## Callbacks

## Promises

## async/await

## Sample & Code Snippets

### Sample for chaining multiple callbacks

| // Function to simulate fetching data asynchronously function fetchData(url) {  return new Promise((resolve, reject) => {  setTimeout(() => {  const data = { id: 1, name: "John Doe" };  resolve(data);  }, 1000); // Simulating delay with setTimeout  }); }  // Function to simulate processing data asynchronously function processData(data) {  return new Promise((resolve, reject) => {  setTimeout(() => {  data.age = 30;  resolve(data);  }, 1000); // Simulating delay with setTimeout  }); }  // Function to simulate saving data asynchronously function saveData(data) {  return new Promise((resolve, reject) => {  setTimeout(() => {  console.log("Data saved successfully:", data);  resolve();  }, 1000); // Simulating delay with setTimeout  }); }  // Chaining async callbacks fetchData('https://example.com/data')  .then(data => {  console.log("Data fetched:", data);  return processData(data);  })  .then(processedData => {  console.log("Data processed:", processedData);  return saveData(processedData);  })  .then(() => {  console.log("All operations completed successfully.");  })  .catch(error => {  console.error("An error occurred:", error);  }); |
| --- |

### Sample for waiting all process from a loop completed before continuing

async function test() {

console.log("Starting the app..");

const listOfThingsToDo = ["Writing", "Cooking", "Running"];

try {

**//must use map instead of foreach**

const promises = listOfThingsToDo.map(async (activity) => {

await doSomething(activity);

})

await Promise.all(promises).then(() => {

console.log("do other things only after all doSomething completed");

});

} catch (error) {

}

}

async function doSomething(activity) {

console.log("Doing " + activity + "...");

await new Promise((wait) => setTimeout(wait, 1000));

console.log(activity + " done.");

const isHeads = Boolean(Math.round(Math.random()));

return isHeads;

}

test();