

```
1 console.clear();
2
3 /*
4
5 =====
6
7           Operators Interview Questions 🚀
8
9 =====
10
11 */
12
13
14 /*
15
16 1. What is the difference between == and === in JavaScript?
17
18 🖱️ == checks whether the values of two operands are equal or not. === checks whether the values and data type of two operands
19 are equal or not. For example: 🖱️
20
21 */
22 let num1 = 10;
23
24 let num2 = "10";
25
26 console.log(num1 == num2);
27
28 console.log(num1 === num2);
29
30 /*
31
32 2. Write a program that determines if a person is eligible to drive based on their age being greater than or equal to 18 and
33 having a valid driving license using ternary and logical operators.
34
35 🖱️ If the person is eligible, print "You are eligible to drive". If the person is not eligible, print "You are not eligible
36 to drive". 🖱️
```

```
36 */
37
38 let age = 18;
39
40 let haveDrivingLicense = true;
41
42 console.log(age >= 18 && haveDrivingLicense ? "You are eligible to drive." : "You are not eligible to drive.");
43
44 /*
```

46 3. What is the difference between null, undefined and Not defined in JavaScript?

47 📌 null is an assigned value representing the intentional absence of any object value.

49 📌 undefined means a variable has been declared but has not been assigned a value.

51 📌 Not defined means a variable has not been declared yet but we are trying to access it.

```
53
54 */
```

```
55
56 let a;
```

```
57
58 console.log(a); // undefined
```

```
59
60 let b = null;
```

```
61
62 console.log(b); // null
```

```
63
64 // console.log(c); ReferenceError: c is not defined
```

```
65
66 /*
```

67
68 4. What is the difference between && and || operators in JavaScript?

69
70 📌 && is the logical AND operator. It returns true if both the operands are true.

71
72 📌 || is the logical OR operator. It returns true if at least one of the operands is true.

```
73
```



```
74 */
75
76 console.log(true && false); // false
77
78 console.log(false || true); // true
79
80 console.log(0 || "Hello"); // "Hello"
81
82 console.log(5 && 10); // 10
83
84 console.log(0 && "Hello"); // 0
85
86 /*
87
88 5. What is the difference between && and ?? (Nullish Coalescing Operator) in JavaScript?
89
90 📌 && is the logical AND operator. It returns true if both the operands are true.
91
92 📌 ?? is the nullish coalescing operator. It returns the first non-nullish operand. ?? (Nullish Coalescing) only checks for
93 null or undefined and returns the right operand if the left is null or undefined.
94
95 */
96 console.log(false && "Hello"); // false
97
98 console.log(null ?? "Default"); // "Default"
99
100 console.log(0 ?? "Fallback"); // 0
101
102 console.log("" ?? "Empty"); // ""
103
104 /*
105
106 =====
107
108 Output Based Interview Questions 🚀
109
110 =====
```



```
111
112 */
113
114 console.log("5" - 3); // 2 (String is converted to number)
115
116 console.log(2 < 12 < 5); // true (2 is less than 12 and 12 is less than 5)
117
118 console.log("20" + 10 + 10); // 201010 (String concatenation)
119
120 console.log("20" - 10 - 10); // 0 (String is converted to number)
121
122 /*
123
124 =====
125
126           Questions on Bitwise Operators 🚀
127
128 =====
129
130 */
131
132 /*
133
134 📌 To solve this questions take the reference of binary representation table.
135
136 📌 We will draw the whole table here:
137
138 */
139
140 /*
141
142 Decimal    Binary
143
144 0           0000
145 1           0001
146 2           0010
147 3           0011
148 4           0100
```



```
149 5      0101
150 6      0110
151 7      0111
152 8      1000
153 9      1001
154 10     1010
155 11     1011
156 12     1100
157 13     1101
158 14     1110
159 15     1111
160
161 */
162
163 // By applying the bitwise operators, we get the following results:
164
165 // In this example, the binary representation of 5 is 0101 and 3 is 0011
166
167 console.log(5 & 3);
168
169 // If the element on both sides is 1, the result will be 1 otherwise 0.
170
171 // Result: 0001
172
173 console.log(5 | 3);
174
175 // If the element on either side is 1, the result will be 1 otherwise 0.
176
177 // Result: 0111
178
179 console.log(5 ^ 3);
180
181 // If the element on both sides is 0, the result will be 0 otherwise 1.
182
183 // Result: 0110
184
185 console.log(~5);
186
```



```
187 // The result will be the one's complement of 5.
188
189 // Result: -6
190
191 /*
192
193 =====
194
195     Questions based on Ternary (Conditional) Operators 🚀
196
197 =====
198
199 */
200
201 let c = 0;
202
203 let d = 10;
204
205 console.log(c || d && "Hello");
206
207 // Result: Hello because 0 is false and 10 is true
208
209 console.log(c && d || "World");
210
211 // Result: World because 0 is false and 10 is true
212
213 console.log(c ?? d ?? "Fallback");
214
215 // Result: Fallback because both c and d are false
216
217 /*
218
219 =====
220
221     Questions based on Type Coercion 🚀
222
223 =====
224
```



```
225 */
226
227 console.log([] + {});
228
229 // Result: [object Object] because [] is an array and {} is an object and both are implicitly converted to strings
230
231 console.log({} + []);
232
233 // Result: [object Object] because {} is an object and [] is an array and both are implicitly converted to strings
234
235 console.log(true + +"10");
236
237 // Result: 11 because true is implicitly converted to 1 and "10" is implicitly converted to Number 10
238
239 console.log(!"false" == !!"true");
240
241 // Result: true because both are implicitly converted to true ("false" coerced to false and "true" coerced to true)
242
243 console.log([] == ![]);
244
245 // Result: true because both are implicitly converted to false (empty array coerced to empty string)
246
247 /*
248
249 =====
250
251     Questions based on typeof Operator 🚀
252
253 =====
254
255 */
256
257 console.log(typeof NaN); // number
258
259 console.log(typeof null); // object
260
261 console.log(typeof undefined); // undefined
262
```



```
263 console.log(typeof []); // object  
264  
265 console.log(typeof function() {}); // function
```

