JavaScript Callback → Promise → Async/Await Conversions — Complete Master Guide

Visual Diagrams (Placed First for Better Understanding)

Diagram 1: Simple Callback Flow

```
function task(callback)
      │
      ▼
 task prepares work
      │
      ▼
callback(result)
```

Diagram 2: Nested Callbacks (Callback Hell)

```
step1(function(){
  step2(function(){
    step3(function(){
      console.log("done")
    })
  })
})
```

Structure:

```
step1 → step2 → step3
   │       │       │
   ▼       ▼       ▼
callback(callback(callback()))
```

Diagram 3: Turning a Callback into a Promise

```
callback version:
  step(cb) → cb(result)

promise version:
  step() → resolve(result)
```

Diagram 4: Promise Chaining → Async/Await

```
Promise Chain:
   step1()
     .then(step2)
     .then(step3)

Async/Await:
   const a = await step1()
   const b = await step2(a)
   const c = await step3(b)
```

Diagram 5: Full Flow Transformation

```
Callback Hell
   ▼
Promise Chain
   ▼
Async/Await
```

---

1. Understanding the Problem — Callback Hell

Problem

Callbacks become deeply nested and hard to read.

Example

```
function getData(callback) {
  setTimeout(() => {
    callback("Data loaded");
  }, 1000);
}

getData(function(result) {
  console.log(result);
});
```

Issue

- Hard to maintain
- Hard to handle errors
- Not scalable

---

2. Convert Callback → Promise

Promise Version

```
function getData() {
  return new Promise(function(resolve, reject) {
    setTimeout(() => {
      resolve("Data loaded");
    }, 1000);
  });
}

g etData().then(res => console.log(res));
```

Why this is better

- Cleaner
- Can chain
- Easy error handling with catch()

---

3. Convert Promise → Async/Await

Async/Await Version

```
function getData() {
  return new Promise(resolve => {
    setTimeout(() => resolve("Data loaded"), 1000);
  });
}

async function show() {
  let result = await getData();
  console.log(result);
}

show();
```

Why async/await is best

- Looks like synchronous code
- Very readable
- Easy error handling using try/catch

---

4. Complex Callback → Promise → Async/Await Conversion

Problem Code (Nested callbacks)

```
function step1(cb) {
  setTimeout(() => cb("Step 1 done"), 1000);
}

function step2(cb) {
  setTimeout(() => cb("Step 2 done"), 1000);
}

step1(function(res1) {
  console.log(res1);
  step2(function(res2) {
    console.log(res2);
  });
});
```

Promise Version

```
function step1() {
  return new Promise(resolve => {
    setTimeout(() => resolve("Step 1 done"), 1000);
  });
}

function step2() {
  return new Promise(resolve => {
    setTimeout(() => resolve("Step 2 done"), 1000);
  });
}

step1()
  .then(res => {
    console.log(res);
    return step2();
  })
  .then(res => console.log(res));
```

Async/Await Version

```
async function runSteps() {
  let r1 = await step1();
  console.log(r1);

  let r2 = await step2();
  console.log(r2);
}

runSteps();
```
