**🎯 Scenario: Fetching user data in steps**

1. Get user ID
2. Fetch user name based on ID
3. Fetch user email based on name

**✅ Using Callbacks (Callback Hell Example)**

function getUserId(callback) {

setTimeout(() => {

console.log("Step 1: Got user ID");

callback(101);

}, 1000);

}

function getUserName(id, callback) {

setTimeout(() => {

console.log(`Step 2: Got user name for ID ${id}`);

callback("John");

}, 1000);

}

function getUserEmail(name, callback) {

setTimeout(() => {

console.log(`Step 3: Got email for ${name}`);

callback("john@example.com");

}, 1000);

}

getUserId((id) => {

getUserName(id, (name) => {

getUserEmail(name, (email) => {

console.log(`Final Result: ${email}`);

});

});

});

👎 This gets messy quickly and is hard to maintain — **Callback Hell**.

**✅ Using Promises (Cleaner and More Readable)**

function getUserId() {

return new Promise((resolve) => {

setTimeout(() => {

console.log("Step 1: Got user ID");

resolve(101);

}, 1000);

});

}

function getUserName(id) {

return new Promise((resolve) => {

setTimeout(() => {

console.log(`Step 2: Got user name for ID ${id}`);

resolve("John");

}, 1000);

});

}

function getUserEmail(name) {

return new Promise((resolve) => {

setTimeout(() => {

console.log(`Step 3: Got email for ${name}`);

resolve("john@example.com");

}, 1000);

});

}

getUserId()

.then(getUserName)

.then(getUserEmail)

.then((email) => {

console.log(`Final Result: ${email}`);

});

✅ Much cleaner, easier to **read**, **maintain**, and **debug**.

**🔁 Optional: Even Cleaner with async/await**

async function getUserDetails() {

const id = await getUserId();

const name = await getUserName(id);

const email = await getUserEmail(name);

console.log(`Final Result: ${email}`);

}

getUserDetails();

🎓 **Conclusion for Students**:

* Callbacks **work**, but quickly become hard to manage when operations are nested.
* Promises make your code **cleaner, flatter, and more maintainable**.
* async/await makes asynchronous code look **synchronous and more intuitive**.

Would you like a visual diagram or slide to go with this for your class?