
async function checkData() {
 return await Promise.resolve('Resolved');
}

checkData().then(data => console.log(data));
```

- A) Resolved
- B) undefined
- C) A Promise object
- D) An error

Correct Answer: A) Resolved

Explanation: The checkData async function awaits the resolution of a promise that resolves with the string 'Resolved'. Because await is used, checkData pauses until the promise is resolved, and then returns the resolved value. When checkData is called, it returns a promise that resolves with the value 'Resolved', which is then logged to the console via .then().

51. How does the Array.prototype.slice method differ from the Array.prototype.splice method?

- A) slice modifies the original array, whereas splice does not.
- B) splice modifies the original array, whereas slice does not.
- C) Both methods modify the original array but in different ways.
- D) Neither method modifies the original array.

Correct Answer: B) splice modifies the original array, whereas slice does not.

Explanation: The slice method returns a shallow copy of a portion of an array into a new array object without modifying the original array. In contrast, splice can remove or replace existing elements and/or add new elements in place, thus modifying the original array.

52. What does the void operator do in JavaScript?

- A) It deletes properties from an object.
- B) It immediately terminates the execution of a function.
- C) It evaluates the given expression and then returns undefined.
- D) It checks if a variable is void of any value.

Correct Answer: C) It evaluates the given expression and then returns undefined.

Explanation: The void operator evaluates the provided expression and then returns undefined. This can be useful in situations where you want to ensure no value is returned from an expression, such as in an immediately invoked function expression (IIFE) or when using a javascript: URL.

53. Which of these is a correct method to create a deep clone of an object in JavaScript?

- A) Using Object.assign({}, obj)
- B) Using the spread syntax { ...obj }
- C) Using JSON.parse(JSON.stringify(obj))
- D) Using Array.prototype.clone(obj)

Correct Answer: C) Using JSON.parse(JSON.stringify(obj))

Explanation: JSON.parse(JSON.stringify(obj)) is a common technique to deep clone an object, assuming the object does not contain functions, undefined values, or circular references, as JSON.stringify cannot handle these cases. Object.assign and the spread syntax create shallow copies.

54. What will the following code snippet log?

```
let a = [1, 2, 3];
let b = [4, 5, 6];
let c = [...a, ...b, ...[7, 8, 9]];

console.log(c.length);
```

- A) 6
- B) 7
- C) 9
- D) 10

Correct Answer: C) 9

Explanation: The spread syntax ... is used to expand the arrays a, b, and [7, 8, 9] into individual elements when creating the new array c. Since c contains all elements from a, b, and the additional array, it has a total of 9 elements.

55. What is the primary use case for the Symbol.iterator property in JavaScript?

- A) To define iteration logic for for-in loops.
- B) To enable an object to become iterable, allowing it to be used with the spread syntax and for-of loops.
- C) To generate unique identifiers for object properties.
- D) To iterate over the keys of an object, including non-enumerable properties.

Correct Answer: B) To enable an object to become iterable, allowing it to be used with the spread syntax and for-of loops.

Explanation: The Symbol.iterator property is used to define the default iterator for an object. When an object implements this method, it becomes iterable, meaning it can be iterated over with a for-of loop, and its elements can be spread into an array or another iterable object.

56. What does the following JavaScript code demonstrate?

```
function debounce(func, wait) {
 let timeout;
 return function executedFunction() {
 const later = () => {
 clearTimeout(timeout);
 func();
 };
 clearTimeout(timeout);
```

