**Explaining async and await:**

Imagine you want to **make a sandwich** 🥪, but first, you need to **toast the bread** 🍞.

1️. **Regular Way (Synchronous)**:

* You stand there **waiting** ⏳ for the bread to toast before making the sandwich.
* You do **nothing else** while waiting.
* This is like **normal (blocking) JavaScript code**.

console.log("Start making sandwich");

console.log("Toasting bread..."); // Waiting...

console.log("Bread is ready!");

console.log("Add butter and jam");

console.log("Sandwich is ready!");

📌 **Problem?** You **waste time** just waiting! 😴

2️. **Smart Way (Asynchronous - async/await)**:

* You **start toasting the bread** 🍞.
* While the bread is **toasting**, you **get butter & jam** 🍯.
* When the bread is **ready**, you quickly **finish the sandwich**!

function toastBread() {

return new Promise((resolve) => {

setTimeout(() => {

console.log("Bread is toasted! 🍞");

resolve();

}, 4000); // Takes 4 seconds

});

}

async function makeSandwich() {

console.log("Start making sandwich");

let bread = toastBread(); // Toasting in background

console.log("Getting butter & jam 🍯");

await bread; // Wait for toast to finish

console.log("Adding butter and jam");

console.log("Sandwich is ready! 🥪");

}

makeSandwich();

🔹 **What Happens?**  
✅ We **don't waste time** waiting! 🎉  
✅ We do **other tasks** while the bread is toasting.

**Final Summary -**

* **Normal code (synchronous)** = Stand & wait for toast 🥱.
* **async/await (asynchronous)** = Toast while preparing other stuff 🏃💨.
* **Faster & smarter!** 🚀

**Async and Await in JavaScript for Automation Testing 🚀**

**1️. What is async and await in JavaScript?**

In JavaScript, async and await are used to handle **asynchronous operations** in a cleaner and more readable way.

📌 **Why do we need it?**  
Automation testing often involves **waiting** for elements, API responses, or page loads. Instead of using callbacks or .then(), we use async/await to make the test flow **smoother and easier to read**.

**2️. Understanding Asynchronous JavaScript**

When testing a web page, some tasks take **time to complete**, such as:  
✅ **Waiting for an element to appear** 🔄  
✅ **Fetching API responses** 🌐  
✅ **Waiting for page navigation** 🕒

JavaScript does **not wait** for these tasks by default. Instead, it keeps running the next line of code, which can cause **flaky tests**.

🚨 **Example of a Problem in Synchronous Code:**

c

**Output:**

Start Test

Test Complete

Checking element... (after 2 sec) ❌ (Out of order)

👉 The test **does not wait** for setTimeout, which can cause race conditions in automation testing.

**3️. async and await – Making Code Wait Properly**

To fix this, we use async/await to **pause execution** until a task is done.

**🔹 async Function**

An async function always **returns a promise** and allows await inside it.

**🔹 await Keyword**

await **pauses** the function execution until the promise is resolved.

**Correct Example Using async/await:**

async function testFlow() {

console.log("Start Test");

await new Promise((resolve) => setTimeout(resolve, 2000)); // Wait for 2 sec

console.log("Checking element...");

console.log("Test Complete");

}

testFlow();

**Output:**

Start Test

(Waits 2 sec ⏳)

Checking element...

Test Complete ✅ (In correct order)

🚀 **Now the test waits properly before moving forward!**

**4. How async/await Works in API Testing**

Automation testers also use async/await for **API testing** with **Axios or Fetch**.

**Example: API Call Using async/await**

const axios = require("axios");

async function testApi() {

try {

let response = await axios.get("https://jsonplaceholder.typicode.com/posts/1");

console.log("Test Passed:", response.data);

} catch (error) {

console.log("Test Failed:", error);

}

}

testApi();

**What Happens?**

1️. **Sends a GET request to an API**  
2️. **Waits** for the response (await axios.get(...))  
3️. **Prints the response data**

✅ **Without await, the script might try to read the response before it's available.**

**6️. Key Benefits of async/await in Automation Testing**

✅ **More Readable** → Looks like normal step-by-step code.  
✅ **No Callbacks** → Avoids the messy "callback hell".  
✅ **Easier Debugging** → Errors can be caught using try...catch.  
✅ **Prevents Flaky Tests** → Ensures elements are ready before actions.

**7️. Common Mistakes and How to Fix Them**

**❌ Forgetting await**

async function test() {

let data = fetch("https://api.example.com/data"); // ❌ Forgot await

console.log(data); // Will print a pending Promise instead of actual data

}

test();

✅ **Fix:** Add await

async function test() {

let data = await fetch("https://api.example.com/data"); // ✅ Correct

console.log(await data.json());

}

test();

**❌ Using await outside an async Function**

let data = await fetch("https://api.example.com/data");

// ❌ Error: Unexpected await

✅ **Fix:** Wrap in an async function

async function fetchData() {

let data = await fetch("https://api.example.com/data");

console.log(await data.json());

}

fetchData();

**8️ Summary**

| **Feature** | **async** | **await** |
| --- | --- | --- |
| What is it? | Declares an asynchronous function | Waits for a Promise inside async function |
| Purpose | Allows await inside it | Pauses execution until a promise is resolved |
| Usage | async function myFunc() {} | await somePromise(); |
| Benefit | Makes code easier to read | Prevents flaky tests by waiting |

**9️Final Thought 💡**

async/await is **essential** for automation testers because:  
✅ It **waits** for elements before interaction.  
✅ It **handles API calls** without unnecessary delays.  
✅ It **prevents race conditions** in tests.  
✅ It makes test scripts **cleaner & more reliable**.