

```
1 console.clear();
2
3 /*
4
5 =====
6
7         Object Interview Questions 🚀
8
9 =====
10
11 */
12
13 /*
14
15 1. Write a function that compares two objects to determine if they have the same properties and values.
16
17 */
18
19 let obj1 = { name: "Vinod", age: 30, isStudent: false };
20
21 let obj2 = { name: "Vinod", age: 30, isStudent: false, country: "India" };
22
23 const compareObjects = (obj1, obj2) => {
24
25     if (Object.keys(obj1).length !== Object.keys(obj2).length) {
26
27         return false;
28     }
29
30     for (let key in obj1) {
31
32         if (obj1[key] !== obj2[key]) {
33
34             return false;
35         }
36     }
37 }
```

```
38     return true;
39 }
40
41 console.log(compareObjects(obj1, obj2));
42
43 /*
44 2. Given an object representing a student, write a function to add a new subject with it's corresponding grade to the
45 student's record. Also check if the grade property exists.
46 */
47
48 let studentData = {
49     name: "Bobby",
50     age: 21,
51     grades: {
52         Maths: 90,
53         Science: 85,
54         History: 88
55     }
56 }
57
58 const addSubject = (student, subject, marks) => {
59     if (!student.grades) {
60         student.grades = {};
61     }
62     return (student.grades[subject] = marks);
63 }
64
65
66
67
68
69
70
71
72
73
74
```

```
75  addSubject(studentData, "English", 95);
76
77  console.log(studentData);
78
79  let employeeData = {
80
81      name: "John",
82
83      age: 30,
84
85      personalInfo: {
86
87          address: "123 Main St",
88
89          phone: "555-555-5555"
90      }
91  }
92
93  // adding email property in the personalInfo object
94
95  const addEmail = (employee, email) => {
96
97      return employee.personalInfo.email = email;
98  }
99
100  addEmail(employeeData, "i8o9g@example.com");
101
102  console.log(employeeData);
103
104  // deleting email property from the personalInfo object
105
106  const deleteEmail = (employee) => {
107
108      delete employee.personalInfo.email;
109  }
110
111  deleteEmail(employeeData);
112
```

```
113 console.log(employeeData);
114
115 /*
116
117 3. Write a function to clone an object (shallow copy).
118
119 */
120
121 const cloneObject = (obj) => {
122
123     return { ...obj };
124 }
125
126 let originalObject = {
127
128     name: "Alice",
129
130     age: 25,
131
132     adress: {
133
134         city: "New York",
135
136         state: "NY"
137     }
138 }
139
140
141 // If your object has nested objects, then spreading ({ ...obj }) only makes a shallow copy. That means nested references are
142 // shared between original and clone.
143
144 let clone = cloneObject(originalObject);
145
146 console.log(clone);
147
148 clone.adress.city = "San Francisco";
149
150 console.log(clone);
```

O

```
150
151 console.log(originalObject);
152
153 /*
154 4. Merge two objects. If both have the same key, the second object should overwrite.
155
156 */
157
158 const mergeObjects = (obj1, obj2) => {
159     return { ...obj1, ...obj2 };
160 }
161
162 const firstObj = { x: 1, y: 2 };
163
164 const secondObj = { y: 3, z: 4 };
165
166 console.log(mergeObjects(firstObj, secondObj));
167
168 /*
169 5. Count the number of properties in an object.
170
171 */
172
173 const countProperties = (obj) => {
174     return Object.keys(obj).length;
175 }
176
177 console.log("The number of properties (keys) in the object is ", countProperties({ x: 1, y: 2, z: 3 }));
178
179 /*
180 6. Check if a property exists in an object.
181
182 */
183
184
185
186
187
```

```
188
189 const hasProperty = (obj, key) => {
190
191   return obj.hasOwnProperty(key)
192 }
193
194 let user = {
195
196   id: 1,
197
198   username: "John Doe"
199 }
200
201 console.log(hasProperty(user, "id"));
202
203 console.log(hasProperty(user, "email"));
204
205 /*
206
207 7. Convert an object to an array of key-value pairs.
208
209 */
210
211 const ObjectToPairs = (obj) => {
212
213   return Object.entries(obj);
214 }
215
216 console.log(ObjectToPairs({ a: 1, b: 2, c: 3 }));
217
218 /*
219
220 8. Create a function that removes a specific key from an object.
221
222 */
223
224 const removeKey = (obj, key) => {
225
```

```
226     delete obj[key];
227
228     return obj;
229 }
230
231 let item = { id: 1, name: "iPhone", price: 100000 };
232
233 console.log(removeKey(item, "id"));
234
235 /*
236
237 9. Iterate over all keys and values in an object.
