150 JavaScript

Questions *with Explanations*

🚀 Test your skills how many can you answer 🚀

In JavaScript, how does the Set object handle NaN values?

- A) It treats all NaN values as undefined.

- B) It cannot include NaN values because they are not considered valid values.
- C) It considers all NaN values as identical, allowing only one NaN value in the Set.
- D) Each NaN value is considered unique, allowing multiple NaN entries.

Correct Answer: C) It considers all NaN values as identical, allowing only one NaN value in the Set.

Explanation: In JavaScript, the Set object treats all NaN values as identical, despite NaN not being equal to itself in equality comparisons. Therefore, a Set can only contain one entry with a value of NaN, demonstrating an exception to the usual strict equality comparison used for other values.

What will the following code output and why?

```
console.log(0.1 + 0.2 === 0.3);
console.log(Math.abs((0.1 + 0.2) - 0.3) < Number.EPSILON);
```

- A) false, true because floating-point arithmetic cannot precisely represent 0.1 + 0.2.
- B) true, false because JavaScript corrects floating-point errors in comparisons.
- C) true, true because both statements accurately compare floating-point numbers.
- D) false, false because of inherent inaccuracies in floating-point arithmetic and the EPSILON comparison.

Correct Answer: A) false, true because floating-point arithmetic cannot precisely represent 0.1 + 0.2.

Explanation: The first comparison directly checks if 0.1 + 0.2 equals 0.3, which it does not due to the way floating-point numbers are represented in JavaScript (and many other programming languages), leading to precision errors. The second comparison uses Number.EPSILON to check if the difference between (0.1 + 0.2) and 0.3 is smaller than the smallest difference JavaScript can represent between two numbers, which is a correct way to compare floating-point numbers for "equality" within a tolerance.

Which statement best describes the use of the Promise.race() method in JavaScript?

- A) It waits for all promises to resolve or for any to be rejected.
- B) It resolves or rejects as soon as one of the promises in the iterable resolves or rejects, with the value or reason from that promise.
- C) It only resolves when all promises in the iterable have resolved, returning an array of the results.
- D) It creates a race condition between multiple asynchronous operations, causing unpredictable results.

Correct Answer: B) It resolves or rejects as soon as one of the promises in the iterable resolves or rejects, with the value or reason from that promise.

Explanation: The Promise.race() method returns a promise that resolves or rejects as soon as one of the promises in the iterable settles (resolves or rejects). The returned promise is fulfilled with the value of the first promise that settled, making it useful for timeout patterns or resolving a promise as soon as the first asynchronous operation completes.

How can you effectively check if an object is an array in JavaScript?

- A) Using typeof object === "array"
- B) Using object instanceof Array
- C) Using Array.isArray(object)
- D) Both B and C are correct methods.

Correct Answer: D) Both B and C are correct methods.

Explanation: While Array.isArray(object) is the most straightforward and recommended way to check if an object is an array because it returns true if the object is an array, object instanceof Array also works by checking if Array.prototype exists in the object's prototype chain. However, Array.isArray() is generally preferred for its clarity and reliability across different execution contexts.

What is the result of the following operation in JavaScript?

"10" * 5

- A) 50
- B) "50"
- C) NaN
- D) TypeError

Correct Answer: A) 50

Explanation: In JavaScript, the multiplication operation coerces string operands to numbers if possible. Since "10" can be coerced to the number 10, and multiplying by 5 yields 50, the operation results in the number 50, not a string or an error.

What does the async function return in JavaScript?

- A) A value or object specified by the return statement within the function.
- B) A promise, which is resolved with the value returned by the function.
- C) An immediately-invoked function expression (IIFE).
- D) A callback function specified within the async function.

Correct Answer: B) A promise, which is resolved with the value returned by the function.

Explanation: An async function implicitly returns a promise, which is resolved with the value that the function returns. If the function throws an error, the returned promise is rejected with that error. This allows async functions to be used in promise chains and with the await operator.

What is the main difference between == and === in JavaScript?

- A) == checks for value equality, while === checks for value and type equality.
- B) == performs type coercion, while === does not.
- C) === is strictly for object comparisons, while == is used for primitive types.
- D) There is no difference; both operators are interchangeable.

Correct Answer: B) == performs type coercion, while === does not.

Explanation: The == operator in JavaScript performs type coercion, trying to convert the operands to the same type before making the comparison. In contrast, the === operator, known as the strict equality operator, does not perform type coercion and directly compares both value and type, which can lead to different results when comparing values of different types.

How can you convert a NodeList to an array in JavaScript?

- A) Using Array.from(nodeList)
- B) Using nodeList.toArray()
- C) Using [] + nodeList
- D) Using nodeList.split(",")

Correct Answer: A) Using Array.from(nodeList)

Explanation: Array.from() is a static method that creates a new, shallow-copied Array instance from an array-like or iterable object, such as a NodeList. This method is the correct and efficient way to convert a NodeList to an array, allowing the use of array methods like map(), filter(), and reduce() on the resulting array.

In JavaScript, what is event bubbling?

- A) The process by which an event triggers the event handlers of child elements before the parent elements.
- B) The direct handling of an event by the element that dispatched it, without involving parent or child elements.

- C) The process by which an event starts from the parent element and propagates down to child elements.
- D) The process by which an event starts from the element that triggered it and flows up to the DOM tree to the document.

Correct Answer: D) The process by which an event starts from the element that triggered it and flows up to the DOM tree to the document.

Explanation: Event bubbling is a type of event propagation in the DOM where the event starts from the specific element that triggered it (the target element) and then flows upward through its ancestors in the DOM tree, potentially triggering event handlers on each of those ancestors. This behavior is the default for most events in the DOM.

What is the output of the following code snippet?

```
let x = { y: 1 };
let z = x;
z.y = 2;
console.log(x.y);
```

- A) 1
- B) 2
- C) undefined
- D) ReferenceError

Correct Answer: B) 2

Explanation: Objects in JavaScript are assigned by reference. When z is assigned the value of x, it does not create a copy of x but rather a reference to the same object. Therefore, modifying z.y also modifies x.y, since z and x refer to the same object in memory. Consequently, the output is 2.

What will the following JavaScript code output to the console?

```
function checkHoisting() {
console.log(text);
var text = 'Hoisted?';
}
checkHoisting();
```

- A) Hoisted?
- B) undefined
- C) ReferenceError: text is not defined
- D) null

Correct Answer: B) undefined

Explanation: Due to variable hoisting in JavaScript, the declaration of the variable text is moved to the top of its scope (the checkHoisting function). However, its assignment happens after the console.log call. Thus, at the time of logging, text is declared but not defined, resulting in undefined.

How can the fetch API handle JSON responses in JavaScript?

- A) fetch(url).then(res => res.json())
- B) fetch(url).json()
- C) JSON.parse(fetch(url))
- D) fetch(url, {dataType: 'json'})

Correct Answer: A) fetch(url).then(res => res.json())

Explanation: The fetch API returns a promise that resolves to a Response object. To extract the JSON body content from the Response object, the json() method is used, which itself returns a promise resolved with the result of parsing the body text as JSON.

Which of the following correctly describes a way to create a deep copy of an object in JavaScript?

- A) const deepCopy = Object.assign({}, sourceObject);
- B) const deepCopy = {...sourceObject};
- C) const deepCopy = JSON.parse(JSON.stringify(sourceObject));
- D) const deepCopy = sourceObject.clone();

Correct Answer: C) const deepCopy = JSON.parse(JSON.stringify(sourceObject));

Explanation: Options A and B create shallow copies of the sourceObject. They do not duplicate objects nested within sourceObject, meaning nested objects remain linked between the original and the copy. Option D (clone()) is not a native JavaScript method for cloning objects. Option C, JSON.parse(JSON.stringify(sourceObject)), creates a deep copy of sourceObject, including all nested objects. However, this method cannot copy functions, circular references, or special objects like Date, RegExp, etc., accurately.

How can you delay the execution of a function in JavaScript?

- A) setTimeout(function, milliseconds)

- B) delay(function, milliseconds)
- C) function.wait(milliseconds)
- D) pause(function, milliseconds)

Correct Answer: A) setTimeout(function, milliseconds)

Explanation: setTimeout() is a standard JavaScript function that asynchronously delays the execution of a function by a specified number of milliseconds. Options B, C, and D are not standard JavaScript methods for delaying function execution.

What is the primary purpose of the Symbol primitive type introduced in ES6?

- A) To provide unique property keys to prevent name collisions
- B) To replace string-based enums for better type safety
- C) To create private object members
- D) To optimize memory usage for string values

Correct Answer: A) To provide unique property keys to prevent name collisions

Explanation: The Symbol primitive type in JavaScript ES6 is primarily used to create unique identifiers for object properties. Because each Symbol is unique, they help prevent name collisions, even if multiple symbols share the same description. While symbols can be used to mimic private object members and may contribute to slightly different memory characteristics, their main purpose is not B, C, or D.

Which method is used to add an element to the beginning of an array in JavaScript?

- A) array.push(element)
- B) array.unshift(element)
- C) array.append(element)
- D) array.prepend(element)

Correct Answer: B) array.unshift(element)

Explanation: array.unshift(element) adds one or more elements to the beginning of an array and returns the new length of the array. array.push(element) adds elements to the end of an array. Options C and D are not standard methods for arrays in JavaScript.

How do you create a new promise in JavaScript?

- A) const promise = new Promise((resolve, reject) => { /* executor */ });
- B) const promise = Promise.create((resolve, reject) => { /* executor */ });
- C) const promise = Promise((resolve, reject) => { /* executor */ });
- D) const promise = async (resolve, reject) => { /* executor */ };

Correct Answer: A) const promise = new Promise((resolve, reject) => { /* executor */ });

Explanation: A new Promise in JavaScript is created using the new Promise constructor that takes an executor function. This executor function is called immediately by the Promise implementation, passing in two functions: resolve and reject that you use to resolve or reject the promise.

