1. **LIST OF KEYWORDS IN JAVASCRIPT**
2. **break**: Terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.
3. **case**: Marks a block of statements to be executed in a switch statement.
4. **catch**: Catches an exception generated by a try statement.
5. **class**: Declares a class.
6. **const**: Declares a read-only reference to a value.
7. **continue**: Jumps to the next iteration of a loop.
8. **debugger**: Stops the execution of JavaScript and calls the debugging function.
9. **default**: Specifies the block of code to be executed if a case clause expression does not match any case clause expression in a switch statement.
10. **delete**: Deletes an object, property, or element.
11. **do**: Starts a loop that executes a block of statements and tests the condition after each iteration.
12. **else**: Specifies a block of code to be executed if a condition is false.
13. **export**: Exports functions, objects, or primitive values from a module.
14. **extends**: Extends a class.
15. **finally**: Specifies a block of code to be executed after a try or catch block.
16. **for**: Starts a loop that executes a block of statements as long as a condition evaluates to true.
17. **function**: Declares a function.
18. **if**: Specifies a block of code to be executed if a condition is true.
19. **import**: Imports functions, objects, or primitive values from a module.
20. **in**: Checks if a specified property is in an object.
21. **instanceof**: Checks if an object is an instance of a specified object type.
22. **let**: Declares a block-scoped local variable.
23. **new**: Creates an instance of a user-defined object type.
24. **return**: Exits a function and specifies the value to be returned.
25. **super**: Calls the parent constructor.
26. **switch**: Evaluates an expression, matching the expression's value to a case clause, and executes the statements associated with that case.
27. **this**: Refers to the current object.
28. **throw**: Throws an exception.
29. **try**: Marks a block of statements to try, and specifies a response, should an exception be thrown.
30. **typeof**: Returns a string indicating the type of the operand.
31. **var**: Declares a variable, optionally initializing it to a value.
32. **void**: Evaluates an expression and returns undefined.
33. **while**: Creates a loop that executes a specified statement as long as the test condition evaluates to true.
34. **with**: Extends the scope chain for a statement.
35. **yield**: Pauses and resumes a generator function.
36. **await**: Pauses the execution of an async function and waits for the resolution of a promise.
37. **var**: Used for declaring variables. However, with the introduction of **let** and **const**, **var** is less commonly used now.
38. **let**: Declares a block-scoped variable, limiting its scope to the block, statement, or expression on which it is used.
39. **const**: Declares a block-scoped constant, meaning its value cannot be re-assigned.
40. **if**: Used to execute a block of code if a specified condition is true.
41. **else**: Used to execute a block of code if the same condition specified in the **if** statement is false.
42. **else if**: Used to specify a new condition if the first condition in the **if** statement is false.
43. **for**: Creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons.
44. **while**: Creates a loop that executes a specified statement as long as the test condition evaluates to true.
45. **do...while**: Similar to **while**, but the condition is evaluated after executing the statement, resulting in the statement executing at least once.
46. **switch**: Evaluates an expression, matching the expression's value to a **case** clause, and executes statements associated with that **case**.
47. **case**: Defines a value in **switch** statement to match against the expression.
48. **break**: Terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.
49. **continue**: Skips the rest of the current iteration of a loop and continues with the next iteration.
50. **function**: Declares a function.
51. **return**: Specifies the value to be returned by a function.
52. **try**: Marks a block of statements to try, and specifies a response should an exception be thrown.
53. **catch**: Specifies a block of code to execute in case an error occurs in the try block.
54. **throw**: Throws a user-defined exception.
55. **finally**: Specifies a block of code to execute after try/catch, regardless of the result.
56. **class**: Declares a class in JavaScript.
57. **extends**: Allows a class to inherit from another class.
58. **super**: Calls the constructor of the parent class.
59. **this**: Refers to the current object.
60. **new**: Creates an instance of a user-defined object type or built-in object type.
61. **delete**: Deletes an object, object's property, or element at a specified index in an array.
62. **typeof**: Returns a string indicating the type of the unevaluated operand.
63. **instanceof**: Checks whether an object is an instance of a specified object type.
64. **void**: Specifies an expression to be evaluated without returning a value.
65. **in**: Checks if a specified property is in an object.
66. **with**: Extends the scope chain for a statement.
67. **debugger**: Stops the execution of JavaScript, allowing debugging of code.
68. **default**: Specifies the default case in a **switch** statement.
69. **export**: Used to export functions, objects, or primitive values from a module.
70. **import**: Used to import functions, objects, or primitive values into a module.
71. **await**: Pauses the execution of an **async** function until the promise is resolved.
72. **async**: Declares an asynchronous function, which enables the use of the **await** keyword.
73. **get**: Defines a method to get a specific property of an object.
74. **set**: Defines a method to set a specific property of an object.
75. **yield**: Pauses and resumes the execution of a generator function.
