# JavaScript Callback Function

A function that is passed as an argument to another function is known as **callback function**. That means, **callback functions** are just any JavaScript function, but can be used as argument for another function.

**Callback Functions** are high order functions. A **high order functions** in javascript includes both **callback functions** and **function that returns a function**

```
function main(x){
    x();
}
function callback(){
    console.log("hi");
}

main(callback);            // return "hi"
```

In the above example, main is a function with parameter x. Inside main function, we are calling `x`. This means x should be a function, not any other datatype. Argument of main function is `callback`. The above example is **synchronous callback** as the code execute immediately.

# Synchronous Callback

**Synchronous callback** is a callback where code execute immediately, i.e. synchronously. All the code is main thread will be executed line by line. See example

```
function showTime(x,done){
    console.log(" Show Time is at ", x);
    done();
}

function showEnd(){
```

```
        console.log("Show Ends")
    }
    showTime(9,showEnd);

    console.log("done");
```

Now lets execute this code one by one with callback.

Show Time is at 9

Show Ends

done

```
    showTime(9,showEnd)        // call showTime
```

The above code execute line by line. That's why, the last line, i.e. `console.log("done");` executes in the end.

# Asynchronous Callback

**Asynchronous callback** is a callback where code within callback is executed after main thread is completed. For **asynchronous callback**, we are using setTimeout timing function. This is non blocking approach. See example

## Blocking synchronous method

1

2

3

```
    console.log(1);
```

```
setTimeout(console.log(2));                    // without callback
console.log(3);
```

## Non blocking asynchronous method

1

3

2

```
console.log(1);
setTimeout(function(){console.log(2)});        // with callback
console.log(3);
```

**Asynchronous callback** is one of the main reason behind performance of JavaScript. The main thread executes first and the long waiting tasks are kept inside setTimeout timing function within **callback**. All non blocking methods are executed using **Asynchronous callback** in JavaScript and Node JS.

To avoid callback hell, we will use JavaScript promises in future for asynchronous operations.

# Parameters in callback function

To pass parameters in **callback function** of setTimeout function, use third parameter of setTimeout as parameter of callback function. See example

show starts at 9

```
function newShow(x){
    console.log("show starts at", x);
}
 setTimeout(newShow,0,9)

 // newShow is callback functions
```

```
    // 0 is time in ms
    // x is argument for callback function
```

## Another example of parameter in callback

Class starts at 10

Projector running using hdmi

```
function startClass(x,y,z){
    console.log(" Class starts at", x);
    y(z);
}

function onProjector(x){
    console.log("Projector running using ", x)
}

startClass(10,onProjector,"hdmi");
```

In the above example, startClass is the main function and onProjector is the callback. First parameter is parameter of startClass, second parameter is **callback function** and third parameter is parameter of **callback function**.

In ES6, we can use **JavaScript Promise** to handle long and nested Asynchronous operations.