

```
1 console.clear();
2
3 /*
4 =====
5
6           Data Types Interview Questions 🧑‍💻
7
8 =====
9 */
10
11 console.log("Data Types Interview Questions");
12
13 /*
14
15 1. What is the difference between null and undefined in JavaScript?
16
17 📌 `null` is an assigned value that represents the intentional absence of any object value.
18
19 📌 `undefined` means a variable is declared but not assigned any value.
20
21 */
22
23 // Example:
24
25 let a = null;
26
27 let b;
28
29 console.log("a:", a, "Type:", typeof a); // object (JS bug)
30
31 console.log("b:", b, "Type:", typeof b); // undefined
32
33 /*
34
35 2. What is the purpose of the `typeof` operator in JavaScript?
36
37 📌 The `typeof` operator is used to find the data type of a variable.
```

```
38
39 */
40
41 // Example:
42
43 console.log(typeof "JavaScript"); // string
44
45 console.log(typeof 42); // number
46
47 console.log(typeof false); // boolean
48
49 /*
50
51 3. What are the primitive data types in JavaScript?
52
53 📌 JavaScript has 7 primitive data types:
54
55   ✓ Number
56
57   ✓ String
58
59   ✓ Boolean
60
61   ✓ Undefined
62
63   ✓ Null
64
65   ✓ BigInt
66
67   ✓ Symbol
68
69 */
70
71 // Example:
72
73 let myFavoriteNumber = -7; // number
74
75 let myFavoriteString = "JavaScript"; // string
```

```
76
77 let myFavoriteBoolean = true; // boolean
78
79 let myFavoriteUndefined; // undefined
80
81 let myFavoriteNull = null; // null
82
83 let myFavoriteBigInt = 1234567890123456789012345678901234567890n; // bigint
84
85 let myFavoriteSymbol = Symbol("id"); // symbol
86
87 /*
88
89 4. What are the non-primitive data types in JavaScript?
90
91 📌 The non-primitive data types are Objects, Arrays, Functions, Maps, Sets, WeakMap, WeakSet etc.
92
93 */
94
95 /*
96
97 5. How do you check if a variable is an array in JavaScript?
98
99 📌 Use `Array.isArray(variable)` method.
100
101 */
102
103 console.log(Array.isArray([1, 2, 3])); // true
104
105 console.log(Array.isArray({ name: "John" })); // false
106
107 /*
108
109 6. What is the difference between implicit and explicit type coercion?
110
111 📌 Implicit Coercion happens automatically by JavaScript.
112
113 📌 Explicit Coercion happens when we manually convert a value.
```

```
114
115 */
116
117 // Implicit
118
119 console.log(10 + "5"); // "105" (number converted to string)
120
121 // Explicit
122
123 console.log(Number("5") + 10); // 15 (string converted to number)
124
125 /*
126
127 7. Convert a number to a string.
128
129 📌 Use `toString()` or concatenate with `""`.
130
131 */
132
133 let num = 123;
134
135 console.log(num.toString(), typeof num.toString()); // "123" string
136
137 console.log("" + num, typeof "" + num); // "123" string
138
139 /*
140
141 8. Convert a string to a number.
142
143 📌 Use `Number()` or `+` operator.
144
145 */
146
147 console.log(Number("50"), typeof Number("50")); // 50 number
148
149 console.log(+ "50", typeof + "50"); // 50 number
150
151 /*
```

```
152
153 9. What is the difference between parseInt() and parseFloat()?
154
155 📌 `parseInt()` converts a string to an integer.
156
157 📌 `parseFloat()` converts a string to a decimal.
158
159 */
160
161 console.log(parseInt("99.99")); // 99
162
163 console.log(parseFloat("99.99")); // 99.99
164
165 /*
166
167 10. What are truthy and falsy values?
168
169 📌 Truthy values: non-empty strings, numbers (except 0), objects, arrays.
170
171 📌 Falsy values: `false`, `0`, `""`, `null`, `undefined`, `NaN`.
