100 JAVASCRIPT INTERVIEW QUESTIONS

1. Define JavaScript:

JavaScript is a versatile, interpreted scripting language primarily utilized for enhancing web page interactivity.

2. Enumerate JavaScript Data Types:

JavaScript encompasses six primitive data types: string, number, boolean, null, undefined, and symbol, alongside objects.

3. Distinguish Null from Undefined:

While null indicates deliberate absence of an object value, undefined denotes an uninitialized variable or absence of value.

4. Explain DOM in JavaScript:

DOM (Document Object Model) is a crucial interface representing document structure in web development, facilitating dynamic manipulation.

5. Elucidate JavaScript Event:

An event signifies a browser action like button clicks or page loading, to which JavaScript responds with corresponding code execution.

6. Detail Anonymous Functions:

Anonymous functions lack names and are often assigned to variables or passed as arguments, commonly for one-time use or callbacks.

7. Describe JavaScript Closures:

Closures retain access to outer function variables even after function execution, encapsulating data and enabling persistent state maintenance.

8. Differentiate == and === in JavaScript: The == operator performs type coercion before checking equality, while === strictly compares both value and type.

9. Explain Hoisting in JavaScript:

Hoisting moves variable and function declarations to the top of their scope during compilation, enabling usage before declaration.

10. Clarify the 'this' Keyword:

'this' refers to the current executing object in JavaScript, dynamically determined and facilitating object property/method access within functions.

11. Explore Function Definition Methods:

JavaScript functions can be defined via declarations, expressions, arrow functions, and object methods, offering flexibility in code structure.

12. Discuss the Role of 'let' Keyword:

'let' declares block-scoped variables, confining their accessibility to the block they're defined within, ensuring better code organization.

13. Elaborate on 'const' Keyword Usage:
'const' establishes block-scoped variables, unchangeable once assigned, although it doesn't ensure immutability for objects/arrays.

14. Define Template Literals:
Template literals, indicated by backticks (`), facilitate string creation with variable interpolation and multi-line support.

15. Introduce JavaScript Promises:
Promises manage asynchronous operations, representing eventual completion/failure, fostering cleaner asynchronous code with .then() and .catch() chaining.

16. Explain async/await Syntax:
async/await offers synchronous-like syntax for asynchronous operations, enhancing code readability and maintainability.

17. Highlight Arrow Functions:
Arrow functions offer concise function syntax with implicit 'this' binding, improving code readability compared to traditional expressions.

18. Illustrate Event Delegation:
Event delegation involves attaching event listeners to parent elements, efficiently managing events for dynamically added child elements.

19. Discuss the Purpose of 'map()' Function: 'map()' creates a new array by applying a function to each element of an existing array, facilitating easy element transformation.

20. Explain the Role of 'filter()' Function:
'filter()' generates a new array containing elements meeting specified conditions, aiding efficient array element filtering based on criteria.

21. Detail the Purpose of 'reduce()' Function: 'reduce()' condenses an array to a single value by applying a function to each element and accumulating results, often for calculations or transformations.

22. Define Callback Functions:
Callback functions, passed as arguments, execute later or in response to events, enabling asynchronous and event driven programming.

23. Differentiate 'let' and 'var' in JavaScript: 'let' declares block-scoped variables, while 'var' declares function-scoped variables, with 'var' being hoisted and 'let' not.

24. Explain JavaScript Modules:
Modules in JavaScript encapsulate related functionality for better organization, encapsulation, and code reuse in large applications.

25. Elucidate Object Destructuring:

Object destructuring allows extracting object properties into variables, facilitating concise value extraction and property manipulation.

26. Introduce JavaScript Classes:

Classes in JavaScript define objects with shared properties and behaviors, serving as templates for creating multiple instances.

27. Discuss Inheritance in JavaScript:

Inheritance enables objects to inherit properties/methods from others, fostering code reuse and hierarchical relationships between objects.

28. Define JavaScript Getters and Setters:

Getters and setters manage object property access, providing control over value retrieval and assignment for data validation and encapsulation.

29. Explain try/catch Statement Purpose:

try/catch handles errors in JavaScript, allowing detection and handling of exceptions during code execution.

30. Compare 'let' and 'const' in JavaScript:

'let' declares reassignable variables, while 'const' declares read-only variables, offering immutable bindings for values.

31. Discuss the Purpose of the forEach() Function:

forEach() executes a provided function for each array element, simplifying iteration and operation execution.