What feature does the async/await syntax in JavaScript simplify?

- A) Callback functions for event listeners
- B) Synchronous execution of asynchronous code
- C) Iteration over arrays
- D) Deep copying objects

Correct Answer: B) Synchronous execution of asynchronous code

Explanation: The async/await syntax in JavaScript simplifies writing asynchronous code by allowing developers to write code that looks synchronous but is actually asynchronous. This syntax makes it easier to read and write promises, reducing the complexity of handling asynchronous operations, such as network requests, timers, or any task that requires waiting for completion.

Which of these statements will correctly remove an element from the beginning of an array?

- A) array.pop()
- B) array.shift()
- C) array.remove(0)
- D) array.delete(0)

Correct Answer: B) array.shift()

Explanation: array.shift() removes the first element from an array and returns that removed element. This method changes the length of the array. array.pop(), on the other hand, removes the last element from an array. Options C and D are not standard array methods in JavaScript.

In JavaScript, what does the document.querySelector() method return?

- A) All elements that match the specified group of selectors.
- B) The first element within the document that matches the specified selector.
- C) An array of elements that match the specified selector.
- D) The last element within the document that matches the specified selector.

Correct Answer: B) The first element within the document that matches the specified selector.

Explanation: document.querySelector() returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned. It is a powerful method for DOM manipulation and querying.

How do you iterate over the properties of an object in JavaScript?

- A) Using a for...of loop
- B) Using a for...in loop
- C) Using Object.forEach()
- D) Using Array.map()

Correct Answer: B) Using a for...in loop

Explanation: A for...in loop is used to iterate over all enumerable properties of an object, including inherited enumerable properties. This makes it suitable for iterating over object properties. for...of is used for iterable objects like Arrays, Maps, Sets, etc., and does not apply directly to plain objects. Object.forEach() and Array.map() are not valid methods for directly iterating over object properties.

What is the output of the following code?

```
let x = 1;
function changeX(y) {
y = 2;
}
changeX(x);
console.log(x);
```

- A) 1
- B) 2
- C) undefined
- D) ReferenceError

Correct Answer: A) 1

Explanation: Primitive values in JavaScript (like numbers) are passed to functions by value. This means that the changeX function operates on a copy of x, not x itself. Changing y inside the function does not affect the original x outside the function. Hence, the output remains 1.

Which statement about JavaScript Promises is true?

- A) A Promise is a JavaScript object that links producing code and consuming code.
- B) A Promise can only be resolved once, either fulfilled or rejected.
- C) Once a Promise is fulfilled or rejected, it is immutable and cannot change its state.
- D) All of the above are true.

Correct Answer: D) All of the above are true.

Explanation: Promises in JavaScript represent the eventual completion (or failure) of an asynchronous operation and its resulting value. A Promise is indeed a link between producing code (where the Promise is created) and consuming code (which awaits the Promise's resolution). It can only resolve once and remains in that state (fulfilled or rejected) immutably, ensuring predictable behavior in asynchronous flow.

How do you create a private method in a JavaScript class?

- A) Prefixing the method name with #
- B) Declaring the method inside the constructor
- C) Using the private keyword before the method name
- D) JavaScript does not support private methods.

Correct Answer: A) Prefixing the method name with #

Explanation: In JavaScript, private class features are achieved by prefixing the name with #. This includes both fields and methods. A method defined with # before its name is a private method, accessible only within the class that defines it. This syntax is part of the recent ECMAScript specifications for class field declarations that provide true privacy.

What does the flatMap method do in JavaScript?

- A) It flattens an array of arrays by one level and maps each element using a mapping function.

- B) It maps each element to a new element using a provided function, without changing the structure of the array.
- C) It flattens a multidimensional array into a one-dimensional array.
- D) It applies a function to each element of the array and flattens the result into a new array.

Correct Answer: A) It flattens an array of arrays by one level and maps each element using a mapping function.

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

What is the Event Loop in JavaScript?

- A) A loop that continuously checks for user events and triggers callbacks.
- B) A programming construct that loops through an array of event listeners.
- C) The mechanism that handles executing multiple JavaScript chunks over the event queue.
- D) A concept in Node.js for handling asynchronous operations.

Correct Answer: C) The mechanism that handles executing multiple JavaScript chunks over the event queue.

Explanation: The Event Loop is a fundamental part of the JavaScript runtime model, which allows JavaScript to perform non-blocking asynchronous operations. Despite JavaScript being single-threaded, the

Event Loop facilitates handling events, callbacks, and I/O operations effectively by managing an event queue and executing chunks of code sequentially.

How do you ensure data immutability in a JavaScript object?

- A) Using const to declare the object
- B) Using Object.freeze()
- C) Using Object.seal()
- D) Using Immutable.js library

Correct Answer: B) Using Object.freeze()

Explanation: Object.freeze() makes an object immutable in JavaScript by preventing new properties from being added to it, existing properties from being removed or changed. While const ensures that the object reference cannot be changed, it does not make the object's properties immutable. Object.seal() prevents new properties from being added and marks all existing properties as non-configurable, but values of existing properties can still be changed if they are writable.

In JavaScript, what is a higher-order function?

- A) A function that only accepts functions as arguments
- B) A function that operates on other functions, either by taking them as arguments or by returning them
- C) A function that returns a higher version of ECMAScript standard
- D) A function with a higher scope than its outer function

Correct Answer: B) A function that operates on other functions, either by taking them as arguments or by returning them

Explanation: Higher-order functions are a concept in functional programming where functions can take one or more functions as arguments and/or return a function. This allows for functional constructs like map, reduce, and filter, which are fundamental in functional programming paradigms.

Which method can be used to ensure a block of code runs repeatedly at specified intervals in JavaScript?

- A) setTimeout()
- B) setInterval()
- C) repeat()
- D) loop()

Correct Answer: B) setInterval()

Explanation: setInterval() is used to execute a block of code or a function repeatedly with a fixed time delay between each call. It returns an interval ID, which can be used to stop the execution with clearInterval(). setTimeout() executes a block of code once after a specified delay.

What does the ?. operator do in JavaScript?

- A) It is a comparison operator that checks if two values are identical without type coercion.
- B) It is a mathematical operator that sums two values and handles nullish values by treating them as zero.

- C) It is an optional chaining operator that allows reading the value of a property located deep within a chain of connected objects without having to expressly validate that each reference in the chain is valid.
- D) It concatenates two strings and returns null if one of the strings is nullish.

Correct Answer: C) It is an optional chaining operator that allows reading the value of a property located deep within a chain of connected objects without having to expressly validate that each reference in the chain is valid.

Explanation: The optional chaining operator ?. permits the reading of the value of a property located deep within a chain of connected objects without explicitly checking that each reference in the chain is valid. It short-circuits and returns undefined if any reference is nullish (null or undefined), making code simpler and preventing runtime errors.

How does JavaScript handle module scoping?

- A) By automatically isolating all variables and functions in a module from the global scope.
- B) By requiring explicit export and import statements to share variables between modules.
- C) Both A and B are correct.
- D) JavaScript modules share the same scope as the global scope.

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

Explanation: JavaScript modules inherently provide scope isolation, meaning that variables and functions defined in a module are not visible to other modules unless they are explicitly exported. Importing these exports into another module requires the use of import statements. This system effectively prevents global scope pollution and enhances code modularity and maintainability.

What does the Promise.allSettled() method do?

- A) Waits for all promises to either resolve or reject and returns an array of objects describing the outcome of each promise.
- B) Immediately returns when the first promise is settled, either resolved or rejected.
- C) Waits for all promises to resolve and returns an array of their results.
- D) Returns a single promise that resolves when any of the input promises resolves.

Correct Answer: A) Waits for all promises to either resolve or reject and returns an array of objects describing the outcome of each promise.

Explanation: The Promise.allSettled() method returns a promise that resolves after all the given promises have either resolved or rejected, with an array of objects that each describe the outcome of each promise. This is particularly useful for handling multiple promise-returning operations that may not be critical to the overall success of the operation batch.

In JavaScript, what is the purpose of the Array.prototype.filter() method?

- A) To execute a function on each element of the array without altering the original array.
- B) To create a new array with all elements that pass the test implemented by the provided function.
- C) To reduce the array to a single value by executing a provided function on each element of the array.
- D) To update each element in an array based on a provided transformation function.

Correct Answer: B) To create a new array with all elements that pass the test implemented by the provided function.

Explanation: The filter() method creates a new array with all elements that pass the test implemented by the provided function. It's a powerful tool for creating subsets of arrays based on conditional logic, without mutating the original array.

What is the significance of using async functions with await in JavaScript?

- A) To execute asynchronous code in parallel.
- B) To write asynchronous code that behaves as if it is synchronous for easier readability and maintenance.
- C) To invoke functions synchronously, blocking code execution until they complete.
- D) To increase the performance of JavaScript code by utilizing multi-threading.

Correct Answer: B) To write asynchronous code that behaves as if it is synchronous for easier readability and maintenance.

Explanation: The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need for a chain of .then() calls. await makes JavaScript wait until the promise returns a result. It makes the asynchronous code look more like traditional synchronous code, improving readability and maintainability.

What is "tree shaking" in the context of JavaScript bundling?

- A) A process of optimizing code by dynamically loading only the necessary modules at runtime.
- B) The process of removing unused code from the final bundle to reduce its size.
- C) A technique for rearranging and optimizing the execution order of scripts for performance improvements.
- D) The act of converting ES6+ code to ES5 to ensure compatibility with older browsers.

Correct Answer: B) The process of removing unused code from the final bundle to reduce its size.

Explanation: Tree shaking is a term commonly used in the context of JavaScript tooling and bundlers like Webpack. It refers to the static analysis of imported modules to identify and eliminate code that is not actually used. This optimizes the bundle's size, improving load times and efficiency.

