JavaScript — 50 Detailed Object Coding Questions

Q1. Access a property by key.

Given: const user = { name: 'Asha', age: 25, city: 'Pune' }
Goal: Return 25 (the value of user.age)

Q2. Add a new property to an object.

Given: const car = { brand: 'Tata', model: 'Nexon' }
Goal: Add year: 2023 to get { brand: 'Tata', model: 'Nexon', year: 2023 }

Q3. Delete a property from an object.

Given: const product = { id: 10, name: 'Pen', price: 20, discount: 5 }
Goal: Remove discount to get { id: 10, name: 'Pen', price: 20 }

Q4. Update an existing property's value.

Given: const profile = { username: 'ratna', followers: 120 }
Goal: Change followers to 125

Q5. Check if a property exists using both in operator and hasOwnProperty.

Given: const config = { darkMode: true }
Goal: Return true for 'darkMode' and false for 'lang'

Q6. Get all keys of an object.

Given: const book = { title: 'JS Basics', pages: 180, author: 'RK' }
Goal: Return ['title', 'pages', 'author']

Q7. Get all values of an object.

Given: const book = { title: 'JS Basics', pages: 180, author: 'RK' }
Goal: Return ['JS Basics', 180, 'RK']

Q8. Get key–value pairs (entries).

Given: const settings = { sound: true, quality: 'HD' }
Goal: Return [['sound', true], ['quality', 'HD']]

Q9. Build an object from two arrays of equal length (keys and values).

Given: keys = ['id', 'name'], values = [101, 'Sam']
Goal: Return { id: 101, name: 'Sam' }

Q10. Merge two shallow objects (right overwrites left on conflicts).

Given: obj1 = { a: 1, b: 2 }, obj2 = { b: 9, c: 3 }
Goal: Return { a: 1, b: 9, c: 3 }

Q11. Deep merge two nested objects (merge inner objects rather than overwrite).

Given: a = { user: { name: 'A', city: 'Pune' } }, b = { user: { city: 'Mumbai', age: 30 } }
Goal: Return { user: { name: 'A', city: 'Mumbai', age: 30 } }

Q12. Create a shallow clone of an object (changes to clone should not affect original top-level fields).

Given: const src = { x: 1, y: 2 }
Goal: Return a new object { x: 1, y: 2 }

Q13. Create a deep clone of a plain-data object using JSON methods (no functions, no Dates).

Given: const src = { a: 1, b: { c: 2 } }
Goal: Clone so that src.b !== clone.b

Q14. Freeze an object and demonstrate that updates do not apply.

Given: const state = { status: 'OPEN' }
Goal: After freezing, attempts to change status are ignored (or throw in strict mode)

Q15. Seal an object and demonstrate you can modify existing values but cannot add/remove keys.

Given: const state = { step: 1 }
Goal: Change step to 2, but adding 'done' should fail

Q16. Count how many properties an object has.

Given: const user = { id: 5, name: 'Lee', active: true }
Goal: Return 3

Q17. Sum numeric property values in an object.

Given: const salaries = { a: 1200, b: 800, c: 600 }
Goal: Return 2600

Q18. Find the key with the maximum numeric value.

Given: const votes = { A: 12, B: 19, C: 7 }
Goal: Return 'B'

Q19. Rename a key in an object (create new object with 'fname' → 'firstName').

Given: const user = { id: 1, fname: 'Aditi', city: 'Pune' }
Goal: Return { id: 1, firstName: 'Aditi', city: 'Pune' }

Q20. Pick a subset of keys into a new object.

Given: const user = { id: 3, name: 'Raj', email: 'r@x.com', role: 'dev' }, keys = ['id', 'email']
Goal: Return { id: 3, email: 'r@x.com' }

Q21. Omit a set of keys from an object.

Given: const user = { id: 3, name: 'Raj', email: 'r@x.com', role: 'dev' }, omit = ['email']
Goal: Return { id: 3, name: 'Raj', role: 'dev' }

Q22. Invert keys and values (assume values are unique strings/numbers).

Given: const colors = { red: '#f00', green: '#0f0' }
Goal: Return { '#f00': 'red', '#0f0': 'green' }

Q23. Group an array of objects by a key into an object of arrays.

Given: users = [{id:1, city:'Pune'}, {id:2, city:'Mumbai'}, {id:3, city:'Pune'}]
Goal: Return { Pune: [{id:1, city:'Pune'}, {id:3, city:'Pune'}], Mumbai: [{id:2, city:'Mumbai'}] }

Q24. Find an object in an array by id and return it.

Given: items = [{id:1, name:'A'}, {id:2, name:'B'}], id = 2
Goal: Return {id:2, name:'B'}

Q25. Update an object in an array by id (immutably return new array).