```
 timeout = setTimeout(later, wait);
 };
}
```

- A) The throttle technique, limiting how often a function can be called.
- B) The debounce technique, preventing a function from being called repeatedly within a short duration.
- C) A memory leak due to improper handling of timeouts.
- D) An example of a closure leaking memory due to the timeout.

Correct Answer: B) The debounce technique, preventing a function from being called repeatedly within a short duration.

Explanation: This code snippet is an example of the debounce technique, which is used to limit the rate at which a function is executed. It ensures that a particular task does not fire so often that it bricks browser performance. debounce is particularly useful for events like scrolling, resizing, keypresses in text inputs, etc., where the event might be triggered frequently.

57. In JavaScript ES6, what is the main benefit of using Set over an array for storing unique values?

- A) Set automatically removes duplicates, whereas arrays do not.
- B) Set provides better performance for addition and removal of elements.
- C) Set can store values of any type, whereas arrays can only store objects.
- D) Set uses less memory than an array for the same number of elements.

Correct Answer: A) Set automatically removes duplicates, whereas arrays do not.

Explanation: The main benefit of using a Set is that it only stores unique values, automatically removing duplicates, which simplifies working with collections of unique items. While both Set and arrays can store values of any type, Set also typically offers better performance for frequent additions and deletions of elements, especially when dealing with large sets of data.

58. What is the purpose of the async keyword in front of a function declaration in JavaScript?

- A) It indicates that the function returns a Promise and enables the use of await within it.
- B) It makes the function execute asynchronously in a separate thread, similar to a Web Worker.
- C) It automatically catches any errors within the function and rejects the promise.
- D) It optimizes the function for performance by running it in parallel with other code.

Correct Answer: A) It indicates that the function returns a Promise and enables the use of await within it.

Explanation: The async keyword before a function declaration or expression makes the function return a promise, and allows the use of await within it to pause execution until the awaited promise is fulfilled or rejected. This simplifies writing asynchronous code by making it look more like synchronous code.

59. What is the result of spreading an object into an array in JavaScript?

```
let obj = {a: 1, b: 2};
let arr = [...obj];
```

- A) [{"a": 1, "b": 2}]

- B) ["a", "b"]
- C) TypeError: obj is not iterable
- D) [{a: 1}, {b: 2}]

Correct Answer: C) TypeError: obj is not iterable

Explanation: Spreading an object directly into an array throws a TypeError because objects are not iterable unless they implement the iterable protocol (i.e., having a [Symbol.iterator] method). Spread syntax works with iterables like arrays, strings, or objects implementing the iterable protocol.

60. How do you ensure that a piece of code runs after several asynchronous operations have completed in JavaScript?

- A) Using a for loop to iterate through the asynchronous operations.
- B) Using the Promise.all method with an array of the asynchronous operations.
- C) Placing the asynchronous operations inside a try-catch block.
- D) Calling each asynchronous operation with an await keyword in sequence.

Correct Answer: B) Using the Promise.all method with an array of the asynchronous operations.

Explanation: Promise.all takes an iterable of promises as an input, and it returns a single Promise that resolves when all of the promises in the iterable have resolved or when the iterable contains no promises. It is the correct choice for running code after multiple asynchronous operations have completed, as it waits for all operations to finish.

61. What will be the output of the following code snippet?

```
console.log(1 < 2 < 3);
console.log(3 > 2 > 1);
```

- A) true and true
- B) true and false
- C) false and true
- D) false and false

Correct Answer: B) true and false

Explanation: The expression 1 < 2 < 3 evaluates to true < 3, which is true because true is coerced to 1, making the expression 1 < 3, which is true. The expression 3 > 2 > 1 evaluates to true > 1, which is false because true is coerced to 1, making the expression 1 > 1, which is false.

62. How can you remove duplicates from an array in JavaScript?

- A) By using the Array.prototype.unique() method.
- B) By creating a Set from the array.
- C) By using the Array.prototype.filter() method.
- D) B and C are both correct ways.

Correct Answer: D) B and C are both correct ways.

Explanation: Creating a Set from the array automatically removes duplicates because a Set only stores unique values. The Array.prototype.filter() method can also be used to remove duplicates by checking the index of each item. There's no Array.prototype.unique() method in JavaScript.

63. What is the purpose of the finally clause in a try...catch statement?

- A) To execute code after the try block succeeds without an error.
- B) To handle any uncaught errors that were not caught by the catch block.
- C) To execute code after the try and catch blocks, regardless of whether an exception was thrown.
- D) To clean up resources that were allocated before the try block.

Correct Answer: C) To execute code after the try and catch blocks, regardless of whether an exception was thrown.

Explanation: The finally clause is executed after the try and catch blocks have completed, regardless of whether an exception was thrown or caught, making it useful for cleanup activities that need to happen under all circumstances.

64. What is NaN in JavaScript?

- A) A global property representing Not-A-Number.
- B) A function that checks if a value is not a number.
- C) An error type indicating a failed number conversion.
- D) A value representing a computation that cannot produce a normal result.

Correct Answer: D) A value representing a computation that cannot produce a normal result.

Explanation: NaN is a property of the global object. In other words, it is a variable in global scope. The initial value of NaN is Not-A-Number — the same as the value of Number.NaN. It represents a value that is not a legal number, often resulting from math operations that fail to produce a normal result.

65. Which method can be used to ensure a function is called with a particular this context?

- A) Function.prototype.call()
- B) Function.prototype.bind()
- C) Function.prototype.apply()
- D) All of the above

Correct Answer: D) All of the above

Explanation: Function.prototype.call(), Function.prototype.apply(), and Function.prototype.bind() can all be used to specify the this context for a function call. call() and apply() invoke the function immediately with a given this value, whereas bind() returns a new function bound to a specified this context.

66. How does JavaScript's garbage collection mechanism handle circular references?

- A) JavaScript cannot collect objects involved in circular references, leading to memory leaks.
- B) Modern JavaScript engines use a mark-and-sweep algorithm, which can collect objects in circular references.
- C) Circular references must be manually broken by the developer to be garbage collected.
- D) JavaScript's garbage collection mechanism does not support circular references.

Correct Answer: B) Modern JavaScript engines use a mark-and-sweep algorithm, which can collect objects in circular references.

Explanation: Modern JavaScript engines employ a garbage collection algorithm known as mark-and-sweep. This algorithm can identify and collect objects in circular references, thus preventing memory leaks that were a problem in older garbage collection strategies, like reference counting.

67. In JavaScript, what will the following code output?

```
async function test() {
 return 21;
}
test().then(console.log);
```

- A) 21
- B) Promise {<resolved>: 21}
- C) undefined
- D) Promise {<pending>}

Correct Answer: A) 21

Explanation: An async function always returns a promise. The promise is resolved with the value the function returns. In this case, test() returns a promise that resolves to 21, so 21 is logged to the console.

68. Which statement about Web Workers in JavaScript is true?

- A) They can directly manipulate the DOM.
- B) They run in the same thread as the main JavaScript code, ensuring synchronous execution.
- C) They allow you to run JavaScript in background threads, communicating with the main thread via messaging.
- D) They are primarily used to improve the performance of JavaScript's event loop.

Correct Answer: C) They allow you to run JavaScript in background threads, communicating with the main thread via messaging.

Explanation: Web Workers provide a way to run scripts in background threads. The worker thread can perform tasks without interfering with the user interface. Although workers operate independently of the main thread and cannot directly manipulate the DOM, they can communicate with the main thread using a system of messages, allowing for parallel execution of JavaScript code.

69. What is the result of the expression typeof null in JavaScript?

- A) "null"
- B) "undefined"
- C) "object"

- D) "no-value"

Correct Answer: C) "object"

Explanation: In JavaScript, typeof null is "object", which is a long-standing bug in the language. null is not actually an object; it is a primitive value that represents the absence of any object value.

70. How can you prevent a form from being submitted using JavaScript?

- A) Set document.form.submit to false.
- B) Use event.preventDefault() in the form's submit event handler.
- C) Remove the form's action attribute.
- D) Use return false; at the end of the form's onsubmit attribute.

Correct Answer: B) Use event.preventDefault() in the form's submit event handler.

Explanation: The event.preventDefault() method prevents the default action the browser takes on the event. When used in a form's submit event handler, it prevents the form from being submitted. This is useful for validating the form before submission or when submitting the form data via JavaScript (e.g., AJAX).

71. What is the output of the following JavaScript code?