How can you detect whether a variable is an array in JavaScript?

- A) Using the typeof operator.
- B) Using the Array.isArray(variable) method.
- C) By checking if variable instanceof Array.
- D) Both B and C are correct methods.

Correct Answer: D) Both B and C are correct methods.

Explanation: The Array.isArray(variable) method is the most reliable way to determine whether a variable is an array, as it returns true if the variable is an array, or false otherwise. The instanceof operator can also be used for this purpose, but it may not work correctly across different execution contexts (e.g., frames or windows). Therefore, both B and C are valid methods, with B being the preferred approach for its accuracy and reliability.

What is the output of the following code snippet?

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(number => number * 2);
console.log(doubled);
```

- A) [1, 2, 3, 4]
- B) [2, 4, 6, 8]
- C) [0, 2, 4, 6]
- D) undefined

Correct Answer: B) [2, 4, 6, 8]

Explanation: The map() method creates a new array populated with the results of calling a provided function on every element in the calling array. In this case, it doubles each number in the numbers array, resulting in [2, 4, 6, 8].

What is the purpose of the static keyword in JavaScript classes?

- A) To declare a variable that retains its value after the class is instantiated.
- B) To indicate that a method or property is common to all instances of a class, rather than an individual instance.
- C) To ensure that a class method is automatically called when the class is initialized.
- D) To secure a method or property so that it cannot be overridden by subclasses.

Correct Answer: B) To indicate that a method or property is common to all instances of a class, rather than an individual instance.

Explanation: The static keyword defines a static method or property for a class. Static members (properties and methods) are called without instantiating their class and cannot be called through a class instance. Static methods are often used to implement functions that belong to the class, but not to any particular object of it.

How does JavaScript's null differ from undefined?

- A) null is a value assigned by the language itself, whereas undefined is a value assigned by developers.

- B) null represents the intentional absence of any object value, whereas undefined represents a variable that has not been assigned a value.
- C) null is used for object pointers, whereas undefined is used for primitive values.
- D) There is no significant difference; both values can be used interchangeably.

Correct Answer: B) null represents the intentional absence of any object value, whereas undefined represents a variable that has not been assigned a value.

Explanation: In JavaScript, null is an assignment value that represents the intentional absence of any object value. It is used when you want to explicitly denote the absence of a value. On the other hand, undefined is a type automatically assigned to a variable that has been declared but not yet assigned a value. They serve different purposes and are not interchangeable.

How do you ensure a function myFunc executes after a 500ms delay in JavaScript?

- A) setTimeout(myFunc(), 500);
- B) setTimeout(myFunc, 500);
- C) setInterval(myFunc, 500);
- D) myFunc.delay(500);

Correct Answer: B) setTimeout(myFunc, 500);

Explanation: setTimeout is used to execute a function or a block of code once after a specified delay, measured in milliseconds. The correct way to use it is setTimeout(myFunc, 500);, where myFunc is the reference to the function you want to delay. Passing myFunc() would execute the function immediately and pass its return value to setTimeout, which is not the intended behavior.

What will the following code snippet log to the console?
console.log(typeof null);

- A) "null"
- B) "object"
- C) "undefined"
- D) "none"

Correct Answer: B) "object"

Explanation: In JavaScript, null is considered a primitive value that represents the intentional absence of any object value. However, due to a historical bug that has been preserved for compatibility reasons, typeof null erroneously returns "object", indicating that null is mistakenly recognized as an object.

How can you stop the propagation of an event in its capturing phase in JavaScript?

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

- D) event.cancelBubble = true

Correct Answer: B) event.stopPropagation()

Explanation: The event.stopPropagation() method is used to prevent further propagation of an event in the capturing and bubbling phases. It stops the event from being propagated to parent elements, but it does not prevent the default action associated with the event.
event.stopImmediatePropagation() not only stops the event from propagating but also prevents other listeners of the same event from being called.

What is the output of the following code?

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a.length);
```

- A) 3
- B) 4
- C) undefined
- D) TypeError

Correct Answer: B) 4

Explanation: Arrays in JavaScript are reference types. When b is assigned the value of a, both a and b point to the same array in memory.
Consequently, when an element is pushed into array b, it is effectively

being added to the same array referenced by a. Therefore, the length of array a becomes 4.

In JavaScript, how do you clone an object while excluding certain properties?

- A) Using Object.assign() with a delete operator for excluded properties.
- B) Using the spread operator and explicitly setting excluded properties to undefined.
- C) Using JSON.parse(JSON.stringify(object)) and manually removing properties from the resulting object.
- D) Both A and B are correct methods.

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

Explanation: To clone an object while excluding certain properties, you can use Object.assign() or the spread operator to create a shallow copy of the object, and then use the delete operator or set excluded properties to undefined. Both methods involve creating a new object and then modifying it to exclude specific properties.

What does the following function return?

```
function checkEquality(a, b) {
return a == b;
}
```

- A) true, if a and b are the same type and value.
- B) false, if a and b are of different types.
- C) true, if a and b are the same value, coercing type if necessary.
- D) false, in all cases unless a and b are strictly equal.

Correct Answer: C) true, if a and b are the same value, coercing type if necessary.

Explanation: The == operator in JavaScript performs type coercion if the operands are of different types, attempting to convert them to a common type before making the comparison. Therefore, the function can return true for values that are conceptually equivalent, even if they are of different types (e.g., "5" == 5).

How do you convert a string to an integer in JavaScript?

- A) Using parseInt(string)
- B) Using Number(string)
- C) Using string.toInt()
- D) Both A and B are correct methods.

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

Explanation: parseInt(string) parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems). Number(string) converts the string argument to a number. Both are correct methods for converting a string to an integer, with parseInt allowing for radix specification and more forgiving parsing, while Number provides direct conversion.

What is a typical use case for the Array.prototype.some() method?

- A) To check if at least one element in the array passes the test implemented by the provided function.
- B) To execute a function on every element in the array without creating a new array.
- C) To find and return the first element in the array that satisfies the provided testing function.
- D) To create a new array with the elements that pass the test implemented by the provided function.

Correct Answer: A) To check if at least one element in the array passes the test implemented by the provided function.

Explanation: The Array.prototype.some() method tests whether at least one element in the array passes the test implemented by the provided function. It returns a Boolean value, true if the callback function returns a truthy value for any array element; otherwise, false.

In the context of asynchronous JavaScript, what does the await keyword do?

- A) It pauses the execution of async function and waits for the Promise to resolve.
- B) It immediately returns a promise and continues executing the function.
- C) It stops the execution of the program until the promise is resolved or rejected.
- D) It asynchronously executes a promise without blocking the thread.

Correct Answer: A) It pauses the execution of async function and waits for the Promise to resolve.

Explanation: Within an async function, the await keyword causes the function execution to pause until a Promise is resolved, at which point the function resumes execution and returns the resolved value. The pausing only affects the async function's flow, allowing other operations to continue running in the meantime.

How do you add an element at the beginning of an array and remove one from the end in a single operation?

- A) Using array.push() and array.shift()
- B) Using array.unshift() and array.pop()
- C) Using array.splice() with appropriate indices
- D) This operation cannot be done in a single step in JavaScript.

Correct Answer: B) Using array.unshift() and array.pop()

Explanation: To add an element at the beginning of an array, array.unshift(element) is used, and to remove an element from the end, array.pop() is used. These methods can be called successively in one line of code to achieve the operation in a single step, altering the array as required.

What will the following code output?

```
const obj = { a: 1, b: 2 };
const copy = { ...obj, b: 3 };
console.log(copy);
```

- A) { a: 1, b: 2 }
- B) { a: 1, b: 3 }

- C) { b: 3, a: 1 }
- D) SyntaxError

Correct Answer: B) { a: 1, b: 3 }

Explanation: The spread operator { ...obj } is used to copy properties from obj into copy. The property b: 3 is then defined in copy, which overwrites the original b value copied from obj. Therefore, the output is { a: 1, b: 3 }, reflecting the updated value of b.

What is the result of trying to spread a Set into an array in JavaScript?

```
const mySet = new Set([1, 2, 3]);
const myArray = [...mySet];
```

- A) TypeError because Sets cannot be spread into arrays.
- B) [1, 2, 3] because the Set's elements are spread into the array.
- C) [] because Sets are not iterable.
- D) [Set(3)] because the entire Set is added as a single element.

Correct Answer: B) [1, 2, 3] because the Set's elements are spread into the array.

Explanation: Sets are iterable, and the spread operator (...) can be used to expand the elements of a Set into an array. In this case, spreading the Set mySet which contains the elements 1, 2, 3, into an array will result in [1, 2, 3].

How do you ensure that an async function throws an error if any of the Promises it awaits are rejected?

- A) By wrapping the awaited Promises in a try/catch block.
- B) By checking the status of each Promise after the await keyword.
- C) async functions automatically throw errors for rejected Promises.
- D) By using Promise.all() with the awaited Promises.

Correct Answer: A) By wrapping the awaited Promises in a try/catch block.

Explanation: While async functions do indeed return a rejected Promise when an error is thrown, to explicitly handle errors (including rejections of awaited Promises), you should wrap the awaited Promises in a try/catch block. This approach allows you to catch and handle the rejection as an error within the function.

Which method can be used to achieve deep cloning of an object in JavaScript, including functions and circular references?

- A) JSON.parse(JSON.stringify(object))
- B) Object.assign({}, object)
- C) A custom recursive cloning function or a library like Lodash's _.cloneDeep()
- D) Spreading the original object {...object}

Correct Answer: C) A custom recursive cloning function or a library like Lodash's _.cloneDeep()

Explanation: Neither JSON.parse(JSON.stringify(object)) nor Object.assign({}, object) nor the spread operator can handle deep cloning of objects that include functions and circular references.

