Task 4 What are Callbacks?

Date / /
Page

Callback :- A Call Back function is a function passed into another function as an argument which is then invoked inside the outer function to complete an action.

→ Asynchronous actions are the actions that we initiate now and they finish later eg setTimeout

→ Synchronous actions are the actions that initiate and finish one-by-one.

```
// Synchronous :- Ex:-
let a = prompt ("what is your name ");
console.log (a + " Hi Good morning ");

~~function loadScript (src, callback){~~
~~let Script = document.createElement ('script')~~

Const posts=[
{ title : 'Post One',
   body : 'This is Post One',
   CreatedAt : new Date ().getTime()
},{
   title : 'Post Two',
   body : 'This is Post Two',
   CreatedAt : new Date ().getTime()
}];
```

Scanned with CamScanner

Date ___
Page ___

Vid :- Traversy media

```
const posts = [
    { title : 'post one', body : 'This is post one'},
    { title : 'post two', body : 'This   two'}

function getPosts (){
    Set TimeOut (() => {
      let output = '';
      posts . forEach ((post, index) => {
      output += '<li>${post.title}</li>';
    });
    document.body.innerHTML = output;
  }, 1000);
}
                                        Callback
function createPost (post){
  setTimeout (() => {
    posts . push (post); callback();
    }, 2000);
}

getPosts();
createPost ({title : 'post Three', body :
      'This is post three'});
                                     getPosts
```

Using callback in above Code
3 change

s is postone's
two?
ex) =>{
e}</li>;
ut;
&
();
body;
t posts

0- Call back Hell
1- Inversion of Control

Call Backs

Cosk's Vid-AS Ep 01 Season 02.

```
Const Cart = ["shoes", "pants", "kurta"];
→ api.createorder
→ api.proceedtopayment

api.createorder(cart, function(){
  api.proceedtopaymet(function(){
    api.showorder summary(function(){
    });
  })
}).
```

Payramid of Doom ↳ Its unreadble & mangble

IOC:- The callback function is passed to another call back, this way we lose the control of our code. We don't know what is happening behind the scene and the program becomes very difficult to maintain.

Call back Hell:- When a function is passed as an argument to another function it becomes a callback function. This process continues & there are many callbacks inside another callback functions.

Tech Suresi vedio:

```
Const datas = [
  {name: "Ajay", profession: "Software Eng"}
  {name: "Anuj", profession: "Software Eng"}
];
```

→ A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of action

```
function getDatas(){
  SetTimeout(()=>{
    let output = "";
    datas.forEach((data, index) =>{
    Output += `<li>${data.name}</li>`;
  })
  document.body.innerHTML = output
  }, 1000);
}
```

callback

```
function createdata (newdata){
  setTimeout (() => {
    datas.push (newdata); callback();
  }, 2000);
}.

create ({name: "vivek", profession: "software"
  , getDatas);
```

Promise:-

```
function createdata(newdata){
  return new promise((resolve, reject) =>{
    setTimeout (() =>{
    datas.push (newdata);
    let error = false;
    if (!error) {
          resolve();
    }else{
      reject ("Kuch sahi nhi")
    }
  getDatas }, 2000);
  })
}
```

```
Create(
            ).then().Catch(err => console.log
(err));
```

Async/Await:

```
async function start(){
  await createdata({name :"vivek",
 profession : "Software Engineer"});
} getData();     }
Start();
```

→ Traversy media video continuation Promises

```
Const posts =[{ title:'post one', body:'This is 1'}
{title:'post two', body:'This is 2'}]

function get posts (){
       SetTimeOut(() => {
       let output = '';
       posts.forEach((post, index) => {
      Output += '<li> ${post.title} </li>';
});
 document.body.innerHTML =output;
 },1000);
}
function CreatePost(post){
 return new Promise((resolve, reject) => {
  SetTimeOut(() => {
      posts.push(post);
      const error = false;
```

Task4

```
if (!error) {
   resolue();
}else{
   reject('Error something went wrong');
  }
 }, 2000);
});
}
CreatePost({ title : 'post Three', body: "This
post 3'})
   .then(getPosts);
```

→ if we want err write .catch (err =>
        console.log (err));

Scanned with CamScanner