Callbacks

A function passed into another function.

```
function greet(name, callback) {
  console.log("Hi " + name);
  callback();
}

function sayBye() {
  console.log("Bye!");
}

greet("Sunny", sayBye);
// Output: Hi Sunny
//         Bye!
```

Callback Hell

- Callback Hell : Nested callbacks stacked below one another forming a pyramid structure.
- Too many nested callbacks (bad structure):
- (Pyramid of Doom)

Promises

- Promise is for “eventual” completion of task. It is an object in JS.
- It is a solution to callback hell.

```
let promise = new Promise((resolve, reject) => {
  let success = true;
  if (success) resolve("Done!");
  else reject("Error");
});

promise
  .then(result => console.log(result))  // if success
  .catch(error => console.log(error)); // if failure
```

Async-Await

- async function always returns a promise.
- await pauses the execution of its surrounding async function until the promise is settled.

```
async function fetchData() {
  try {
    let response = await fetch("https://api.example.com/data");
    let data = await response.json();
    console.log(data);
  } catch (err) {
    console.log("Error:", err);
  }
}
```

⚡ IIFE (Immediately Invoked Function Expression)

- A function that runs as soon as it’s defined.

```
(function () {
  console.log("I run immediately!");
})();
```