JSON.parse(JSON.stringify(object)) only works for JSON-compatible data types, omitting functions and throwing errors on circular references. A custom recursive cloning function or using a utility function like Lodash's _.cloneDeep() is required for deep cloning complex objects.

What is the primary use case for the Array.prototype.reduceRight() method?

- A) To reduce the array to a single value, processing each item from right to left.
- B) To reverse the array and then apply the reduce() method.
- C) To apply a function against an accumulator and each value of the array from the last to the first.
- D) Both A and C are correct.

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

Explanation: The Array.prototype.reduceRight() method is similar to Array.prototype.reduce(), but it processes the array from right to left (from the last element to the first). It is used to reduce the array to a single value by applying a function against an accumulator and each value of the array (from right to left), which can be particularly useful for operations where the direction affects the outcome.

How can you temporarily store data in a user's browser using JavaScript?

- A) localStorage.setItem('key', 'value')
- B) document.cookie = 'key=value'
- C) sessionStorage.setItem('key', 'value')

- D) All of the above

Correct Answer: D) All of the above

Explanation: JavaScript provides several ways to store data locally in a user's browser: localStorage for persistent storage across sessions, sessionStorage for storage that lasts for the duration of the page session, and cookies using document.cookie. Each method has its use case depending on the longevity and scope of data storage needed.

What will the following code snippet output?
console.log(!!"false");

- A) true
- B) false
- C) "false"
- D) TypeError

Correct Answer: A) true

Explanation: The !! operator is used to convert a value to its boolean equivalent. Since "false" is a non-empty string, it is truthy, and applying !! to it returns true. The operation effectively checks the truthiness of the string "false", which, despite its content, is considered true because it's not an empty string.

In the ECMAScript 2015 (ES6) class syntax, how do you define a static method?

- A) By prefixing the method name with the keyword static.
- B) By declaring the method outside of the class body.
- C) By using the @static decorator before the method definition.
- D) Static methods are not supported in ES6 classes.

Correct Answer: A) By prefixing the method name with the keyword static.

Explanation: In ES6, static methods are defined by prefixing the method definition with the static keyword within the class body. Static methods are called on the class itself, not on instances of the class, and are often used for utility functions that do not require a class instance.

How do you remove a specific element from the DOM using JavaScript?

- A) element.remove();
- B) document.removeChild(element);
- C) document.body.removeChild(element);
- D) element.parentNode.removeChild(element);

Correct Answer: D) element.parentNode.removeChild(element);

Explanation: The correct way to remove a specific element from the DOM is to call removeChild() on the parent node of the element you wish to remove, passing in the element as an argument. The element.remove(); method is also valid for directly removing an element without needing to reference its parent node, making both A and D correct in modern browsers.

What is the purpose of the async attribute on a <script> tag?

- A) To defer the execution of the script until after the page has loaded.
- B) To specify that the script will execute asynchronously, as soon as it is available.
- C) To force the script to load synchronously in the order it appears in the document.
- D) To indicate that the script contains asynchronous functions or awaits.

Correct Answer: B) To specify that the script will execute asynchronously, as soon as it is available.

Explanation: The async attribute on a <script> tag tells the browser to execute the script asynchronously, meaning it will be downloaded in the background and executed as soon as it’s downloaded, without blocking the parsing of the rest of the page. This is different from the defer attribute, which also downloads scripts in the background but defers their execution until after the document has been parsed.

Which statement correctly checks if the browser supports Web Workers?

- A) if (window.Worker) { ... }
- B) if (navigator.worker) { ... }
- C) if (typeof Worker !== 'undefined') { ... }
- D) Both A and C are correct.

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

Explanation: You can check for Web Workers support by verifying if the Worker constructor exists. This can be done either by checking if (window.Worker) { ... } to see if the Worker property exists on the window object, or if (typeof Worker !== 'undefined') { ... } to check the type of the Worker. Both methods effectively determine if the current browser environment supports Web Workers.

How can you prevent a form submission using JavaScript?

- A) Using event.cancel()
- B) Using event.preventDefault() in the form's submit event handler
- C) Using return false in the form's onsubmit attribute
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: To prevent a form from being submitted, you can either use event.preventDefault() within the form's submit event handler or return false from the inline onsubmit attribute of the form. Both methods will stop the form from submitting and allow you to perform additional validation or processing before submission.

What does the finally clause in a Promise chain do?

- A) Executes code after the Promise is either resolved or rejected
- B) Replaces the need for both .then() and .catch() methods
- C) Ensures that the Promise is resolved
- D) Executes only if the Promise is resolved

Correct Answer: A) Executes code after the Promise is either resolved or rejected

Explanation: The finally clause in a Promise chain is used to execute code after the Promise has either been resolved or rejected, regardless of the outcome. It is typically used for cleanup or finishing tasks, such as hiding loading indicators, that need to be done whether the Promise was fulfilled or rejected.

What is the default value of the position property in CSS for an element targeted by document.getElementById() in JavaScript?

- A) static
- B) relative
- C) absolute
- D) fixed

Correct Answer: A) static

Explanation: The default value of the position property in CSS is static. This means that, unless explicitly changed, elements targeted by document.getElementById() or any other method will have a position of static, causing them to flow into the normal document layout without respect to the positioning properties (top, right, bottom, left).

How can you check if an object is an instance of a specific class in JavaScript?

- A) Using the instanceof operator

- B) Using the typeof operator
- C) Checking if the object's prototype is the class prototype
- D) Comparing the object's constructor property with the class

Correct Answer: A) Using the instanceof operator

Explanation: The instanceof operator in JavaScript is used to check if an object is an instance of a specific class or constructor function. It examines the prototype chain of the object for the presence of the prototype property of the constructor, making it a reliable way to determine instance relationships.

In JavaScript, how do you find the character at a specific index in a string?

- A) string.charAt(index)
- B) string[index]
- C) string.charCodeAt(index)
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Both string.charAt(index) and the bracket notation string[index] can be used to access the character at a specific index in a string in JavaScript. While charAt is a method specifically designed for this purpose, the bracket notation is a more general property access syntax that also works for strings.

Which method would you use to schedule a function to execute repeatedly at specified intervals?

- A) setTimeout()
- B) setInterval()
- C) requestAnimationFrame()
- D) repeat()

Correct Answer: B) setInterval()

Explanation: setInterval() is used to schedule a function to be called repeatedly at specified intervals, in milliseconds. Unlike setTimeout(), which executes a function once after a delay, setInterval() continues to call the function until it is explicitly stopped with clearInterval().

How do you remove all child elements from a DOM node in JavaScript?

- A) node.removeChild()
- B) node.innerHTML = ''
- C) while(node.firstChild) node.removeChild(node.firstChild)
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: Setting node.innerHTML = '' effectively removes all child elements of a DOM node by clearing its HTML content. Alternatively, a loop like while(node.firstChild) node.removeChild(node.firstChild) can be used to remove each child node one by one until none remain. Both methods achieve the goal of removing all child elements from a node.

What does the Array.prototype.map() method return?

- A) The original array, modified by the function passed to map()
- B) A new array, with each element being the result of the function passed to map()
- C) The number of elements processed by the function
- D) undefined

Correct Answer: B) A new array, with each element being the result of the function passed to map()

Explanation: The Array.prototype.map() method creates a new array populated with the results of calling a provided function on every element in the calling array. It does not modify the original array but instead returns a new array, making it a pure function useful for transformations of array data.

In JavaScript, what is the significance of using document.createDocumentFragment()?

- A) It creates a new HTML document.
- B) It is used to parse HTML strings into DOM nodes.
- C) It creates a lightweight, non-DOM node container for temporary storage of DOM elements.
- D) It initializes a new Document object for XML parsing.

Correct Answer: C) It creates a lightweight, non-DOM node container for temporary storage of DOM elements.

Explanation: document.createDocumentFragment() creates a document fragment, which acts as a lightweight container for storing DOM elements temporarily. Document fragments can be used to assemble changes off-DOM, improving performance by minimizing reflows and repaints when manipulating the document.

How do you encode a URL component in JavaScript?

- A) encodeURI(component)
- B) encodeURIComponent(component)
- C) escape(component)
- D) URLEncode(component)

Correct Answer: B) encodeURIComponent(component)

Explanation: The encodeURIComponent() function is used to encode a URI component by escaping certain characters. It encodes special characters, including URI delimiters, ensuring that the component can be safely included in URL query parameters. Unlike encodeURI(), which encodes a full URI, encodeURIComponent() is intended for individual URI components, such as query string values.

How can you determine if a variable is an instance of a specific constructor in JavaScript?

- A) Using the instanceof operator
- B) Using the typeof operator
- C) Checking the constructor property: variable.constructor === ConstructorName
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: The instanceof operator in JavaScript is used to test whether the prototype property of a constructor appears anywhere in the prototype chain of an object. The variable.constructor === ConstructorName check compares the constructor reference directly, but it can be unreliable if the prototype chain is altered. While both can be used to check an instance, instanceof is generally more robust due to its ability to handle prototype chain complexities.

What is the purpose of the Array.prototype.slice() method?

- A) To remove elements from an array and, optionally, replace them
- B) To extract a shallow copy of a portion of an array into a new array object
- C) To concatenate two or more arrays
- D) To iterate over an array and execute a function on each element

Correct Answer: B) To extract a shallow copy of a portion of an array into a new array object

Explanation: Array.prototype.slice() is used to return a shallow copy of a portion of an array into a new array object without modifying the original array. It can take two arguments that specify the start and end indices of the slice, where the end index is not included in the result.

How do you add an element to the end of an array and remove the first element in one operation?

- A) array.push(element); array.shift();
- B) array.unshift(element); array.pop();

- C) array.splice(0, 1, element);
- D) array.push(element); array.splice(0, 1);

Correct Answer: D) array.push(element); array.splice(0, 1);

Explanation: To add an element to the end of an array, array.push(element) is used. To remove the first element, array.shift() would normally be used, but since the question asks for a single operation, combining push to add to the end, followed by splice(0, 1) to remove the first element, achieves the desired effect in what can be considered a single logical operation split into two method calls.

