Node js Modules

#Node JS Notes

JavaScript Callbacks

- Callbacks are a great way to handle something after something else has been completed. By something here we mean a function execution.
- Callback Function : “A function is a block of code that performs a certain task when called. “

Benefit of Callback Function

- The benefit of using a callback function is that you can wait for the result of a previous function call and then execute another function call.

JS demo.js

JS demo.js > ...

```javascript
// function
function greet(name, callback) {
    console.log('Hi' + ' ' + name);
    callback();
}
// callback function
function callMe() {
    console.log('I am callback function');
}
// passing function as an argument
greet('Akash', callMe);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\filedemo>node demo.js
Hi Akash
I am callback function

D:\filedemo>
```

Example

```
// function
function greet(name, callback) {
    console.log('Hi' + ' ' + name);
    callback();
}
// callback function
function callMe() {
    console.log('I am callback function');
}
// passing function as an argument
greet('Akash', callMe);
```

Program with setTimeout()

JS demo.js

JS demo.js > ...

```js
//  program that shows the delay in execution
function greet() {
    console.log('Hello world');
}
function sayName(name) {
    console.log('Hello' + ' ' + name);
}
// calling the function
setTimeout(greet, 2000);
sayName('Akash');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\filedemo>node demo.js
Hello Akash
Hello world

D:\filedemo>
```

- Function is called after **2000** milliseconds (**2** seconds)

```
//  program that shows the delay in execution
function greet() {
   console.log('Hello world');
}
function sayName(name) {
   console.log('Hello' + ' ' + name);
}
// calling the function
setTimeout(greet, 2000);
sayName('Akash');
```

setTimeout(cb, ms)

- The setTimeout() calls a function (cb) after a specified number of milliseconds (ms).
- The timeout must be in the range of 1-2,147,483,647 inclusive.
- If the value is outside that range, it's changed to 1 millisecond.

- setTimeout(function, milliseconds);
  - function - a function containing a block of code
  - milliseconds - the time after which the function is executed

Example

```
setTimeout(function(){
    console.log('I have come after 500 miliseconds')},500);
```

JS demo.js

JS demo.js > ...

```
1   setTimeout(function(){
2       console.log('I have come after 500 miliseconds')},500);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: cr

```
D:\filedemo>node demo.js
I have come after 500 miliseconds
```

Example

```
// program to display time every 3 seconds
function showTime() {
    // return new date and time
    let dateTime= new Date();
    // returns the current local time
    let time = dateTime.toLocaleTimeString();
    console.log(time)
    // display the time after 3 seconds
     setTimeout(showTime, 3000);
}
// calling the function
showTime();
```

JS demo.js

JS demo.js > showTime

```
1   // program to display time every 3 seconds
2   function showTime() {
3       // return new date and time
4       let dateTime= new Date();
5       // returns the current local time
6       let time = dateTime.toLocaleTimeString();
7       console.log(time)
8       // display the time after 3 seconds
9        setTimeout(showTime, 3000);
10  }
11  // calling the function
12  showTime();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\filedemo>node demo.js
12:48:00 pm
12:48:03 pm
12:48:06 pm
```

clearTimeout(t)

- The clearTimeout() is used to cancel a timeout that was set with setTimeout(). The callback will not execute.

setInterval(cb, ms)

- setinterval() calls a function (cb) repeatedly at specified intervals (in milliseconds (ms)).
- The interval must be in the range of 1-2,147,483,647 inclusive.
- If the value is outside that range, it's changed to 1 millisecond.

Example

```
setInterval(function(){
    console.log('Welcome to Node.js')
    }, 500);
```

JS demo.js

JS demo.js > ...

```
1
2  setInterval(function(){
3      console.log('Welcome to Node.js')
4      }, 500);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

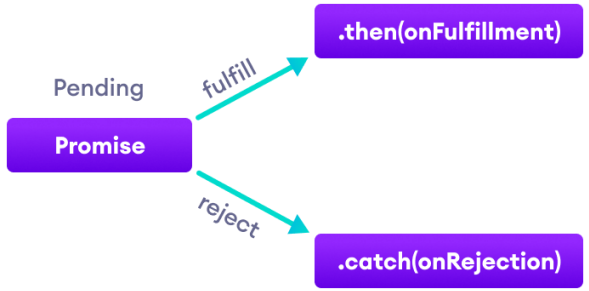
```
D:\filedemo>node demo.js
Welcome to Node.js
Welcome to Node.js
Welcome to Node.js
Welcome to Node.js
Welcome to Node.js
Welcome to Node.js
```

clearInterval(t)

- The clearInterval() is used to stop a timer that was set with setInterval(). The callback will not execute.

Promises

- A promise is basically an advancement of callbacks in Node. While developing an application you may encounter that you are using a lot of nested callback functions.
- A promise is an object that allows you to handle asynchronous operations. It's an alternative to plain old callbacks.
- Promises have many advantages over callbacks. To name a few:
  - Make the async code easier to read.
  - Provide combined error handling.
  - Better control flow. You can have async actions execute in parallel or series.
  - Promises are used to handle asynchronous http requests.



Call Back vs Promises

```
a(() => {
  b(() => {
    c(() => {
      d(() => {
        // and so on ...
      });
    });
  });
});
```

```
Promise.resolve()
  .then(a)
  .then(b)
  .then(c)
  .then(d)
  .catch(console.error);
```

Syntax

- **then()** : is invoked when a promise is either resolved or rejected.
- **catch()** : is invoked when a promise is either rejected or some error has occured in execution.
- Syntax :

```
.then(function(result){
        //handle success
    }, function(error){
        //handle error
    })
```

- *catch()* is invoked when a promise is either rejected or some error has occured in execution.

Example

```
var mypromise = new Promise(function(resolve, reject) {
  const x = 100;
  const y = 100;
  if(x === y) {
    resolve();
  } else {
    reject();
  }
});
mypromise.
    then(function () {
        console.log('Success');
    }).
    catch(function () {
        console.log('Error');
    });
```

JS demo.js

JS demo.js > ...

```
1  var mypromise = new Promise(function(resolve, reject) {
2    const x = 100;
3    const y = 100;
4    if(x === y) {
5      resolve();
6    } else {
7      reject();
8    }
9  });
10
11 mypromise.
12     then(function () {
13         console.log('Success');
14     }).
15     catch(function () {
16         console.log('Error');
17     });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Success
```

Get Exclusive Video Tutorials

www.aptutorials.com

https://www.youtube.com/user/Akashtips

Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials

Get More Details

www.akashsir.com

If You Liked It !

Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad