



## Developing Back-End Apps with Node.js and Express

### Module 2 Cheat Sheet: Asynchronous I/O with Callback Program

Package/Method	Description	Code Example
<b>Async-await</b>	We can await promises as long as they are being called inside asynchronous functions.	<pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 </pre>
		<pre> 1. const axios = require('axios').default; 2. let url = "some remote url" 3. async function asyncCall() { 4.   console.log('calling'); 5.   const result = await axios.get(url); 6.   console.log(result.data); 7. } 8. asyncCall(); </pre>
<b>Callback</b>	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.	<div>Copied!</div> <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 </pre>
		<pre> 1. //function(res) and function(err) are the anonymous callback functions 2. axios.get(url).then(function(res) { 3.   console.log(res); 4. }).catch(function(err) { 5.   console.log(err) 6. }) </pre>
<b>Promise</b>	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 thrown.	<div>Copied!</div> <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 </pre>
		<pre> 1. axios.get(url).then( 2.   //do something 3. ).catch( 4.   //do something 5. ) </pre>
<b>Promise use case</b>	Promises are used when the processing time of the function we invoke takes time like remote URL access, I/O operations file reading, etc.	<div>Copied!</div> <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 14. 14 15. 15 16. 16 </pre> <pre> 1. let prompt = require('prompt-sync')(); 2. let fs = require('fs'); 3. const methCall = new Promise((resolve, reject)=&gt;{ 4.   let filename = prompt('What is the name of the file ?'); 5.   try { 6.     const data = fs.readFileSync(filename, {encoding:'utf8', flag:'r'}); 7.     resolve(data); 8.   } catch(err) { 9.     reject(err) 10.  } 11. }); 12. console.log(methCall); 13. methCall.then( 14.   (data) =&gt; console.log(data), 15.   (err) =&gt; console.log("Error reading file") </pre>

16. );

Copied!

1. 1  
2. 2  
3. 3  
4. 4  
5. 5  
6. 6  
7. 7  
8. 8  
9. 9  
10. 10  
11. 11

**object.on()**

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

```
1. http.request( options, function(response) {  
2.   let buffer = '';  
3.   ...  
4.   response.on('data', function(chunk) {  
5.     buffer += chunk;  
6.   });  
7.   response.on('end', function() {  
8.     console.log(buffer);  
9.   });  
10. }).end();  
11.
```

Copied!

1. 1  
2. 2  
3. 3  
4. 4  
5. 5  
6. 6  
7. 7  
8. 8  
9. 9  
10. 10  
11. 11

**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.

```
1. const makeCake = nextStep => {  
2.   buyIngredients(function(shoppingList) {  
3.     combineIngredients(bowl, mixer, function(ingredients){  
4.       bakeCake(oven, pan, function(batter) {  
5.         decorate(icing, function(cake) {  
6.           nextStep(cake);  
7.         });  
8.       });  
9.     });  
10.   });  
11. };  

```

Copied!

1. 1  
2. 2  
3. 3  
4. 4  
5. 5  
6. 6  
7. 7  
8. 8  
9. 9  
10. 10  
11. 11  
12. 12  
13. 13  
14. 14

**Axios Request**

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

```
1. const axios = require('axios').default;  
2. const connectToURL=(url)=>{  
3.   const req=axios.get(url);  
4.   console.log(req);  
5.   req.then(resp=>{  
6.     console.log("Fulfilled");  
7.     console.log(resp.data);  
8.   })  
9.   .catch(err=>{  
10.    console.log("Rejected");  
11.   });  
12. }  
13. connectToURL('valid-url')  
14. 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

© IBM Corporation 2022. All rights reserved.