In JavaScript, what will the following code snippet output?
console.log(1 + '2' + 3);

- A) 6
- B) 123
- C) 15
- D) undefined

Correct Answer: B) 123

Explanation: In this expression, the number 1 is concatenated with the string '2', resulting in the string '12'. Then, '12' is concatenated with the number 3, resulting in the string '123'. JavaScript performs type coercion, converting numbers to strings when using the + operator with a string operand.

How do you make a shallow copy of an object in JavaScript?

- A) Using Object.copy()
- B) Using Object.assign({}, obj)
- C) Using the spread operator { ...obj }
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: Both Object.assign({}, obj) and the spread operator { ...obj } are used to create a shallow copy of an object obj. These methods copy the properties from the original object to a new object, but they do not deeply clone objects nested within obj.

What does the Array.prototype.every() method do?

- A) Checks if at least one element in the array passes the test implemented by the provided function.
- B) Checks if every element in the array passes the test implemented by the provided function.
- C) Executes a provided function once for each array element.
- D) Reduces the array to a single value by executing a reducer function on each element.

Correct Answer: B) Checks if every element in the array passes the test implemented by the provided function.

Explanation: The Array.prototype.every() method tests whether all elements in the array pass the test implemented by the provided function. It returns a

Boolean value - true if all elements satisfy the condition, and false otherwise.

How can you stop the default action of an event without stopping the event from bubbling up in JavaScript?

- A) event.stopImmediatePropagation()
- B) event.stopPropagation()
- C) event.preventDefault()
- D) event.cancelBubble = true

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 from propagating through the DOM. This allows you to disable default behavior like clicking on a link without preventing other event handlers from being executed.

How do you create a script that runs only after the DOM has fully loaded in JavaScript?

- A) Add async attribute to the script tag
- B) Use document.onload = function() { ... }
- C) Use window.onload = function() { ... }
- D) Use document.addEventListener('DOMContentLoaded', function() { ... })

Correct Answer: D) Use document.addEventListener('DOMContentLoaded', function() { ... })

Explanation: The DOMContentLoaded event fires when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading. document.addEventListener('DOMContentLoaded', function() { ... }) is the recommended way to run code after the DOM is ready, ensuring that DOM elements are accessible.

What is the purpose of using Symbol in JavaScript?

- A) To create unique identifiers for object properties
- B) To define constants with symbolic names
- C) To provide symbols for built-in methods and properties
- D) All of the above

Correct Answer: D) All of the above

Explanation: Symbol is a primitive data type introduced in ECMAScript 2015 (ES6) that is used to create unique identifiers for object properties, ensuring they are unique and non-conflicting. Symbols can also define constants with unique values and are used by JavaScript's internal mechanisms to provide symbols for built-in methods and properties, like Symbol.iterator for iteration protocols.

How can you reverse a string in JavaScript?

- A) Using the reverse() method on the string
- B) Splitting the string into an array, using reverse(), and then joining the array
- C) Using a for loop to create a new reversed string

- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: JavaScript strings do not have a reverse() method. To reverse a string, you can first split it into an array of characters using string.split(''), then reverse the array using array.reverse(), and finally join the array back into a string with array.join(''). Alternatively, you can use a for loop to iterate over the string in reverse order and construct a new reversed string.

How do you ensure a piece of JavaScript runs after the page has fully loaded including images and stylesheets?

- A) document.onload = function() { ... }
- B) window.onload = function() { ... }
- C) document.addEventListener('DOMContentLoaded', function() { ... })
- D) document.addEventListener('load', function() { ... })

Correct Answer: B) window.onload = function() { ... }

Explanation: The window.onload event handler executes a block of code after the entire page, including all dependent resources such as images and stylesheets, has loaded. Unlike DOMContentLoaded, which triggers as soon as the HTML is fully parsed, window.onload waits for everything to load, making it suitable for operations that depend on fully loaded resources.

What is the output of the following JavaScript code?

```
const numbers = [1, 2, 3, 4, 5];
const [first, , third] = numbers;
console.log(first, third);
```

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

Correct Answer: B) 1 3

Explanation: This code snippet demonstrates array destructuring, where first and third variables are assigned the first and third elements of the numbers array, respectively. The second element is skipped by leaving an empty space in the destructuring assignment, resulting in 1 and 3 being logged to the console.

How can you convert an array-like object into an array in JavaScript?

- A) Array.from(arrayLikeObject)
- B) arrayLikeObject.toArray()
- C) [...arrayLikeObject]
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: Array.from() method creates a new, shallow-copied Array instance from an array-like or iterable object. The spread operator (...) can also be used to expand iterable or array-like objects into an array. Both methods are effective for converting array-like objects (e.g., NodeList, arguments object) into true array instances.

How do you define a property on an object that cannot be enumerated in a for...in loop or appear in the object's keys array?

- A) Using Object.defineProperty(obj, 'prop', { value: value })
- B) obj.prop = value; obj.prop.enumerable = false;
- C) obj.defineProperty('prop', value, false);
- D) Object.setNonEnumerable(obj, 'prop', value);

Correct Answer: A) Using Object.defineProperty(obj, 'prop', { value: value })

Explanation: Object.defineProperty() allows detailed control over a property's configuration. By default, properties added this way are not enumerable, not writable, and not configurable unless otherwise specified. This method is used to define a property with enumerable set to false so that it won't appear in for...in loops or in the object's keys array.

What will the following code snippet output?

```
function multiply(x, y) {
y = typeof y === 'undefined' ? 1 : y;
return x * y;
}
console.log(multiply(5));
```

- A) NaN
- B) 5
- C) undefined
- D) 0

Correct Answer: B) 5

Explanation: In the multiply function, the parameter y is given a default value of 1 if it's not passed or undefined. When calling multiply(5), y defaults to 1, and the function returns 5 * 1, which is 5.

In JavaScript, how do you copy text to the clipboard?

- A) document.clipboard.writeText('text')
- B) navigator.clipboard.writeText('text')
- C) Clipboard.writeText('text')
- D) window.clipboardData.setData('Text', 'text')

Correct Answer: B) navigator.clipboard.writeText('text')

Explanation: The modern and recommended way to programmatically copy text to the clipboard in JavaScript is using the navigator.clipboard.writeText('text') method. It returns a promise that resolves once the text has been copied to the clipboard. This API provides a more secure and flexible way to access the clipboard.

What is the use of the document.createDocumentFragment() method?

- A) To create a new empty Document object for XML parsing.
- B) To create a new DocumentFragment object that can contain DOM nodes.
- C) To parse HTML strings and convert them into DOM nodes.
- D) To clone an existing document.

Correct Answer: B) To create a new DocumentFragment object that can contain DOM nodes.

Explanation: document.createDocumentFragment() creates a lightweight, document-like container (DocumentFragment) that can hold DOM nodes. You can append nodes to the fragment without causing reflow and repaint in the document, making it a performant way to build up a set of DOM changes and apply them all at once.

How do you create a read-only property in an object?

- A) Using Object.freeze(obj)
- B) Using Object.defineProperty(obj, 'prop', { value: 'value', writable: false })
- C) obj.prop = 'value'; delete obj.prop;
- D) const obj = { readonly prop: 'value' };

Correct Answer: B) Using Object.defineProperty(obj, 'prop', { value: 'value', writable: false })

Explanation: Object.defineProperty() allows the configuration of property descriptors, including writable. Setting writable: false makes the property read-only. Object.freeze(obj) makes all properties of obj read-only, but it applies to the entire object, not individual properties.

What does the void operator do in JavaScript?

- A) Deletes properties from an object.

- B) Returns undefined from any operation without affecting the surrounding expression.
- C) Immediately terminates the execution of a function.
- D) Creates a void block that does not return any value.

Correct Answer: B) Returns undefined from any operation without affecting the surrounding expression.

Explanation: The void operator evaluates the given expression and then returns undefined. This can be useful in cases where you want to explicitly return undefined from an expression, or to ensure an anchor tag (<a href="javascript:void(0);">) does nothing when clicked.

In JavaScript, what does the new operator do?

- A) Creates a new empty object.
- B) Instantiates a new object from a constructor function or class.
- C) Allocates memory for a new variable.
- D) Clones an existing object.

Correct Answer: B) Instantiates a new object from a constructor function or class.

Explanation: The new operator creates an instance of an object that has a constructor function or belongs to a class. It creates a new object, sets the object's prototype to the constructor's prototype property, executes the constructor with the given arguments, and returns the object (unless the constructor returns a non-primitive value, which would be returned instead).

How can you programmatically force a browser to navigate to a new page using JavaScript?

- A) document.location.href = 'http://newpage.com'
- B) window.navigate('http://newpage.com')
- C) browser.redirect('http://newpage.com')
- D) location.assign('http://newpage.com')

Correct Answer: D) location.assign('http://newpage.com')

Explanation: The location.assign() method loads a new document, effectively navigating to the specified URL. While setting document.location.href or window.location to a new URL also results in page navigation, location.assign() is the explicit method intended for this purpose.

Which JavaScript event is fired when the document has been completely loaded, including all scripts, images, and stylesheets?

- A) DOMContentLoaded
- B) documentLoaded
- C) load
- D) complete

Correct Answer: C) load

Explanation: The load event is fired on the window object when the whole page has loaded, including all dependent resources such as stylesheets and images. This is different from the DOMContentLoaded event, which is fired as soon as the HTML document has been fully parsed, without waiting for stylesheets, images, and subframes to finish loading.

How do you achieve inheritance in JavaScript?

- A) Using the extends keyword in class declarations
- B) Setting the prototype of one constructor function to an instance of another
- C) Using the inherit() method on the parent object
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Inheritance in JavaScript can be achieved through classical inheritance using the extends keyword in ES6 class syntax, where one class can extend another. For more traditional prototype-based inheritance, the prototype of one constructor function can be set to an instance of another, effectively creating a prototype chain.