238
239 */
240
241 const printObject = (obj) => {
242
243     for (let [key, value] of Object.entries(obj)) {
244
245         console.log(`${key}: ${value}`);
246     }
247 }
248
249 let userProfile = { name: "Sara", Profession: "Software Engineer" };
250
251 printObject(userProfile);
252
253 /*
254
255 10. Get only keys or values from an object.
256
257 */
258
259 let sampleObj = { a: 10, b: 20, c: 30 };
260
261 console.log(Object.keys(sampleObj));
262
263 console.log(Object.values(sampleObj));
```

```
264
265  /*
266
267  11. Convert an object to a string.
268
269  */
270
271  const objectToString = (obj) => {
272
273      return JSON.stringify(obj);
274  }
275
276  console.log(objectToString({ name: "John", age: 25, city: "New York" }));
277
278  /*
279
280  12. Convert a string to an object.
281
282  */
283
284  const stringToObject = (str) => {
285
286      return JSON.parse(str);
287  }
288
289  console.log(stringToObject('{"name": "John", "age": 25, "city": "New York"}'));
290
291  /*
292
293  13. Check if an object is empty.
294
295  */
296
297  const isEmptyObject = (obj) => {
298
299      return Object.keys(obj).length === 0;
300  }
301
```



```
302 console.log(isEmptyObject({}));
303
304 console.log(isEmptyObject({ name: "John", age: 25 }));
305
306 /*
307
308 14. Get the first key in an object.
309
310 */
311
312 const getFirstKey = (obj) => {
313
314     return Object.keys(obj)[0];
315 }
316
317 console.log(getFirstKey({ a: 1, b: 2, c: 3 }));
318
319 /*
320
321 15. Get the last key in an object.
322
323 */
324
325 const getLastKey = (obj) => {
326
327     return Object.keys(obj)[Object.keys(obj).length - 1];
328 }
329
330 console.log(getLastKey({ a: 11, b: 22, c: 33 }));
331
332 /*
333
334 16. Get the first value in an object.
335
336 */
337
338 const getFirstValue = (obj) => {
339
```

```
340     return Object.values(obj)[0];
341 }
342
343 console.log(getFirstValue({ a: 100, b: 200, c: 300 }));
344
345 /*
346 17. Get the last value in an object.
347 */
348
349
350
351 const getLastValue = (obj) => {
352
353     return Object.values(obj)[Object.values(obj).length - 1];
354 }
355
356 console.log(getLastValue({ a: 1000, b: 2000, c: 3000 }));
357
358 /*
359 18. Get the first key and value in an object.
360 */
361
362
363
364 const getFirstKeyValuePair = (obj) => {
365
366     return Object.entries(obj)[0];
367 }
368
369 console.log(getFirstKeyValuePair({ a: 1, b: 2, c: 3 }));
370
371 /*
372 19. Get the last key and value in an object.
373 */
374
375
376
377 const getLastKeyValuePair = (obj) => {
```

```
378
379     return Object.entries(obj)[Object.entries(obj).length - 1];
380 }
381
382 console.log(getLastKeyValuePair({ a: 10000, b: 20000, c: 30000 }));
383
384 /*
385
386 20. Get the sum of all values in an object.
387
388 */
389
390 const getSumOfValues = (obj) => {
391
392     return Object.values(obj).reduce((acc, currentValue) => acc + currentValue, 0);
393 }
394
395 console.log(getSumOfValues({ a: 10, b: 20, c: 30 }));
396
397 /*
398
399 21. Get the average of all values in an object.
400
401 */
402
403 const getAverageOfValues = (obj) => {
404
405     return Object.values(obj).reduce((acc, currentValue) => acc + currentValue, 0) / Object.values(obj).length;
406 }
407
408 console.log(getAverageOfValues({ a: 10, b: 20, c: 30 }));
409
410 /*
411
412 22. Get the maximum value in an object.
413
414 */
415
```

O

```
416 const getMaxValue = (obj) => {
417
418     return Math.max(...Object.values(obj));
419 }
420
421 console.log(getMaxValue({ a: 10, b: 20, c: 30 }));
422
423 /*
424
425 23. Get the minimum value in an object.
426
427 */
428
429 const getMinValue = (obj) => {
430
431     return Math.min(...Object.values(obj));
432 }
433
434 console.log(getMinValue({ a: 10, b: 20, c: 30 }));
435
436 /*
437
438 24. Get the length of the longest key in an object.
439
440 */
441
442 const getLongestKey = (obj) => {
443
444     return Math.max(...Object.keys(obj).map((key) => key.length));
445 }
446
447 console.log(getLongestKey({ a: 10, b: 20, c: 30 }));
448
449 /*
450
451 25. Get the length of the shortest key in an object.
452
453 */
```

```
454  
455 const getShortestKey = (obj) => {  
456  
457   return Math.min(...Object.keys(obj).map((key) => key.length));  
458 }  
459  
460 console.log(getShortestKey({ a: 10, b: 20, c: 30 }));
```