**⚔️ 2-Day Hardcore Practice Sheet (80+ Questions)**

**✅ Day 1 Topics & Questions**

**1**. Closure (10 Questions)

1. Create a function that returns another function to add 5 to a number.
2. Build a counter with private count using closure.
3. Return multiple functions from one closure.
4. Create a multiplier using closure (double, triple, etc.).
5. Implement a closure inside a loop (fix the classic var problem).
6. Use closure to encapsulate secret message (get/set methods).
7. Use setTimeout in closure to print variable after 1s delay.
8. Write a function that remembers the last argument passed to it.
9. Return a function that maintains running sum of passed numbers.
10. Closure to create a function that executes only once.

**2. Callback Functions (8 Questions)**

1. Create a function that takes another function and calls it after a delay.
2. Implement your own version of map using callbacks.
3. Write a function that accepts a callback and data and logs modified data.
4. Create a calculator that accepts operation as a callback.
5. Pass callback to sort an array of objects by key.
6. Call multiple callbacks in order from a main function.
7. Implement a mini event system with register/trigger using callbacks.
8. Create a button click simulator function with a callback.

**3. IIFE (6 Questions)**

1. Write a simple IIFE to log "Hello World".
2. Use IIFE to create private variable counter.
3. Create a function using IIFE to hide data (e.g. password).
4. Use IIFE to initialize a module with private config.
5. Create a stopwatch with start/stop using IIFE.
6. IIFE that returns an object with get/set methods.

**4. OOP + Getter/Setter (10 Questions)**

1. Create a Person class with name and age.
2. Add getter/setter for full name in an object.
3. Implement a class with private variable using #.
4. Create a BankAccount class with deposit and withdraw.
5. Implement a User class with read-only email.
6. Add computed property using getter.
7. Create setter that modifies data (e.g., capitalizes name).
8. Create method in class that uses this to access other properties.
9. Inherit from class and override method.
10. Use getters/setters to track access count to a property.

**5. Promise (8 Questions)**

1. Create a Promise that resolves after 2 seconds.
2. Reject a Promise conditionally based on random number.
3. Chain .then() to modify resolved data.
4. Wrap setTimeout in a Promise.
5. Create a function that returns a Promise and resolves square of a number.
6. Handle Promise with .catch() properly.
7. Add .finally() to a promise example.
8. Create a fake API call function using Promise.

**✅ Day 2 Topics & Questions**

**6. Promise Chaining (6 Questions)**

1. Chain three .then() methods to transform a number.
2. Create Promise chain that simulates async steps: login → fetch data → show.
3. Return nested Promises and flatten them using chaining.
4. Use .then() to log each step with delay.
5. Chain with mix of sync and async return values.
6. Handle error in the middle of a chain.

**7. Async/Await (8 Questions)**

1. Rewrite a Promise function using async/await.
2. Await two time-based tasks sequentially.
3. Await a fake API call and log result.
4. Use try/catch in async function.
5. Return value from async function and use it.
6. Call async function inside another function.
7. Combine async/await with normal code and see order.
8. Handle error using try/catch and finally in async function.

**8. Parallel Execution (6 Questions)**

1. Run 3 promises in parallel using Promise.all.
2. Use Promise.allSettled and handle results.
3. Compare sequential vs parallel execution timing.
4. Create delayed Promises and run them in parallel.
5. Use Promise.race to return fastest result.
6. Cancel slow promise using Promise.race trick.

**9. API Basics (6 Questions)**

1. Explain what an API is using an example function.
2. Simulate calling an API function (e.g., getUser).
3. Create a function that accepts endpoint and prints result.
4. Simulate API error and handle it.
5. Build a function that takes method (GET/POST) and simulates API.
6. Explain request-response using dummy JS code.

**10. Fetch API – GET/POST (10 Questions)**

1. Use fetch to GET data from JSONPlaceholder.
2. Use fetch to POST data to dummy endpoint.
3. Create a function that calls any API and logs JSON.
4. Add headers in fetch GET request.
5. Send POST with Content-Type: application/json.
6. Handle fetch errors using .catch() and status check.
7. Show loader before fetch, hide after.
8. Create function that fetches data and updates DOM.
9. Use async/await to fetch data.
10. Chain multiple fetch calls (e.g. user → posts → comments).

**📦 API Practice: GET Requests**

1. **Joke Generator API (GET)**  
   🔗 https://official-joke-api.appspot.com/jokes/random  
   🧠 *Task:* API call karke joke print karo — setup + punchline.
2. **User List API (GET)**  
   🔗 https://jsonplaceholder.typicode.com/users  
   🧠 *Task:* API call karke sab users ka naam aur email list karo.
3. **Advice API (GET)**  
   🔗 https://api.adviceslip.com/advice  
   🧠 *Task:* Ek button pe click se advice laake dikhana hai.
4. **Cat Fact API (GET)**  
   🔗 https://catfact.ninja/fact  
   🧠 *Task:* Cat ka ek fact fetch karo aur page pe show karo.
5. **Random Dog Image (GET)**  
   🔗 https://dog.ceo/api/breeds/image/random  
   🧠 *Task:* Ek image tag me dog photo dikhani hai.

**📨 API Practice: POST Requests**

1. **Create New Post (POST)**  
   🔗 https://jsonplaceholder.typicode.com/posts  
   🧠 *Task:* Ek dummy object (title, body, userId) POST karo.

{

title: "Bhai ka post",

body: "Yeh content bhai ne bheja hai",

userId: 1

}

1. **Send Form Data (POST)**  
   🔗 https://reqres.in/api/users  
   🧠 *Task:* Ek form banao (name, job), submit pe POST call maro.
2. **Fake Login (POST)**  
   🔗 https://reqres.in/api/login  
   🧠 *Task:* POST this data to get token:

{

"email": "eve.holt@reqres.in",

"password": "cityslicka"

}