What method would you use to parse a JSON string into a JavaScript object?

- A) JSON.parse(jsonString)
- B) JSON.stringify(jsonString)
- C) Object.parse(jsonString)
- D) Object.fromString(jsonString)

Correct Answer: A) JSON.parse(jsonString)

Explanation: JSON.parse() is used to parse a JSON string, constructing the JavaScript value or object described by the string. Conversely, JSON.stringify() converts a JavaScript object or value to a JSON string.

How can you detect a user's device type in JavaScript?

- A) Using the navigator.device property
- B) Using the navigator.userAgent property
- C) Using window.deviceType
- D) Using screen.type

Correct Answer: B) Using the navigator.userAgent property

Explanation: The navigator.userAgent property returns a string representing the user agent string for the browser. This string can be analyzed to infer information about the device, browser, and operating system, although it requires parsing and can be spoofed or inaccurate in some environments.

How do you stop a CSS animation in JavaScript?

- A) element.style.animation = "none";
- B) element.animation.pause();
- C) element.style.animationPlayState = "paused";
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: To stop a CSS animation on an element, you can either remove the animation by setting element.style.animation to "none" or pause it by setting element.style.animationPlayState to "paused". The latter allows the animation to be resumed from where it was paused.

In JavaScript, how do you check if an object is empty?

- A) Object.isEmpty(obj)
- B) Object.keys(obj).length === 0
- C) obj.length === 0

- D) obj === {}

Correct Answer: B) Object.keys(obj).length === 0

Explanation: Object.keys(obj) returns an array of a given object's own enumerable property names. Checking if the length of this array is 0 is a reliable way to determine if an object has no own properties, and thus is "empty". The other options provided are not valid methods for checking if an object is empty.

What is the difference between localStorage and sessionStorage in web storage API?

- A) localStorage persists data between browser sessions; sessionStorage data is cleared when the session ends
- B) localStorage has a larger storage limit than sessionStorage
- C) sessionStorage is shared between tabs and windows; localStorage is not
- D) localStorage is synchronous; sessionStorage is asynchronous

Correct Answer: A) localStorage persists data between browser sessions; sessionStorage data is cleared when the session ends

Explanation: The main difference between localStorage and sessionStorage pertains to the lifetime of the data stored. Data in localStorage persists across browser sessions, meaning the data is still accessible after closing and reopening the browser. sessionStorage data is limited to the window or tab's lifetime and is cleared when the tab or window is closed.

How do you remove duplicate values from an array in JavaScript?

- A) Array.from(new Set(array))
- B) array.unique()
- C) array.filter((item, index) => array.indexOf(item) === index)
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: Creating a Set from an array automatically removes duplicates because a Set can only contain unique values. Array.from(new Set(array)) converts it back into an array without duplicates. Alternatively, array.filter((item, index) => array.indexOf(item) === index) can be used to achieve the same result by explicitly filtering out duplicate values.

How can you defer the execution of a script in an HTML document?

- A) Adding the defer attribute to the <script> tag
- B) Using setTimeout() in the script
- C) Placing the script at the bottom of the <body> section
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: The defer attribute in a <script> tag tells the browser to execute the script file after the document has been parsed, but before firing DOMContentLoaded. Placing the script at the bottom of the <body> is a common technique to ensure it runs after the HTML content above it has loaded, though it doesn't guarantee execution after the entire document is parsed like defer does.

What method would you use to merge two arrays without creating duplicate values in JavaScript?

- A) array1.concat(array2)
- B) [...new Set([...array1, ...array2])]
- C) array1.push(...array2)
- D) Array.merge(array1, array2)

Correct Answer: B) [...new Set([...array1, ...array2])]

Explanation: By combining the spread operator (...) with the Set constructor, you can merge two arrays and remove duplicates in the process. A Set is a collection of unique values, so converting the arrays to a set and then back to an array ensures that all values are unique.

How can you dynamically import a module in JavaScript?

- A) import(moduleName)
- B) require(moduleName)
- C) import(moduleName).then(module => {})
- D) import(moduleName).then(module => {})

Correct Answer: D) import(moduleName).then(module => {})

Explanation: Dynamic imports in JavaScript are performed using the import() function, which returns a promise. This makes it possible to load modules on demand or conditionally, enhancing performance and resource utilization by loading code only when it's needed.

What will the following code snippet output?

```
let x = { y: 1 };
let z = { ...x };
z.y = 2;
console.log(x.y);
```

- A) 1
- B) 2
- C) undefined
- D) ReferenceError

Correct Answer: A) 1

Explanation: The spread operator ({...x}) is used to create a shallow copy of the object x and assign it to z. Modifying z.y does not affect the original object x since z is a separate object. Therefore, console.log(x.y) outputs 1.

In JavaScript, how do you convert a string to a number?

- A) Number.parseInt(string)
- B) Number(string)
- C) string.toNumber()
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Number(string) converts the string argument to a number, including floating-point numbers. Number.parseInt(string) specifically parses a string argument and returns an integer of the specified radix or base. Both methods are correct for converting strings to numbers, with the choice depending on whether you need a floating-point or an integer result.

What is the purpose of template literals in JavaScript?

- A) To define complex objects
- B) To facilitate mathematical operations
- C) To create multi-line strings and incorporate expressions
- D) To declare functions in a concise format

Correct Answer: C) To create multi-line strings and incorporate expressions

Explanation: Template literals provide an easy syntax to create multi-line strings and to embed expressions within strings. Enclosed by backticks ( ), they can contain placeholders for embedding expressions, which are indicated by ${expression} syntax. This feature simplifies string concatenation and dynamic content generation.

How do you remove a property from an object in JavaScript?

- A) delete object.propertyName
- B) object.propertyName.remove()
- C) object.remove(propertyName)
- D) object.propertyName = undefined

Correct Answer: A) delete object.propertyName

Explanation: The delete operator removes a property from an object. It directly modifies the object by removing the specified property if it exists.

Setting a property to undefined does not remove the property; it just changes its value.

How can you stop further propagation of an event currently being handled?

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

Correct Answer: B) event.stopPropagation()

Explanation: The event.stopPropagation() method is used to prevent further propagation of the current event in the capturing and bubbling phases. It stops the event from being propagated to parent elements, but it does not prevent the default action associated with the event.

What does the async keyword do for a function in JavaScript?

- A) Makes it run synchronously
- B) Allows it to return a promise
- C) Ensures it runs in a separate thread
- D) Allows it to use the await keyword within its body

Correct Answer: D) Allows it to use the await keyword within its body

Explanation: The async keyword before a function definition makes the function return a promise and allows the use of the await keyword inside the function to pause execution on asynchronous operations until they are resolved, simplifying the handling of promises.

What will the following code snippet output?
const promise = Promise.resolve(5);
promise.then(value => console.log(value));

- A) 5
- B) Promise {<resolved>: 5}
- C) undefined
- D) TypeError

Correct Answer: A) 5

Explanation: The Promise.resolve(value) method returns a promise that is resolved with the given value. When .then() is called on this promise, the callback function receives the resolved value (5) as its argument, which is then logged to the console.

How can you clone an array in JavaScript?
- A) const clone = array.clone()
- B) const clone = [...array]
- C) const clone = array.slice()
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: Both the spread operator ([...array]) and the slice() method can be used to create a shallow clone of an array. These methods copy the elements of the original array into a new array, allowing modifications to the clone without affecting the original array.

How do you ensure that a function always receives a particular context for this, regardless of how it is called?

- A) function.bind(context)
- B) function.call(context)
- C) function.apply(context)
- D) function.context = context

Correct Answer: A) function.bind(context)

Explanation: The bind method creates a new function that, when called, has its this keyword set to the provided value (context). Unlike call and apply, which immediately invoke the function with the specified this context, bind returns a new function with the context permanently bound, making it ideal for ensuring a function receives a particular this context regardless of how it is called.

What feature of ES6+ allows for the concise declaration of anonymous functions?

- A) Arrow functions
- B) The function keyword
- C) Template literals
- D) Destructuring assignments

Correct Answer: A) Arrow functions

Explanation: Arrow functions provide a concise syntax for writing anonymous functions. They allow for shorter function syntax and lexically bind the this value, making them particularly useful for callbacks and functional programming patterns. Arrow functions do not have their own this, arguments, super, or new.target bindings.

What is the output of the following code snippet?

```
let x = 10;
let y = "10";
console.log(x === y);
```

- A) true
- B) false
- C) "10"
- D) TypeError

Correct Answer: B) false

Explanation: The strict equality operator (===) checks both the value and the type of its operands. In this case, x is a number and y is a string. Even though they have the same value ("10"), their types are different, resulting in false.

In JavaScript, which method is used to asynchronously fetch data from the server without reloading the page?

- A) XMLHttpRequest
- B) fetch
- C) axios.get
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Both XMLHttpRequest and the fetch API can be used to asynchronously request data from the server without reloading the page. fetch provides a more modern, promise-based approach to handling asynchronous HTTP requests, while XMLHttpRequest is the older standard. axios.get is also a valid method but requires the Axios library, which wasn't listed as an option here.

How do you create a read-only date object in JavaScript?

- A) JavaScript does not support read-only date objects.
- B) Using Object.freeze(new Date())
- C) const date = new Date(); date.readOnly = true;
- D) new Date().asReadOnly()

Correct Answer: A) JavaScript does not support read-only date objects.

Explanation: JavaScript does not have built-in support for creating read-only date objects. While Object.freeze() can prevent modification to the properties of an object, it does not prevent the modification of mutable objects contained within, such as the state of a Date object. Therefore, there is no native, foolproof method to create a read-only Date object in JavaScript.

What does the Array.prototype.find() method return if no elements match the provided testing function?

