Lecture Plan

Friday, July 22, 2022 6:13 PM

Understanding **Promises :**

What is a Promise

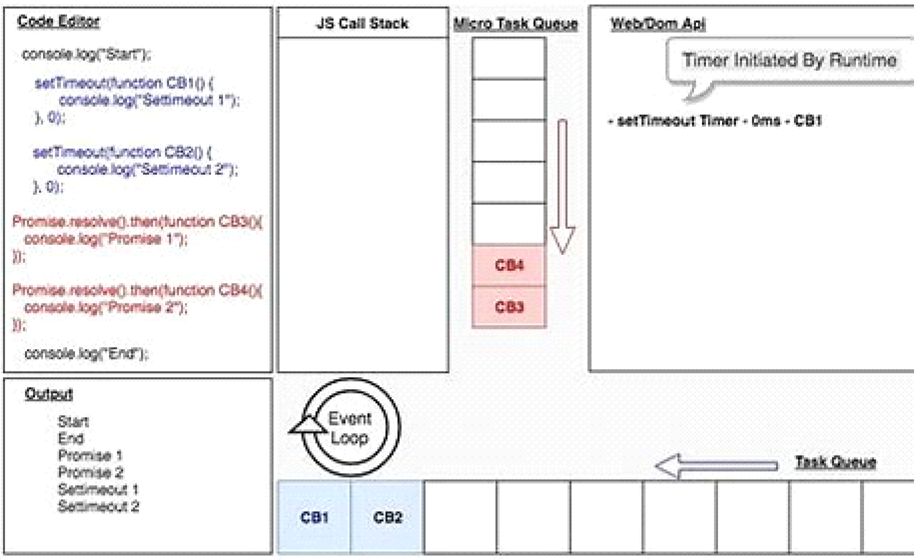
1.

A **Promise** is a proxy for a value not necessarily known when the promise is created.
It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.

A Promise is in one of these states:

- *pending*: initial state, neither fulfilled nor rejected.
- *fulfilled*: meaning that the operation was completed successfully.
  This is called as Promise RESOLVED
  When Promise is RESOLVED the "then" API is called
- *rejected*: meaning that the operation failed
  When PROMISE is REJECTED the "catch" API is called

1. Simple promise Demo - E:\Courses\node\iet\promise0.js
2. Callback queue Vs Promise ---- MicroTask Queue or Promise Queue has higher priority than callback queue
   E:\Courses\node\iet\promise1.js



Code Editor

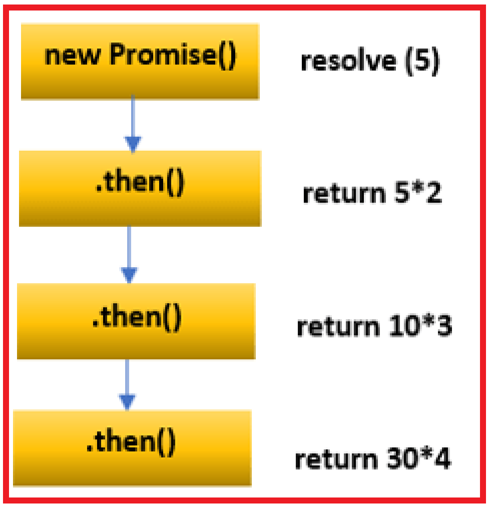
```
console.log("Start");

  setTimeout(function CB1() {
      console.log("Settimeout 1");
  }, 0);

  setTimeout(function CB2() {
      console.log("Settimeout 2");
  }, 0);

Promise.resolve().then(function CB3(){
  console.log("Promise 1");
});

Promise.resolve().then(function CB4(){
  console.log("Promise 2");
});

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

Output

```
Start
End
Promise 1
Promise 2
Settimeout 1
Settimeout 2
```

____

Chaining with "then " = E:\Courses\node\iet\promise3.js , E:\Courses\node\iet\promise4.js



new Promise().then().then().then() …….

RETURN VALUE of **then** is ANOTHER PROMISE !!!!

```
let  x = new Promise(….)
 let  y = x.then(…)
let z =  y.then(….)
         z.then()
```

x, y , z all three Promise objects

___

___

HTTP CALLS using REACT -------------------

___

Routing

Step 0 -- npm audit fix --force
Step 1 --- npm install react-router-dom =   cd to my-app folder and execute this command !!
This command will add the react-router and react-router-dom libraries in node_modules

ROUTING --------- on different URLs show different pages in a Single Page Application
----- All html partials --- react components are ALREADY with the browser
---- these parts will be shown when their **url is clicked**
We don't go to server on url click
We are on client , just the component that is clicked is shown in the outlet

The url to be clicked is called as **Route**

<Routes>
Route will MAP URL-path with the Component
</Routes>

Step 2 ----------define Routes in index.js
Step 3------------   define <nav > and <link > <Outlet>   in a React Component

If we pass a Route Parameter then the component can retrieve the route parameter using
useParams()   hook