76. **let**: Declares a block-scoped variable, limiting its scope to the block, statement, or expression on which it is used.
77. **const**: Declares a block-scoped constant, meaning its value cannot be re-assigned.
78. **null**: Represents an intentional absence of any object value.
79. **undefined**: Represents a variable that has not been assigned a value.
80. **true**: Represents the boolean value true.
81. **false**: Represents the boolean value false.
82. **NaN**: Represents "Not-a-Number," returned when a mathematical operation is performed but the result is not a valid number.
83. **Infinity**: Represents positive infinity.
84. **typeof**: Returns a string indicating the type of the unevaluated operand.
85. **instanceof**: Checks whether an object is an instance of a specified object type.
86. **function**: Declares a function.
87. **return**: Specifies the value to be returned by a function.
88. **try**: Marks a block of statements to try, and specifies a response should an exception be thrown.
89. **catch**: Specifies a block of code to execute in case an error occurs in the try block.
90. **finally**: Specifies a block of code to execute after try/catch, regardless of the result.
91. **class**: Declares a class in JavaScript.
92. **extends**: Allows a class to inherit from another class.
93. **super**: Calls the constructor of the parent class.
94. **this**: Refers to the current object.
95. **new**: Creates an instance of a user-defined object type or built-in object type.
96. **delete**: Deletes an object, object's property, or element at a specified index in an array.
97. **with**: Extends the scope chain for a statement.
98. **break**: Terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.
99. **continue**: Skips the rest of the current iteration of a loop and continues with the next iteration.
100. **switch**: Evaluates an expression, matching the expression's value to a **case** clause, and executes statements associated with that **case**.
101. **case**: Defines a value in a **switch** statement to match against the expression.
102. **if**: Used to execute a block of code if a specified condition is true.
103. **else**: Used to execute a block of code if the same condition specified in the **if** statement is false.
104. **else if**: Used to specify a new condition if the first condition in the **if** statement is false.
105. **for**: Creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons.
106. **while**: Creates a loop that executes a specified statement as long as the test condition evaluates to true.
107. **do...while**: Similar to **while**, but the condition is evaluated after executing the statement, resulting in the statement executing at least once.
108. **function**: Declares a function.
109. **return**: Specifies the value to be returned by a function.
110. **try**: Marks a block of statements to try, and specifies a response should an exception be thrown.
111. **catch**: Specifies a block of code to execute in case an error occurs in the try block.
112. **throw**: Throws a user-defined exception.
113. **finally**: Specifies a block of code to execute after try/catch, regardless of the result.
114. **class**: Declares a class in JavaScript.
115. **extends**: Allows a class to inherit from another class.
116. **super**: Calls the constructor of the parent class.
117. **this**: Refers to the current object.
118. **new**: Creates an instance of a user-defined object type or built-in object type.
119. **delete**: Deletes an object, object's property, or element at a specified index in an array.
120. **typeof**: Returns a string indicating the type of the unevaluated operand.
121. **instanceof**: Checks whether an object is an instance of a specified object type.
122. **void**: Specifies an expression to be evaluated without returning a value.
123. **in**: Checks if a specified property is in an object.
124. **break**: Terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.
125. **continue**: Skips the rest of the current iteration of a loop and continues with the next iteration.
126. **switch**: Evaluates an expression, matching the expression's value to a **case** clause, and executes statements associated with that **case**.
127. **case**: Defines a value in **switch** statement to match against the expression.
128. **default**: Specifies the default case in a **switch** statement.
129. **throw**: Throws a user-defined exception.
130. **with**: Extends the scope chain for a statement.
131. **debugger**: Stops the execution of JavaScript, allowing debugging of code.
132. **typeof**: Returns a string indicating the type of the unevaluated operand.
133. **delete**: Deletes an object, object's property, or element at a specified index in an array.
134. **void**: Specifies an expression to be evaluated without returning a value.
135. **yield**: Pauses and resumes the execution of a generator function.
136. **async**: Declares an asynchronous function, which enables the use of the **await** keyword.
137. **await**: Pauses the execution of an **async** function until the promise is resolved.
138. **get**: Defines a method to get a specific property of an object.
139. **set**: Defines a method to set a specific property of an object.
140. **eval**: Evaluates JavaScript code represented as a string.
141. **isNaN**: Determines whether a value is NaN (Not-a-Number).
142. **parseFloat**: Parses a string argument and returns a floating point number.
143. **parseInt**: Parses a string argument and returns an integer.
144. **isFinite**: Determines whether a value is a finite number.
145. **decodeURI**: Decodes a Uniform Resource Identifier (URI) previously created by **encodeURI**.
146. **decodeURIComponent**: Decodes a Uniform Resource Identifier (URI) component previously created by **encodeURIComponent**.
147. **encodeURI**: Encodes a Uniform Resource Identifier (URI) by replacing each instance of certain characters with one, two, three, or four