- A) null
- B) undefined
- C) 0
- D) An empty array []

Correct Answer: B) undefined

Explanation: The Array.prototype.find() method returns the value of the first element in the provided array that satisfies the provided testing function. If no elements in the array satisfy the testing function, the method returns undefined.

How can you achieve module encapsulation and privacy in JavaScript?

- A) Using immediately invoked function expressions (IIFE)
- B) Declaring all variables with the const keyword
- C) Placing the code inside a class
- D) Using the # prefix for private variables in a class

Correct Answer: A) Using immediately invoked function expressions (IIFE)

Explanation: Before the introduction of ES6 modules and class fields, one common way to achieve module encapsulation and privacy in JavaScript was through the use of IIFEs. An IIFE is a function that is executed as soon as it is defined, creating a new scope for its variables and functions, thus preventing them from polluting the global scope and achieving a level of privacy. With the advent of ES6, classes with private fields (using the # prefix) also provide a means for encapsulation and privacy.

What will the following JavaScript code output to the console?

```
const obj = {
a: 1,
b: 2,
a: 3
};
console.log(obj.a);
```

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

Correct Answer: C) 3

Explanation: In JavaScript objects, property keys are unique, and if you define the same key multiple times, the last assignment is the one that takes effect. In the given code snippet, the property a is defined twice, with the last assignment being a: 3. Therefore, console.log(obj.a); will output 3.

How do you ensure that a block of code is executed after a certain delay in JavaScript?

- A) setTimeout(codeBlock, delay)
- B) setInterval(codeBlock, delay)
- C) time.sleep(delay); execute(codeBlock);
- D) Promise.after(delay).then(codeBlock)

Correct Answer: A) setTimeout(codeBlock, delay)

Explanation: The setTimeout() function is used to execute a block of code or a function after a specified delay in milliseconds. It runs the code block or function exactly once after the delay. setInterval(), on the other hand, repeatedly executes the code block or function with the delay between each execution, until it is stopped with clearInterval().

What is the primary use of the ?. operator in JavaScript?

- A) To specify optional function parameters
- B) To perform nullish coalescing
- C) To execute a method or access a property if it exists, without causing an error if it doesn't
- D) To concatenate strings in a null-safe way

Correct Answer: C) To execute a method or access a property if it exists, without causing an error if it doesn't

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

How do you copy the properties of one object to another in JavaScript?

- A) Object.assign(target, source)
- B) target = { ...source }
- C) $.extend(target, source)

- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Object.assign(target, source) is used to copy the values of all enumerable own properties from one or more source objects to a target object. The spread operator { ...source } is used within object literals to achieve a similar result, creating a new object with properties copied from the source. Both methods effectively copy properties from the source to the target.

What is the result of trying to access an undeclared variable in JavaScript?

- A) undefined
- B) null
- C) ReferenceError
- D) false

Correct Answer: C) ReferenceError

Explanation: Accessing an undeclared variable in JavaScript results in a ReferenceError because the variable is not defined in the accessible scope. Unlike accessing a property of an undefined object, which returns undefined, trying to use a completely undeclared variable is not allowed and triggers an error.

Which method is used to encode a URI component in JavaScript?

- A) encodeURI(component)
- B) encodeURIComponent(component)
- C) escape(component)
- D) URIEncode(component)

Correct Answer: B) encodeURIComponent(component)

Explanation: encodeURIComponent(component) is used to encode a URI component by escaping all characters except alphabetic, decimal digits, and a few special characters. This function is more comprehensive than encodeURI because it encodes characters such as =, &, ?, which could have special meanings in URIs.

How do you define a constant array in JavaScript that cannot be modified?

- A) const arr = Object.freeze([1, 2, 3])
- B) const arr = [1, 2, 3]
- C) var arr = [1, 2, 3].freeze()
- D) let arr = const [1, 2, 3]

Correct Answer: A) const arr = Object.freeze([1, 2, 3])

Explanation: While declaring an array with const prevents reassignment of the variable arr, it does not prevent modification of the array itself. Object.freeze([1, 2, 3]) makes the array immutable, preventing any changes to its items or structure, achieving the effect of a constant array.

What is the purpose of the Map object in JavaScript?

- A) To store key-value pairs where keys are strictly numeric
- B) To map values to specific functions
- C) To store key-value pairs with any value as both keys and values
- D) To provide a built-in mechanism for iterating over object properties

Correct Answer: C) To store key-value pairs with any value as both keys and values

Explanation: The Map object holds key-value pairs and remembers the original insertion order of the keys. Unlike objects, a Map can have any value as both keys and values, offering more flexibility and specific use cases such as preserving insertion order.

How can you detect if the code is running in strict mode in JavaScript?

- A) Checking the value of this in the global scope, which is undefined in strict mode
- B) Using isStrictMode() global function
- C) if ("strict" in this) { ... }
- D) Strict mode cannot be detected programmatically

Correct Answer: A) Checking the value of this in the global scope, which is undefined in strict mode

Explanation: In strict mode, the value of this in functions that are called in the global scope is undefined, unlike in non-strict mode, where this refers to the global object. This behavior can be used to check for strict mode, although it's an indirect method and might not cover all use cases or scopes.

What will the following JavaScript expression return?
"Cat" && "Dog"

- A) true
- B) false
- C) "Cat"
- D) "Dog"

Correct Answer: D) "Dog"

Explanation: The logical AND (&&) operator returns the first falsy value found when evaluated from left to right; if all values are truthy, the last value is returned. Since both "Cat" and "Dog" are truthy, the expression returns "Dog", the last value.

How do you round a number to two decimal places in JavaScript?
- A) Math.round(num, 2)
- B) Number(num.toFixed(2))
- C) Math.floor(num * 100) / 100
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: num.toFixed(2) converts a number to a string, rounding it to 2 decimal places. Wrapping this with Number() converts the string back to a numeric type. Alternatively, multiplying the number by 100, using Math.floor or Math.round on the result, and then dividing by 100 rounds the number to two decimal places. Both methods achieve the desired rounding effect.

In JavaScript, how do you check if a number is finite and not NaN?

- A) isFinite(number)
- B) Number.isFinite(number)
- C) number.isFinite()
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: isFinite(number) is a global function that returns false if the argument is NaN, +Infinity, -Infinity, or not a number; otherwise, it returns true. Number.isFinite(number) is a more precise method introduced in ES6 that does not coerce the argument to a number, returning true only if the argument is of the type Number and finite.

How can you temporarily disable a button in HTML using JavaScript?

- A) document.getElementById('myButton').disabled = true;
- B) $('#myButton').disable();
- C) document.getElementById('myButton').setAttribute('disabled', 'disabled');
- D) Both A and C are correct

Correct Answer: D) Both A and C are correct

Explanation: Setting the disabled property of a button to true directly in JavaScript or using setAttribute to add the disabled attribute both achieve the effect of disabling the button. While option B resembles a jQuery approach, it's not correctly stated (disable() is not a valid jQuery method for this purpose).

How can you remove all falsy values from an array in JavaScript?

- A) array.filter(value => value)
- B) array.map(value => !!value)
- C) array.removeAll(false)
- D) array.compact()

Correct Answer: A) array.filter(value => value)

Explanation: The filter method creates a new array with all elements that pass the test implemented by the provided function. By simply passing value => value, the filter callback leverages the truthiness of array values, effectively removing all falsy values (false, null, undefined, 0, "", and NaN) from the array.

What will the following JavaScript expression return?
typeof typeof 0

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

Correct Answer: B) "string"

Explanation: The inner typeof 0 returns "number", which is a string. Then, the outer typeof operates on the result "number", which is of type string, so it returns "string".

How do you concatenate two arrays in JavaScript?

- A) array1.concat(array2)
- B) array1 + array2
- C) array1.push(array2)
- D) ...array1, ...array2

Correct Answer: A) array1.concat(array2)

Explanation: The concat method is used to merge two or more arrays by returning a new array without changing the existing arrays. While ...array1, ...array2 is a valid syntax for spreading elements into a new array [...array1, ...array2], as presented, it's syntactically incorrect outside of an array or function argument context.

How do you create a new date object representing the current date and time in JavaScript?

- A) new Date()
- B) Date.now()
- C) Date.getCurrent()
- D) new Date().now()

Correct Answer: A) new Date()

Explanation: new Date() creates a new date object with the current date and time. Date.now() returns the number of milliseconds elapsed since January 1, 1970 00:00:00 UTC, but it does not create a Date object.

What is the output of the following code snippet?

