Callback function in JavaScript

When a function simply accepts another function as an argument, this contained function is known as a callback function

Using callback functions we can achieve functional programming concept, and you can find them in most JavaScript code

Here is an example by which we can understand how it works. As function in JavaScript is considered as an object so we can pass is as an argument to another function.

function wishing (name, msg, say)

{

say (msg);

}

function sayHello (msg) {

console.log (msg);

}

wishing ('abc','gd afternoon', sayHello);

Example for anonymous function as callback.

SetTimeout (function () {

console.log ("Water plant")

}, 3000) ;}

Promises in JavaScript

A promise is used to handle the asynchronous result of an operation. JavaScript is designed to not wait for an asynchronous block of code to completely execute before other synchronous parts of the code can run. For instance, when making API requests to servers, we have no idea if these servers are offline or online, or how long it takes to process the server request. So we create a promise object and chain .**then ()** and .**catch ()** to our promise object.

**then** () receives a function with an argument which is the resolve value of our promise. **catch** () returns the reject value of our promise.

**Syntax for creating a promise**

const myNum = new Promise (function (resolve, reject) {});

**Promises have three states**:

* **Pending**: This is the initial state of the Promise before an operation begins
* **Fulfilled**: This means the specified operation was completed
* **Rejected**: The operation did not complete an error value is usually thrown

Example:-

var num = 15;

const myNum = new Promise (function (evenNum, oddNum ){

console.log ('wait is it in process');

if(num % 2 == 0 )

{

evenNum (num);

}

else

{

OddNum (odd);

}

});

const abc = myNum.then (evenNum,oddNum);

function evenNum (even)

{

console.log (even +' is Even');

}

function oddNum (odd)

{

console.error (odd +' is Odd');

}

Fetch API

**fetch ()**allows you to make network requests similar to XMLHttpRequest (XHR). The main difference is that the Fetch API uses Promises, which enables a simpler and cleaner API, avoiding callback hell and having to remember the complex API of XMLHttpRequest.

The response of a fetch () request is a [promise](https://streams.spec.whatwg.org/) object, which means that when we call the **json** () method, a Promise is returned since the reading of the stream will happen asynchronously.

**A sample code for Example Fetch API**

function getData (){

fetch ('test.json').then (function (result) {

console.log (result);

return result.json ();

}).then (function (data){

console.log (data);

const newData = `<ul>

<li> name : ${data.name}</li>

<li> roll : ${data.roll}</li>

<li> location : ${data.location}</li>

</ul>`

document.getElementById ('studentData').innerHTML = newData;

})

}

var data = document.getElementById('getData');

data.addEventListener('click',getData);

HTML code for submit button from where we call our javascript method

<input id="getData"type="submit"

Style="border: 1px solid #48C9B0;

border-radius: 1px;

height: 40px;

width: 280px;

margin: 30px;

background: #48C9B0;

font-family: Century Gothic;

font-size: 20px;

color: white"

value="Get student data">