32. Elaborate on the localStorage Object:

localStorage stores key-value pairs locally in the browser, facilitating persistent data storage for web applications.

33. Differentiate JavaScript arrow functions from regular functions:

Arrow functions offer a concise syntax and lexically bind 'this', unlike regular functions, promoting cleaner code and avoiding 'this' context issues.

34. Explain the Role of the setTimeout() Function:
setTimeout() schedules function execution after a specified delay, enabling time-based code execution and timeouts.

35. Define Event Bubbling in JavaScript:

Event bubbling propagates an event from the target element up through its ancestors in the DOM hierarchy, allowing event handling at multiple levels.

36. Explore the fetch() Function Purpose:

fetch() initiates HTTP requests and retrieves resources, providing a modern approach to asynchronous network operations.

37. Differentiate between null and undefined in JavaScript:

null signifies deliberate absence of value, while undefined indicates a variable lacking a defined value, often used as a default.

38. Discuss Event Propagation in JavaScript: Event propagation includes event capturing and bubbling, enabling event handling at different levels of the DOM tree.
39. Explain the Object.keys() Function Usage: Object.keys() extracts object keys, returning them as an array for easy iteration and property manipulation.

40. Highlight the addEventListener() Method Purpose: addEventListener() attaches event handlers to elements, facilitating response to specific events with corresponding function execution.

41. Detail the Purpose of the parentNode Property: parentNode accesses an element's immediate parent in the DOM, aiding traversal and manipulation of the DOM tree.

42. Elaborate on the querySelector() Method: queryselector() selects the first element matching a CSS selector, streamlining DOM element retrieval.

43. Discuss the querySelectorAll() Method Usage: querySelectorAll() selects all elements matching a CSS selector, returning a collection for iteration or access.

44. Compare querySelector() and getElementById(): querySelector() selects elements based on CSS selectors, while getElementById() specifically targets elements by unique

IDs.

45. Explore Function Declarations vs. Function Expressions:

Function declarations are hoisted and callable before definition, while function expressions aren't hoisted and must be defined before use.

46. Clarify the bind() Method Usage:

bind() creates a new function with a specified 'this' value, facilitating explicit context binding within functions.

47. Discuss the Purpose of the call() Method: call() invokes a function with a provided 'this' value and individual arguments, enabling method borrowing and explicit function invocation.

48. Elaborate on the apply() Method Purpose: apply() invokes a function with a specified 'this' value and arguments as an array, allowing method borrowing and function invocation.

49. Explain the Role of the Array.isArray() Method: Array.isArray() determines if a value is an array, returning true if it is, and false otherwise.

50. Discuss Event Capturing in JavaScript: Event capturing triggers events on parent elements before reaching the target element, enabling event handling from outer to inner elements.

51. Explore Event Delegation in JavaScript: Event delegation attaches event listeners to parent elements, efficiently managing events for dynamically added child elements.

52. Detail the Purpose of the startsWith() Method:
startsWith() verifies if a string begins with a specified substring, returning true if it does, and false otherwise.

53. Explain the endsWith() Method Usage: endsWith() checks if a string ends with a specified substring, returning true if it does, and false otherwise.

54. Discuss the includes() Method Purpose: includes() determines if a string contains a specified substring, returning true if found, and false otherwise.

55. Elaborate on the padStart() Method:
padStart() pads a string's beginning with a specified character until reaching a desired length, often used for formatting.

56. Detail the padEnd() Method Usage:
padEnd() pads a string's end with a specified character until reaching a desired length, aiding formatting.

57. Explain the charAt() Method Purpose:
charAt() retrieves a character at a specified index in a string, returning the character or an empty string if out of

range.

58. Discuss the charCodeAt() Method:

charCodeAt() retrieves the Unicode value of a character at a specified index, returning the Unicode value or NaN if out of range.

59. Elaborate on the String.fromCharCode() Method:

String.fromCharCode() creates a string from Unicode values, converting them to corresponding characters.

60. Discuss the JSON.stringify() Method:

JSON.stringify() converts JavaScript objects/values to JSON strings, commonly used for data serialization.

61. Elaborate on the JSON.parse() Method:

JSON.parse() parses JSON strings into JavaScript objects/values, facilitating data deserialization.

62. Explain the encodeURIComponent() Function:

encodeURIComponent() encodes special characters in URL components, ensuring valid URL inclusion.

63. Detail the decodeURIComponent() Function:

decodeURIComponent() decodes URL-encoded components, restoring original characters.