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a.length);
```

- A) 3
- B) 4
- C) undefined
- D) TypeError

Correct Answer: B) 4

Explanation: Arrays in JavaScript are reference types, so a and b point to the same array object in memory. When b.push(4) is executed, it modifies the array by adding a new element, affecting both a and b. Thus, a.length returns 4.

In JavaScript, which method is best suited for transforming the values of an array and accumulating the results in a single value?

- A) Array.prototype.reduce()
- B) Array.prototype.map()
- C) Array.prototype.forEach()
- D) Array.prototype.filter()

Correct Answer: A) Array.prototype.reduce()

Explanation: The reduce method applies a function against an accumulator and each value of the array (from left to right) to reduce it to a single value. This makes it well-suited for transformations that accumulate results, such as summing values.

What does the Event.preventDefault() method do?

- A) Stops the event from bubbling up the event chain
- B) Prevents the default action associated with the event
- C) Immediately stops the event's propagation
- D) Cancels the event if it is cancelable, without stopping further propagation

Correct Answer: B) Prevents the default action associated with the event

Explanation: The Event.preventDefault() method is used to prevent the browser's default action for the event, without stopping the event from propagating through the DOM. It's commonly used in form submission handling or to prevent links from navigating to a URL.

How do you check if an object obj is an array in JavaScript?

- A) obj.isArray()
- B) Array.isArray(obj)
- C) obj instanceof Array
- D) Both B and C are correct

Correct Answer: D) Both B and C are correct

Explanation: Array.isArray(obj) is a standard method for checking if an object is an array. obj instanceof Array also checks if obj inherits from Array.prototype, which can be true for arrays. However, Array.isArray() is more reliable, especially when dealing with frames or windows where the prototype chain may differ.

How can you delay the execution of a function in JavaScript?

- A) setTimeout(function, milliseconds)
- B) setInterval(function, milliseconds)
- C) function.delay(milliseconds)
- D) wait(function, milliseconds)

Correct Answer: A) setTimeout(function, milliseconds)

Explanation: The setTimeout() function is used to execute a function or a block of code once after a specified delay in milliseconds. It's the standard way to delay function execution in JavaScript.

How do you add an event listener that runs only once to an element in JavaScript?

- A) element.addEventListener('click', handler, true)
- B) element.addEventListener('click', handler, {once: true})
- C) element.one('click', handler)
- D) element.addEventListener('click', handler).once()

Correct Answer: B) element.addEventListener('click', handler, {once: true})

Explanation: The addEventListener method accepts an options object as its third argument, where you can set once: true to indicate that the listener should be invoked at most once after being added. If true, the listener would be automatically removed when invoked.

Which statement correctly creates a JavaScript Promise that resolves to "Hello" after 2 seconds?

- A) new Promise((resolve) => setTimeout(() => resolve("Hello"), 2000))
- B) Promise.resolve(setTimeout("Hello", 2000))
- C) setTimeout(() => Promise.resolve("Hello"), 2000)
- D) Promise.setTimeout(resolve("Hello"), 2000)

Correct Answer: A) new Promise((resolve) => setTimeout(() => resolve("Hello"), 2000))

Explanation: Option A correctly uses the Promise constructor to create a new Promise. It utilizes setTimeout to delay the execution, and after 2 seconds, it resolves the promise with the value "Hello". The resolve function is called within the setTimeout callback to ensure the promise resolves after the specified delay.

What is the result of the following code snippet?
console.log("2" + 3 + 5);

- A) 235
- B) 10
- C) "25"

- D) "10"

Correct Answer: A) 235

Explanation: JavaScript performs left-to-right evaluation for the + operator. When a string is involved, it concatenates the following values as strings rather than adding them numerically. So, "2" concatenated with 3 becomes "23", and "23" concatenated with 5 becomes "235".

How do you deeply clone an object in JavaScript, including objects with circular references?

- A) Using JSON.parse(JSON.stringify(object))
- B) Using Object.assign({}, object)
- C) Using a custom clone function or a library like Lodash's _.cloneDeep()
- D) Using the spread operator { ...object }

Correct Answer: C) Using a custom clone function or a library like Lodash's _.cloneDeep()

Explanation: Options A, B, and D cannot clone objects with circular references or complex objects (like functions, Date objects, etc.) correctly. A custom function capable of handling circular references or a utility library like Lodash's _.cloneDeep() method is necessary for deep cloning complex objects, including those with circular references.

How do you convert a NodeList to an array in ES6?

- A) Array.from(nodeList)

- B) [...nodeList]
- C) nodeList.toArray()
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: Both Array.from(nodeList) and the spread operator [...nodeList] are valid ES6 methods to convert a NodeList to an array. These methods allow array methods like map, filter, and reduce to be used on the resulting array.

What does the ?. operator in JavaScript return if the property before it is null or undefined?

- A) false
- B) null
- C) undefined
- D) Throws a TypeError

Correct Answer: C) undefined

Explanation: The optional chaining operator ?. permits reading the value of a property located deep within a chain of connected objects without having to check that each reference in the chain is valid. If any reference is null or undefined, the expression short-circuits and returns undefined, without throwing an error.

Which method should be used to schedule a code block to be executed repeatedly every X milliseconds in JavaScript?

- A) setTimeout(codeBlock, X)
- B) setInterval(codeBlock, X)
- C) requestAnimationFrame(codeBlock)
- D) repeat(codeBlock, X)

Correct Answer: B) setInterval(codeBlock, X)

Explanation: setInterval() is used to repeatedly run a function or executes a code snippet, with a fixed time delay between each call. It continues calling the function until clearInterval() is called or the window is closed, making it suitable for repeated executions at specific intervals.

How can you check if a string contains a specific substring in JavaScript?

- A) string.includes(substring)
- B) string.contains(substring)
- C) substring.in(string)
- D) string.indexOf(substring) !== -1

Correct Answer: D) string.indexOf(substring) !== -1

Explanation: While string.includes(substring) is the modern and straightforward way to check if a string contains a specific substring, string.indexOf(substring) !== -1 is a more universally supported method that works by checking if the index of the substring is not -1 (indicating the substring exists within the string). Hence, both A and D are correct, but only D was listed as an option, which reflects a traditional approach.

What will the following code snippet output?

```
let x = 10;
function changeX() {
x = 20;
}
changeX();
console.log(x);
```

- A) 10
- B) 20
- C) undefined
- D) ReferenceError

Correct Answer: B) 20

Explanation: The function changeX modifies the global variable x by setting its value to 20. Since x is declared in the global scope, calling changeX() affects the global variable directly. Therefore, console.log(x) outputs 20.

How do you remove the last element from an array and return it in JavaScript?

- A) array.pop()
- B) array.shift()
- C) array.removeLast()
- D) array.splice(-1, 1)

Correct Answer: A) array.pop()

Explanation: The pop() method removes the last element from an array and returns that element. This method changes the length of the array. shift(), in contrast, removes the first element. splice(-1, 1) also removes the last element but returns it inside an array.

In JavaScript, what will the following code snippet output?
console.log(typeof NaN);

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

Correct Answer: B) "number"

Explanation: NaN stands for "Not-a-Number", but it is still considered a numeric type in JavaScript. Thus, typeof NaN returns "number", reflecting its classification as a numeric type despite its value indicating an invalid number.

What does the Array.prototype.splice() method return?
- A) A new array containing the deleted elements.
- B) The length of the new array.
- C) The modified array.
- D) undefined.

Correct Answer: A) A new array containing the deleted elements.

Explanation: The splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place. It returns an array containing the elements that were removed from the original array.

How can you determine the number of characters in a string?

- A) string.charCount
- B) string.length
- C) string.size()
- D) string.chars()

Correct Answer: B) string.length

Explanation: The length property of a string in JavaScript returns the number of characters present in the string.

Which statement about JavaScript's switch statement is true?

- A) The switch statement can only test for equality against strings.
- B) The switch statement executes all cases that follow a matching case unless a break is encountered.
- C) The switch statement tests for strict equality (===) only.
- D) Variables declared in one case block are scoped to that block only.

Correct Answer: B) The switch statement executes all cases that follow a matching case unless a break is encountered.

Explanation: The switch statement evaluates an expression and executes all following case clauses that match the expression's value. It continues executing down the list of cases, regardless of matches, until it encounters a break statement or reaches the end of the switch block. This behavior is known as "fall-through".

What will the following code output?

```
let arr = [1, 2, 3];
let [a, , c] = arr;
console.log(a + c);
```

- A) 3
- B) 4

- C) NaN
- D) SyntaxError

Correct Answer: B) 4

Explanation: This code snippet demonstrates array destructuring where a is assigned the value 1 from the array, and c is assigned the value 3, skipping the second element. The output of console.log(a + c) is 4.

How do you create a private variable in a JavaScript function?

- A) By declaring the variable with the var keyword inside a function.
- B) By declaring the variable with the private keyword.
- C) By prefixing the variable name with a # symbol.
- D) By declaring the variable outside of any function.

Correct Answer: A) By declaring the variable with the var keyword inside a function.

Explanation: Variables declared with var, let, or const inside a function are local to the function and cannot be accessed outside of it, effectively making them private to that function. The private keyword and # symbol are used in classes to define private fields or methods, not within functions.

What is the result of "5" - 2 in JavaScript?

- A) "3"
- B) 3
- C) "52"
- D) NaN

Correct Answer: B) 3

Explanation: In JavaScript, the - operator triggers numeric conversion of its operands. The string "5" is converted to the number 5, and then 2 is subtracted, resulting in the numeric value 3.

How do you stop a setInterval() from further executing in JavaScript?

- A) setInterval.stop()
- B) clearInterval(intervalID)
- C) intervalID.stop()
- D) stopInterval(intervalID)

Correct Answer: B) clearInterval(intervalID)

Explanation: The clearInterval() function is used to stop a timer that was previously established by calling setInterval(). The intervalID is the identifier of the interval you want to clear, returned by setInterval().

What HTML event attribute can run JavaScript code when a user changes the content of an <input> element?

- A) onchange
- B) oninput
- C) onedit
- D) ontextchange

Correct Answer: A) onchange

Explanation: The onchange event attribute executes JavaScript code when the content of a form element (like <input>, <select>, or <textarea>) changes. The change typically needs to be committed (e.g., by moving focus away from the element) for onchange to trigger. For immediate reaction to changes, oninput is more suitable, making both A and B viable depending on the context.

In ES6, which keyword is used to declare a variable that cannot be reassigned?

- A) var
- B) let
- C) const
- D) static

Correct Answer: C) const

Explanation: The const keyword in ES6 is used to declare a block-scoped variable that cannot be reassigned after its initial assignment. While variables declared with const cannot be reassigned, the contents of a const object can still be modified (e.g., properties can be added or changed).

How do you check if a JavaScript object obj has a property key?

- A) obj.hasOwnProperty('key')
- B) 'key' in obj
- C) obj.contains('key')
- D) Both A and B are correct

Correct Answer: D) Both A and B are correct

Explanation: The hasOwnProperty() method checks if an object has a property as its own (not inherited), while the in operator checks if the property exists on the object or its prototype chain. Both can be used to check for the presence of a property, but they have different uses depending on whether inherited properties should be considered.