Given: items = [{id:1, qty:2}, {id:2, qty:5}], id = 1, newQty = 3
Goal: Return [{id:1, qty:3}, {id:2, qty:5}]

Q26. Remove an object from an array by id (immutably).

Given: items = [{id:1}, {id:2}, {id:3}], id = 2
Goal: Return [{id:1}, {id:3}]

Q27. Convert an object to a query string 'a=1&b;=2' (encode keys/values).

Given: const params = { a: 1, b: 'hi' }
Goal: Return 'a=1&b;=hi'

Q28. Parse a query string into an object (simple pairs, no arrays).

Given: str = 'a=1&b;=hi'
Goal: Return { a: '1', b: 'hi' }

Q29. Safely read a nested property with a path string (e.g., 'user.address.city').

Given: obj = { user: { address: { city: 'Pune' } } }, path = 'user.address.city'
Goal: Return 'Pune'

Q30. Set a nested property by path, creating objects as needed.

Given: obj = {}, path = 'a.b.c', value = 5
Goal: Resulting obj is { a: { b: { c: 5 } } }

Q31. Flatten a nested object to dot-notation keys.

Given: obj = { a: { b: { c: 1 } }, d: 2 }
Goal: Return { 'a.b.c': 1, d: 2 }

Q32. Unflatten a dot-notation object back to nested structure.

Given: flat = { 'a.b': 2, 'a.c': 3 }
Goal: Return { a: { b: 2, c: 3 } }

Q33. Check shallow equality of two objects (same keys and values, order irrelevant).

Given: a = { x: 1, y: 2 }, b = { y: 2, x: 1 }
Goal: Return true

Q34. Check deep equality of two plain objects (including nested objects/arrays).

Given: a = { x: { y: 2 } }, b = { x: { y: 2 } }

Goal: Return true

Q35. Convert an array of [key, value] pairs into an object (like Object.fromEntries).

Given: pairs = [['id', 7], ['name', 'Neo']]
Goal: Return { id: 7, name: 'Neo' }

Q36. Convert an object to a Map and back to an object.

Given: obj = { a: 1, b: 2 }
Goal: Create Map([['a',1],['b',2]]) then back to { a: 1, b: 2 }

Q37. Remove properties with falsy values from an object.

Given: obj = { a: 0, b: 2, c: '', d: 'ok' }
Goal: Return { b: 2, d: 'ok' }

Q38. Filter an object to keep only numeric values.

Given: obj = { a: 1, b: '2', c: 3 }
Goal: Return { a: 1, c: 3 }

Q39. Return a new object whose keys are sorted alphabetically (values preserved).

Given: obj = { b: 2, a: 1, c: 3 }
Goal: Return { a: 1, b: 2, c: 3 }

Q40. Stringify an object to JSON and parse it back (round-trip).

Given: obj = { x: 1, y: [2,3] }
Goal: Return same structure after parse

Q41. Define getters and setters on an object using property descriptors.

Given: const account = { _balance: 1000 }
Goal: Define get balance() and set balance(v)

Q42. Create an object with a prototype and a method; then create a child object using Object.create().

Given: proto = { greet(){ return 'hi'; } }
Goal: child.greet() returns 'hi'

Q43. Check prototype chain relation using isPrototypeOf.

Given: parent = {}, child = Object.create(parent)
Goal: Return true for parent.isPrototypeOf(child)

Q44. Define a class and instantiate it into an object.

Given: class User { constructor(id, name){ this.id=id; this.name=name; } }
Goal: new User(1, 'Sam') → { id:1, name:'Sam' }

Q45. Build a frequency object (histogram) from an array of strings.

Given: arr = ['x','y','x','z']
Goal: Return { x: 2, y: 1, z: 1 }

Q46. Toggle a boolean property in an object and return a new object (immutably).

Given: settings = { wifi: true, bt: false }
Goal: Toggle bt → { wifi: true, bt: true }

Q47. Compute cart total from an array of item objects (sum price * qty).

Given: cart = [{ price: 50, qty: 2 }, { price: 30, qty: 1 }]
Goal: Return 130

Q48. Remove a deeply nested key (delete by path) and return new object.

Given: obj = { a: { b: { c: 1, d: 2 } } }, path = 'a.b.c'
Goal: Return { a: { b: { d: 2 } } }

Q49. Replace keys using a mapping object (rename multiple keys).

Given: user = { fn: 'Asha', ln: 'Sharma' }, map = { fn: 'firstName', ln: 'lastName' }
Goal: Return { firstName: 'Asha', lastName: 'Sharma' }

Q50. Validate an object against a simple schema object of expected types (string, number, boolean).

Given: obj = { name: 'Neo', age: 30, active: true }, schema = { name: 'string', age: 'number', active: 'boolean' }
Goal: Return true