```
let x = 'outer scope';
(function() {
 let x = 'inner scope';
 (function() {
 x = 'innermost scope';
 })();
 console.log(x);
})();
```

- A) outer scope
- B) inner scope
- C) innermost scope
- D) undefined

Correct Answer: C) innermost scope

Explanation: The code snippet demonstrates lexical scoping in JavaScript. The innermost self-invoking function modifies the x variable declared in its parent scope (the first self-invoking function) to 'innermost scope'. Since the console.log(x); statement is executed in the same scope where x is modified, it logs innermost scope. The outer x variable is shadowed by the inner x and remains unchanged.

72. How can you achieve module encapsulation in JavaScript ES6?

- A) Using the module keyword
- B) Using the export and import statements
- C) Encapsulating module code inside a function
- D) Using the encapsulate keyword

Correct Answer: B) Using the export and import statements

Explanation: In ES6, module encapsulation is achieved using export and import statements. export allows you to expose variables, functions, or classes as part of the module's public API, while import allows you to bring them into another module. This system provides a way to encapsulate module functionality and expose only the desired parts to the consuming code.

73. What will the following JavaScript code print?

```
const promise1 = Promise.resolve(3);
const promise2 = 42;
const promise3 = new Promise((resolve, reject) => {
 setTimeout(resolve, 100, 'foo');
```

```
});

Promise.all([promise1, promise2, promise3]).then(values => {
 console.log(values);
});
```

- A) [3, 42, 'foo']
- B) [Promise, 42, Promise]
- C) [3, undefined, 'foo']
- D) SyntaxError

Correct Answer: A) [3, 42, 'foo']

Explanation: Promise.all takes an iterable of promises and returns a single Promise that resolves when all of the promises in the iterable have resolved or when the iterable contains no promises. It resolves with an array of the results of the input promises. In this case, promise1, promise2, and promise3 resolve with the values 3, 42, and 'foo', respectively. Therefore, the logged output is [3, 42, 'foo'].

74. In JavaScript, what does the ?. operator represent?

- A) Nullish coalescing
- B) Optional chaining
- C) Existential operator
- D) Logical OR

Correct Answer: B) Optional chaining

Explanation: The ?. operator in JavaScript represents optional chaining. It allows you to safely access deeply nested properties of an object without having to check if each reference in the chain is null or undefined. If a reference is nullish (null or undefined), the expression short-circuits and returns undefined.

75. What does the following code demonstrate?

```
async function fetchData() {
 try {
 const resp = await fetch('https://api.example.com/data');
 if (!resp.ok) {
 throw new Error('Network response was not ok');
 }
 const data = await resp.json();
 return data;
 } catch (error) {
 console.error('Fetch error:', error.message);
 }
}
```

- A) Synchronous error handling in asynchronous code
- B) Use of the Fetch API to make network requests
- C) Handling network errors and parsing JSON response
- D) All of the above

Correct Answer: D) All of the above

Explanation: This code snippet demonstrates several concepts: (A) it shows how try...catch can be used to handle errors in asynchronous code using async/await; (B) it uses the Fetch API to make network requests; (C) it handles potential network errors by checking the ok property of the response and parsing the JSON response with await resp.json(). Therefore, all of the listed options are demonstrated by this code.

76. How can you stop the propagation of an event in JavaScript?

- A) event.stop()
- B) event.preventDefault()
- C) event.stopPropagation()

- D) event.cancelBubble()

Correct Answer: C) event.stopPropagation()

Explanation: The event.stopPropagation() method is used to stop the propagation of an event through the DOM. It prevents the event from bubbling up to the parent elements or capturing down to the child elements, effectively isolating the event to the current element.

77. What is the default return value of a JavaScript function that does not explicitly return a value?

- A) null
- B) NaN
- C) undefined
- D) 0

Correct Answer: C) undefined

Explanation: In JavaScript, if a function does not explicitly return a value using the return statement, it implicitly returns undefined. This is a fundamental aspect of JavaScript's function behavior.

78. What is the main use of the Set object introduced in ES6?

- A) To store a collection of unique values of any type
- B) To keep key-value pairs for complex data structures
- C) To organize data into a structured table format
- D) To provide a mechanism for inheritance

Correct Answer: A) To store a collection of unique values of any type

Explanation: The Set object lets you store unique values of any type, whether primitive values or object references. It is particularly useful for ensuring the uniqueness of elements in a collection without having to manually check for duplicates.

79. Consider the following code. What will it output and why?