64. Discuss the Math.random() Function Purpose:

Math.random() generates random floating-point numbers between 0 (inclusive) and 1 (exclusive), introducing randomness in JavaScript.

65. Explore the Math.floor() Function:
Math.floor() rounds a number down to the nearest integer, removing the decimal part.

66. Clarify the Math.ceil() Function Usage:
Math.ceil() rounds a number up to the nearest integer, disregarding the decimal part.
67. Elaborate on the Math.round() Function:
Math.round() rounds a number to the nearest integer, adjusting based on the decimal part.

68. Discuss the Purpose of the Math.max() Function:
Math.max() identifies the highest number among arguments, returning the maximum value.

69. Elaborate on the Math.min() Function: Math.min() finds the lowest number among arguments, returning the minimum value.

70. Explore the Math.pow() Function Usage: Math.pow() calculates the power of a number, raising it to a specified exponent.

71. Discuss the Math.sqrt() Function Purpose:
Math.sqrt() computes the square root of a number, returning the positive square root.

72. Elaborate on the Math.abs() Function:

Math.abs() determines the absolute value of a number, disregarding its sign.

73. Discuss the Purpose of Math.floor() with Math.random():

Combining Math.floor() with Math.random() generates random integers within a specified range.

74. Explain the Date() Constructor Usage:

The Date() constructor creates Date objects representing specific dates and times for manipulation.

75. Explore the getFullYear() Method:

getFullYear() retrieves the four-digit year value from a Date object, facilitating date information extraction.

76. Elaborate on the getMonth() Method Purpose:

getMonth() retrieves the month index from a Date object, providing month information.

77. Discuss the getDate() Method Usage:

getDate() retrieves the day of the month from a Date object, offering day information.

78. Elaborate on the getDay() Method Purpose:

getDay() retrieves the day of the week index

from a Date object, enabling day-of-week information extraction.

79. Explore the getHours() Method Usage: getHours() retrieves the hour value from a Date object, offering time information.

80. Discuss the getMinutes() Method Purpose: getMinutes() retrieves the minute value from a Date object, aiding time extraction.

81. Elaborate on the getSeconds() Method Usage:
getSeconds() retrieves the second value from a Date object, facilitating time information extraction.

82. Explain the getMilliseconds() Method Purpose: getMilliseconds() retrieves the millisecond value from a Date object, providing precise time information.

83. Discuss the Date.now() Method Usage: Date.now() returns the current timestamp in milliseconds since the Unix epoch, facilitating time-related calculations.

84. Elaborate on the setTime() Method Purpose: setTime() sets the time of a Date object based on a specified number of milliseconds since the Unix epoch.

85. Explore the setFullYear() Method Usage: setFullYear() sets the year of a Date object, allowing date manipulation.

86. Discuss the setMonth() Method Purpose: setMonth()

sets the month of a Date object, enabling date modification.

87. Elaborate on the setDate() Method Usage: setDate() sets the day of the month for a Date object, aiding date manipulation.

88. Explain the setHours() Method Purpose: setHours() sets the hour for a Date object, facilitating time adjustment.

89. Discuss the setMinutes() Method Usage:
setMinutes() sets the minutes for a Date object, enabling precise time modification.

90. Elaborate on the setSeconds() Method Purpose: setSeconds() sets the seconds for a Date object, aiding time adjustment.

91. Explore the setMilliseconds() Method Usage: setMilliseconds() sets the milliseconds for a Date object, allowing precise time modification.

92. Discuss the Array.from() Method Purpose: Array.from() creates a new array from an array-like or iterable object, facilitating array transformation.

93. Elaborate on the Array.isArray() Method Usage: Array.isArray() verifies if a value is an array, returning true if it is, and false otherwise.

94. Explain the Array.of() Method Purpose: Array.of()

creates a new array with provided elements, regardless of type or number.

95. Discuss the concat() Method Usage:

concat() merges two or more arrays, creating a new array with combined elements.

96. Elaborate on the copyWithin() Method Purpose: copyWithin() copies array elements to specified positions within the same array, facilitating in-place modification.

97. Explore the entries() Method Usage:

entries() creates an iterator object containing key/value pairs for each array index, aiding array iteration.

98. Discuss the fill() Method Purpose:

fill() populates array elements with a specified value, facilitating array initialization.

99. Elaborate on the filter() Method Usage: filter() creates a new array with elements passing a specified condition, enabling efficient array filtering.

100. Explain the find() Method Purpose:

find() retrieves the first array element meeting a specified condition, returning the element or undefined if not found.