■ JavaScript Async Cheat Sheet

1. Promise Basics

```
const p = new Promise((resolve, reject) => {
  resolve("Success");
});

p.then(res => console.log(res)) // Success
 .catch(err => console.log(err));
// States: pending → fulfilled or rejected
// .then() for success, .catch() for errors, .finally() always runs
```

2. Async / Await

```
async function getData() {
  try {
    const res = await Promise.resolve("Done");
    console.log(res);
  } catch (err) {
    console.log(err);
  }
}
getData(); // Done
// async makes function return a Promise
// await pauses execution until resolved/rejected
```

3. Event Loop Priority

```
Order of execution:
1. Synchronous code
2. Microtasks (Promises, await)
3. Macrotasks (setTimeout, setInterval)
```

4. Tricky Example

```
console.log("Start");

setTimeout(() => console.log("setTimeout"), 0);

Promise.resolve().then(() => console.log("Promise then"));

(async function test() {
  console.log("Inside async");
  await Promise.resolve();
  console.log("After await");
})();

console.log("End");

// Output:
// Start
// Inside async
// End
// Promise then
// After await
// setTimeout
```

5. Quick Interview Wrap-Up

```
- Promises solve callback hell
- Async/await is syntactic sugar over Promises → cleaner code
- Event loop: sync → microtasks → macrotasks
```