```
console.log(0.1 + 0.2 === 0.3);
```

- A) true, because the sum of 0.1 and 0.2 is exactly 0.3
- B) false, due to floating-point arithmetic errors
- C) true, because JavaScript corrects floating-point errors automatically
- D) false, because === checks for type and value, and they are of different types

Correct Answer: B) false, due to floating-point arithmetic errors

Explanation: In JavaScript, the sum of 0.1 and 0.2 does not exactly equal 0.3 due to the way floating-point arithmetic is handled by the IEEE 754 standard, which JavaScript follows. This standard causes small rounding errors in calculations, making the strict equality comparison return false.

80. How does JavaScript handle asynchronous functions such as setTimeout, fetch, and other promise-based operations internally?

- A) By blocking the main thread until the asynchronous operation completes
- B) Using a multi-threading model similar to languages like Java or C#
- C) Through an event loop that allows non-blocking I/O operations
- D) By pausing the execution of the script until the operation completes

Correct Answer: C) Through an event loop that allows non-blocking I/O operations

Explanation: JavaScript uses an event loop mechanism to handle asynchronous operations. This model allows JavaScript to perform non-blocking I/O operations, despite being single-threaded, by offloading

operations to the system where possible and managing an event queue for callbacks associated with these operations.

81. What is a potential pitfall of using typeof operator in JavaScript?

- A) It can incorrectly identify null as an object.
- B) It cannot differentiate between arrays and objects.
- C) It returns 'undefined' for both null values and uninitialized variables.
- D) Both A and B are correct.

Correct Answer: D) Both A and B are correct.

Explanation: The typeof operator in JavaScript returns 'object' for null values, which can be misleading because null is not actually an object. Additionally, it does not differentiate between arrays and objects, returning 'object' for both, which can be a pitfall when trying to accurately determine the type of a collection.

82. What does the finally block in a try-catch-finally construct do?

- A) It executes code after the try block only if no exceptions were thrown.
- B) It executes code after the catch block only if an exception was caught.
- C) It executes code after the try and catch blocks regardless of an exception being thrown or caught.
- D) It cleans up resources used in the try block before the function returns.

Correct Answer: C) It executes code after the try and catch blocks regardless of an exception being thrown or caught.

Explanation: The finally block in a try-catch-finally construct is executed after the execution of the try and catch blocks, regardless of whether an exception was thrown or caught. It is often used for cleaning up resources

or executing code that must run after try-catch blocks, such as closing files or releasing resources.

83. How do you create a private variable in a JavaScript class?

- A) Using the var keyword inside the class declaration.
- B) Prefixing the variable name with the # symbol.
- C) Declaring the variable inside the constructor only.
- D) Using the private keyword before the variable name.

Correct Answer: B) Prefixing the variable name with the # symbol.

Explanation: In JavaScript, private class fields can be created by prefixing the field name with the # symbol. These fields are only accessible within the class itself and are not accessible from outside the class, providing encapsulation.

84. What is the purpose of the Map object in JavaScript?

- A) To create a simple key-value pair collection.
- B) To store unique values of any type.
- C) To store key-value pairs with keys of any type and maintain insertion order.
- D) To synchronize data access in asynchronous programming.

Correct Answer: C) To store key-value pairs with keys of any type and maintain insertion order.

Explanation: The Map object in JavaScript is used to store key-value pairs. Unlike objects, which only support string and symbol keys, Map allows keys of any type, including objects and primitive types. Additionally, Map maintains the insertion order of its elements, which can be important for certain algorithms or data processing tasks.

85. Which method is used to serialize an object into a JSON string in JavaScript?

- A) JSON.stringify()
- B) JSON.parse()
- C) Object.serialize()
- D) Object.toString()

Correct Answer: A) JSON.stringify()

Explanation: JSON.stringify() method is used to convert a JavaScript object or value to a JSON string. This method can take an object, array, or primitive value and produce a JSON string representation, which can be used for storage or transmission of data, such as in web requests.

86. How can you prevent the default action of an event in JavaScript?

- A) event.cancel()
- B) event.stop()
- C) event.preventDefault()
- D) event.stopPropagation()

Correct Answer: C) event.preventDefault()

Explanation: The event.preventDefault() method is used to prevent the browser's default action associated with an event, without stopping the event's propagation through the DOM. This is commonly used in form submission handling, link navigation, and other events where the default behavior is not desired.

87. What is event delegation in JavaScript?

- A) Assigning an event listener to a parent element rather than multiple child elements.
- B) A method of optimizing event handling by consolidating event listeners.
- C) Redirecting events from one element to another.
- D) Both A and B are correct.

Correct Answer: D) Both A and B are correct.

Explanation: Event delegation is a technique where instead of adding an event listener to each child element, you add a single event listener to a parent element. This listener analyzes bubbled events to find a match on child elements. This technique optimizes performance by reducing the number of event listeners and simplifies event management in dynamic applications where elements may be added or removed.

88. What will the following code output?

