Skills Network

Developing Back-End Apps with Node.js and Express
Module 2 Cheat Sheet: Asynchronous I/O with Callback Program

Package/Method	Description	Code Example

Async-await	We can await promises as long as they are being called inside asynchronous functions.	`const axios = require('axios').default;`
		`let url = "some remote url"`
		`async function asyncCall() {`
		`  console.log('calling');`
		`  const result = await axios.get(url);`
		`  console.log(result.data);`
		`}`
		`asyncCall();`
		Copied!

Callback	Callbacks are methods that are passed as parameters. They are invoked within the method to which they are passed as a parameter, conditionally or unconditionally. We use callbacks with a promise to process the response or errors.	`//function(res) and function(err) are the anonymous callback functions`
		`axios.get(url).then(function(res) {`
		`    console.log(res);`
		`}).catch(function(err) {`
		`    console.log(err)`
		`})`
		Copied!

Promise	An object that is returned by some methods, representing eventual completion or failure. The code continues to run without getting blocked until the promise is fulfilled or an exception is	`axios.get(url).then(`
		`//do something`
		`).catch(`
		`//do something`
		`)`
		Copied!

thrown.

Promise use case

Promises are used when the processing time of the function we invoke takes time like remote URL access, I/O operations file reading, etc.

```
let prompt = require('prompt-sync')();
let fs = require('fs');
const methCall = new Promise((resolve,reject)=>{
    let filename = prompt('What is the name of the file ?');
    try {
      const data = fs.readFileSync(filename, {encoding:'utf8', flag:'r'});
      resolve(data);
    } catch(err) {
      reject(err)
    }
});
console.log(methCall);
methCall.then(
  (data) => console.log(data),
  (err) => console.log("Error reading file")
);
```

Copied!

object.on()

It defines an event handler that the framework calls when an event occurs

```
http.request( options, function(response) {
 let buffer = ‘’;
 ...
 response.on('data', function(chunk) {
   buffer += chunk;
 });
 response.on('end', function() {
   console.log(buffer);
 });
}).end();

```

Copied!

Callback Hell/The Pyramid of Doom

Nested callbacks stacked below one another and waiting for the previous callback. This creates a pyramid structure that affects the readability and maintainability of the code.

```
const makeCake = nextStep => {
  buyIngredients(function(shoppingList) {
    combineIngredients(bowl, mixer, function(ingredients){
      bakeCake(oven, pan, function(batter) {
        decorate(icing, function(cake) {
          nextStep(cake);
        });
      });
    });
  });
};
```

Copied!

Axios Request

The axios package handles HTTP requests and returns a promise object.

```
const axios = require('axios').default;
const connectToURL=(url)=>{
  const req=axios.get(url);
  console.log(req);
  req.then(resp=>{
  console.log("Fulfilled");
  console.log(resp.data);
  })
  .catch(err=>{
  console.log("Rejected");
  });
}
connectToURL('valid-url')
connectToURL('invalid-url')
```

Copied!

Changelog

Date	Version	Changed by	Change Description
04-07-2022	1.0	Pallavi	Initial version created
18-10-2022	1.1	K Sundararajan	Cheatsheet updated
17-11-2022	1.2	K Sundararajan	IDSN logo updated based on Beta feedback
29-11-2022	1.3	K Sundararajan	Title updated based on Beta feedback