

# JavaScript Async Deep Dive

## Callback → Promise → Async/Await (Interview + Practical)

---

### 1. CALLBACK (Old but Foundation)

#### What is Callback? (Human explanation)

A callback is a function passed to another function and executed **after** an async task completes.

**Real-life example:** "Tea ready aaguna apram sollu"

#### Interview one-liner

A callback is a function executed after the completion of an asynchronous operation.

#### Program 1 – Basic Callback (Runnable)

```
function getDataCallback(callback) {
  setTimeout(() => {
    callback("Data loaded using Callback");
  }, 2000);
}

getDataCallback((result) => {
  console.log(result);
});
```

**Output:** Data loaded using Callback

#### Drawback

- Callback Hell
- Hard to read & debug

---

### 2. PROMISE (Improved Solution)

#### What is Promise? (Human explanation)

A Promise is a JS object that represents a value **available now, later, or never**.

## Promise States

- Pending
- Fulfilled
- Rejected

## Interview one-liner

Promise represents the eventual completion or failure of an async operation.

## Program 2 – Promise Version

```
function getDataPromise() {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      resolve("Data loaded using Promise");  
    }, 2000);  
  });  
}  
  
getDataPromise()  
  .then(result => console.log(result))  
  .catch(err => console.log(err));
```

Output: Data loaded using Promise

## then() Explanation (Very Important)

- `.then()` runs **only when promise is resolved**
- It receives the value passed to `resolve()`

---

## 3. ASYNC / AWAIT (Modern & Best Practice)

### What is async/await? (Human explanation)

Async/await allows us to write async code **like synchronous code**.

## Interview one-liner

Async/await is syntactic sugar over promises.

## Program 3 – Async/Await Version

```
function getDataPromise() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve("Data loaded using Async/Await");
    }, 2000);
  });
}

async function showData() {
  const result = await getDataPromise();
  console.log(result);
}

showData();
```

**Output:** Data loaded using Async/Await

## 4. FETCH API (Real-World)

### fetch() Intro

- Introduced around **2015**
- Built-in Web API
- **Promise-based**

### Interview one-liner

fetch returns a Promise and is used for making HTTP requests.

## 5. FETCH USING 3 METHODS

### A) fetch + Callback (Simulated)

```
function fetchUsersCallback(callback) {
  fetch("https://jsonplaceholder.typicode.com/users")
    .then(res => res.json())
    .then(data => callback(null, data))
    .catch(error => callback(error, null));
}

fetchUsersCallback((error, data) => {
  if (error) {
    console.log("Error:", error);
  }
});
```

```
    } else {  
      console.log("Users:", data);  
    }  
  });  
});
```

**Why** `callback(null, data)`?

This follows **Node.js error-first convention**:

```
callback(error, result)
```

- Success → `null, data` - Failure → `error, null`

## Interview line

First parameter is reserved for error handling consistency.

---

## B) fetch + Promise

```
function fetchUsersPromise() {  
  return fetch("https://jsonplaceholder.typicode.com/users")  
    .then(res => res.json());  
}  
  
fetchUsersPromise()  
  .then(data => console.log(data))  
  .catch(err => console.log(err));
```

## C) fetch + Async/Await (Recommended)

```
async function fetchUsersAsync() {  
  try {  
    const res = await fetch("https://jsonplaceholder.typicode.com/users");  
    const data = await res.json();  
    console.log(data);  
  } catch (error) {  
    console.log(error);  
  }  
}
```

```
fetchUsersAsync();
```

## 6. COMPARISON (Interview Table)

Feature	Callback	Promise	Async/Await
Readability	✗	♀	✓
Error Handling	✗	catch	try/catch
Modern Usage	✗	♀	✓

## 7. ASSIGNMENTS (IMPORTANT)

### CALLBACK – 5 Programs

1. setTimeout callback print message
2. Callback-based calculator
3. Nested callback example
4. Callback error handling
5. Simulate API using callback

### PROMISE – 5 Programs

1. Promise resolve after delay
2. Promise reject example
3. Promise chaining
4. Promise.all usage
5. Convert callback to promise

### ASYNC/AWAIT – 5 Programs

1. Async function returning value
2. try/catch error handling
3. Sequential await calls
4. Parallel await using Promise.all
5. Convert promise to async/await

## FINAL INTERVIEW SUMMARY

Callback is the base, Promise is the improvement, Async/Await is the professional standard used in modern JavaScript projects.