```
let num = 8;
let result = num++ + ++num;
console.log(result);
```

- A) 16
- B) 17
- C) 18
- D) 19

Correct Answer: D) 19

Explanation: This code snippet involves both post-increment and pre-increment operations. Initially, num is 8. In the expression num++, the value of num is used before it is incremented, so it contributes 8 to the sum. After num++, num becomes 9. Then, ++num pre-increments num to 10 before it is added to the sum. Thus, the result is 8 + 11 = 19.

89. How do you copy an object in JavaScript?

- A) Using the assignment operator (=)
- B) Using the Object.copy() method
- C) Using the Object.assign() method or the spread operator (...)
- D) Objects cannot be copied in JavaScript; they are always passed by reference.

Correct Answer: C) Using the Object.assign() method or the spread operator (...)

Explanation: Objects in JavaScript are reference types and are assigned or passed by reference. To create a shallow copy of an object, you can use the Object.assign() method with an empty object as the target and the source object as the second argument. Alternatively, the spread operator (...) can be used to spread the properties of the source object into a new object literal, effectively creating a shallow copy.

90. What does the Array.prototype.reduce() method do in JavaScript?

- A) It reduces the array to a single value by executing a reducer function on each element of the array.
- B) It removes elements from an array that do not meet certain criteria.
- C) It creates a new array with the results of calling a provided function on every element in the calling array.
- D) It reduces the size of an array by removing empty elements.

Correct Answer: A) It reduces the array to a single value by executing a reducer function on each element of the array.

Explanation: The Array.prototype.reduce() method in JavaScript takes a reducer function as an argument and an optional initial value. It executes the reducer function on each element of the array, resulting in a single output value. The reducer function is called with four arguments: accumulator, currentValue, currentIndex, and the array itself. This method is useful for summing up numbers, concatenating strings, or compiling a single result from an array of values.

91. Which of the following options correctly describes the behavior of the this keyword in JavaScript?

- A) this always refers to the global object.

- B) this refers to the object from which the currently executing function was called.
- C) this reference is statically scoped to the context where a function is defined.
- D) this can be bound at runtime using function methods like bind, call, and apply.

Correct Answer: D) this can be bound at runtime using function methods like bind, call, and apply.

Explanation: The this keyword in JavaScript does not have a static scope; its value is determined based on how the function is called. It can refer to different objects depending on the calling context. Furthermore, this can be explicitly bound at runtime using bind, call, or apply methods, altering its default binding mechanism.

92. What is the output of the following code snippet?

```
function Person() {
 this.age = 20;