172
173 */
174
175 console.log(Boolean("hello")); // true
176
177 console.log(Boolean("")); // false
178
179 /*
180
181 11. How to check if a variable is `NaN`?
182
183 📌 Use `Number.isNaN(value)`
184
185 */
186
187 console.log(Number.isNaN(NaN)); // true
188
189 console.log(Number.isNaN("hello")); // false
```

```
190
191 /*
192
193 12. What is the output of `typeof NaN`?
194
195 */
196 console.log(typeof NaN); // number ! (weird JS behavior)
197
198 /*
199
200
201 13. How to check if a number is finite?
202
203 📌 Use `Number.isFinite(value)`
204
205 */
206 console.log(Number.isFinite(100)); // true
207
208 console.log(Number.isFinite(Infinity)); // false
209
210 /*
211
212
213 14. What is Symbol in JavaScript?
214
215 📌 `Symbol` is a unique primitive value used for object properties.
216
217 */
218
219 const sym1 = Symbol("id");
220
221 const sym2 = Symbol("id");
222
223 console.log(sym1 === sym2); // false (each symbol is unique)
224
225 /*
226
227 15. What is the difference between shallow copy and deep copy?
```

```
228
229 📌 Shallow copy only copies references, while deep copy clones all values.
230
231 */
232
233 const obj1 = { name: "John" };
234
235 const obj2 = obj1;
236
237 console.log(obj1 === obj2); // true (shallow copy)
238
239 const obj3 = { ...obj1 };
240
241 console.log(obj1 === obj3); // false (deep copy)
242
243 /*
244
245 =====
246
247           Output-Based JavaScript Interview Questions 🔥
248
249 =====
250 */
251
252 console.log("Output-Based Interview Questions");
253
254 // 1 String + Number
255
256 console.log("10" + 20); // "1020"
257
258 // 2 Number - String
259
260 console.log(10 - "5"); // 5 (string converted to number)
261
262 // 3 Boolean + Number
263
264 console.log(true + 1); // 2 (true is 1)
265
```

```
266 // 4 Boolean - Boolean
267
268 console.log(false - true); // -1 (false = 0, true = 1)
269
270 // 5 Null + Number
271
272 console.log(null + 10); // 10 (null treated as 0)
273
274 // 6 Undefined + Number
275
276 console.log(undefined + 10); // NaN (undefined cannot be converted)
277
278 // 7 Empty String + Number
279
280 console.log("" + 10); // "10" (string concatenation)
281
282 // 8 Empty String - Number
283
284 console.log("" - 10); // -10 (" is treated as 0)
285
286 // 9 Comparing null and 0
287
288 console.log(null == 0); // false
289
290 console.log(null >= 0); // true ! (unexpected behavior)
291
292 // 10 Comparing undefined and null
293
294 console.log(undefined == null); // true
295
296 console.log(undefined === null); // false
297
298 // 1 1 Comparing Boolean values
299
300 console.log(true == "1"); // true (string "1" is converted to number)
301
302 // 1 2 Logical OR (||) behavior
303
```



```
304 console.log(null || 5); // 5 (null is falsy)
305
306 console.log(undefined || 10); // 10 (undefined is falsy)
307
308 // 1 3 Logical AND (&&) behavior
309
310 console.log(1 && "Hello"); // "Hello" (1 is truthy, so returns second value)
311
312 console.log(0 && "World"); // 0 (0 is falsy, so returns first value)
313
314 // 1 4 Double NOT (!! ) coercion
315
316 console.log(!!"Hello"); // true
317
318 console.log(!!0); // false
319
320 // 1 5 Using `typeof` in strange cases
321
322 console.log(typeof null); // object !
323
324 console.log(typeof NaN); // number !
325
326 console.log(typeof function () { }); // function !
327
328 console.log(typeof []); // object ! (arrays are objects)
329
330 // Extra: Checking empty object
331
332 console.log(Object.keys({}).length === 0); // true (empty object)
333
334 console.log(Object.keys({ a: 1 }).length === 0); // false (non-empty object)
```