Tutorial-3(**ANONAMOUS FUNCTIONS**)

We can assign function to variables: The advantage it offers is that you can pass the variable(containing) inside a function. For example: **setTimeout(navtesh,5000);**

Here navtesh is a variable to which a function is assigned.

For example🡺 var navtesh=function(){

Console.log(‘hi navtesh’);

};

So if I call, navtesh() the given function will print out**, it is not nessasary that every time we need to name the function** like function navtesh(),we can do as in the example, such functions are called **anonamous functions.**

**setTimeout(navtesh,5000); 🡺 it will run after 5sec of the last statement**

**TUTORIAL-4**

**Soooooooooooo?**

**setTimeout(navtesh,5000); 🡺 what is this function??????what this function does is that when**

**it just schedules the execution to be done after 5sec and goes on to the next statement immediately.**

**Tutorial-5**

**Everything is a reference.**

**For example:**

**Var navtesh= {**

**FavFood:’tuna’,**

**FavMovie:’fight club’**

**};**

**Var arsh = navtesh;**

**arsh.FavFood=’salad’;**

**console.log(navtesh.FavFood);**

**we expect it to print tuna, but it prints salad because “Var arsh = navtesh”**

**does not mean that arsh is a completely new object, it is just a reference to navtesh.**

**Difference between == and ===**

**== 🡺 compares only the value**

**===🡺compares values and data type**

**Tutorial-6(this key word)**

**‘this’ refers to what ever thing is calling it.**

**var navtesh={**

**fav: function(){**

**console.log('my name is navtesh');**

**console.log(this===navtesh) //true**

**}**

**};**

**navtesh.fav()**

**this refers to which object is calling that function.**

**So in navtesh.fav() which object is calling the fav function??? navtesh , so this ===navtesh.**

**But……..if it is not called by any of the objects then what is this ===?**

**function fav(){**

**console.log('my name is navtesh');**

**console.log(this===global); //true**

**};**

**fav();**

**\*in this the function fav is not called by any object so, this ===global.\***

**Tutorial-7(prototyping)**

What is **prototyping**????

Adding cool methods and properties to class(es).

**player.prototype.punch = function punch(opponent){**

**code inside function**

**}**

this adds the method punch to the player class.

We can access it by **navtesh.punch(opponent)**

it is like list.append in python, it adds the method punch in player class.

**Tutorial-8**

**Modules: Breaking the code and clustering the code related to a particular thing at one place. And then importing it from there to the main file.**

**1.Exporting some part of the file to the other file.**

**module.exports.name=functionName;**

**name is just a variable.**

**Functioname = name of the function you want to export.**

**2.importing the code.**

**When we import an another module we set it equal to an another variable.**

**Here movies.**

**Var movies =require(‘./filename’);**

**Filename= the name of file from which the code is imported.**

**HOW to use it in the imported file🡺**

**movies.name();**

**name=name of the variable in the file which is exported.**

**module.exports.name =functionName;**

**name is actually assigned a function.**

**Movies=name of variable in the impoted file**

**Tutorial-9**

**In the way discussed in last tutorial, for Every function we want to export ,we would need to assign it to a variable.**

**One more way of exporting:**

**In every file the is an default module.export={}; object.**

**What module.exports.name=functionName; does is that it adds the name:functionName as a method to the module.exports={} object.**

**But we can directly write our functions inside module.exports={} object.**

**module.exports={**

**name1:function(){}**

**name2=””;**

**.**

**}**

**In the file where we import the other file, how do we access the functions, variable .etc,??????????????**

**var varName=require(‘./ModuleName’);**

**now the varName contains all the functions, variables of the imported file.**

**Name2 is a variable defined in the imported file, varName contains all the functions, variables of the imported file.**

**So varName.Name2=”sometext”; is how we access that.**

**Tutorial-10**

**Every file that imports a module shares it, they don’t get a copy of their own.**

**Tutorial-11(object factory)**

**But if you want to have a copy of your own then just make an object in the module that makes other objects.**

**Tutorial-12(core modules)**

**Some modules come built-in with node.js🡺core**

**In core we don’t include the path, the varName equals to the module name(just a practice)**

**Some coremodules🡺**

1. **var fs=require('fs');**

**fs.writeFileSync("path","data");**

**this statement creates a file**

**path=file.type**

**data=contents of the file**

**console.log(fs.writeread(‘file name’).tostring)**

**function to read a file as a string**

**2. var path=require('path');**

**var websiteHome='desktop//navtesh/yo.html';**

**console.log(path.normalize(websiteHome));**

**console.log(path.dirname(websiteHome));**

**console.log(path.basename(websiteHome));**

**console.log(path.extname(websiteHome));**