 setTimeout(() => {
 console.log(this.age);
 }, 1000);
}

var p = new Person();
```

- A) 20
- B) undefined
- C) ReferenceError
- D) NaN

Correct Answer: A) 20

Explanation: Arrow functions do not have their own this context but inherit this from the surrounding code's context. In this case, the arrow function inside setTimeout inherits this from the Person constructor function context. Therefore, this.age refers to the age property of the Person instance, and 20 is printed to the console.

93. How can you ensure that a script is executed after the HTML document has been fully loaded?

- A) Place the script tag at the end of the body section.
- B) Use the document.onload event.
- C) Use the window.onload event.
- D) Both A and C are correct.

Correct Answer: D) Both A and C are correct.

Explanation: Placing the script tag at the end of the body section ensures that the script is executed after the HTML document is parsed. Alternatively, attaching an event handler to window.onload ensures that your script runs after the entire page, including all dependent resources like images, has been fully loaded. document.onload is not commonly used for this purpose; DOMContentLoaded is a more appropriate choice for waiting on the document to be fully parsed.

94. What is "hoisting" in JavaScript?

- A) The process of moving variable declarations to the top of their containing scope.
- B) Automatically converting local variables to global variables.
- C) Raising an error when variables are used before declaration.
- D) Elevating function declarations to the top of the script file.

Correct Answer: A) The process of moving variable declarations to the top of their containing scope.

Explanation: Hoisting in JavaScript refers to the behavior of moving declarations (variables and functions) to the top of their containing scope (global or function) before code execution begins. It allows variables and functions to be used before they are declared. Note that only the declarations are hoisted, not initializations.

95. Which statement about JavaScript modules is true?

- A) Modules automatically run in strict mode.
- B) Variables declared in modules are globally scoped.
- C) Modules can only export a single value or function.
- D) Modules are executed synchronously.

Correct Answer: A) Modules automatically run in strict mode.

Explanation: JavaScript modules (import/export) inherently operate in strict mode, and this mode applies to the entire module. Variables declared in modules are scoped to the module, not global. Modules can export multiple values and are loaded asynchronously.

96. What does the following expression evaluate to?

typeof new Boolean(false)

- A) "boolean"
- B) "object"
- C) "false"
- D) "undefined"

Correct Answer: B) "object"

Explanation: The new Boolean(false) expression creates a Boolean object wrapper for the false value. The typeof operator returns "object" for objects, arrays, and null, so in this case, it returns "object". It's important to distinguish between primitive boolean values and Boolean objects.

97. How can you capture all click events in a document, including those inside iframes?

- A) By adding a click event listener to the document object.
- B) By adding a click event listener to the window object.
- C) By adding a click event listener to each iframe element.
- D) By using the capture phase of event propagation on the document's body.

Correct Answer: C) By adding a click event listener to each iframe element.

Explanation: To capture click events inside iframes as well as in the main document, you would need to attach event listeners to each iframe document in addition to the main document. This is because an iframe has its own separate document object. Note that this requires access to the iframe's content, which is subject to the same-origin policy.

98. What is the result of the following code snippet?

```
console.log("1" - - "1");
```

- A) "11"
- B) "2"
- C) 2
- D) SyntaxError

Correct Answer: B) "2"

Explanation: In this expression, the subtraction operator (-) triggers implicit type coercion, converting the string operands to numbers. The second - is interpreted as the unary negation operator, making the expression equivalent to 1 - (-1), which results in 2. The key point is understanding how JavaScript handles type coercion with arithmetic operators.

99. What feature does the async keyword enable in a function in JavaScript?

- A) The function returns a promise.
- B) The function can use the await keyword.
- C) The function executes in a separate thread from the main execution thread.
- D) Both A and B are correct.

Correct Answer: D) Both A and B are correct.

Explanation: The async keyword before a function declaration or function expression makes it an async function. This means two things: first, the function returns a promise, and second, within the function body, the await keyword can be used to pause the async function's execution until a promise is resolved, in a way that looks synchronous.

100. How does the JavaScript engine handle the following code snippet?

```
const obj = {
 a: 1,
 b: () => {
 console.log(this.a);
 },
 c: function() {
 console.log(this.a);
 }
};

obj.b(); // Call 1
obj.c(); // Call 2
```

- A) Both calls log 1.

- B) Call 1 logs undefined; Call 2 logs 1.
- C) Call 1 logs 1; Call 2 logs undefined.
- D) Both calls log undefined.

Correct Answer: B) Call 1 logs undefined; Call 2 logs 1.

Explanation: Arrow functions do not have their own this context; they inherit it from the parent scope, which, in the case of obj.b(), is the global context (or undefined in strict mode), so this.a is undefined. Regular functions, however, do have their own this context, which refers to the object from which they were called. Thus, obj.c() logs 1, as this refers to obj, and obj.a is 1.