Async Await Keyword

Tuesday, March 15, 2022 5:39 PM

- ES2015 makes it easier to handle a Promise by using async await keyword.
- await automatically waits for a promise to complete

- Assume you have function that returns a promise

```
function  airlinesSearch( name, param ) {
    return new Promise(  (resolve, reject ) =>{

        //write the promise logic here
        if( error)
            reject( new Error("invalid input"));
        else
            resolve( result );
    };
}
```

- We have another function that is calling the first first function

Handle Promise using then, catch

```
function printAirlineSearchResult(params){


    let  promise = airlinesSearch (" Indigo", params);

    Promise
        . then ( data => console.log(data) ) ; //on success
        .catch(error=>console.log(error.message));

}
```

Handling using async await keyword

```
async function printAirlineSearchResult(params){

    try{
        let data= await airlinesSearch("INDIGO",params);
        //on success
        console.log(data);
    }catch(error){

        console.log(error.message);
    }
}
```

What does await do?

- It waits for a promise to resolve or reject
- We don't get promise object we get the data but in future
- We directly print the data
- If promise is rejected it is thrown as exception and should be handled catch block

What does 'async' do?

- You must use 'await' keyword in an async function only.
- You can't use 'await' in non-async function or global area.
- Remember await must wait for promise to complete
- Async function automatically makes the function return a promise

```
function    sum( x, y ){


    return x+y;

}
```

This function returns a number

- number

```
async  function sum(x,y) {


    return x+y;
};
```

This function returns a Promise of Number

```
function  sum(x,y) {

    return new Promise(resolve=>{

        resolve(x+y); //return x+y
```

- An Async function always returns a promise

```
    });

}
```

Converting setTimeOut to a promise

- We want to convert set time out to a simple delay
- Then we can await for time out

```
function  delay(  time  ){
        return Promise( resolve => {

                setTimeOut( ()=>resolve(), time);

        }
}
```

Write a countdown promise that can count from max to 0 at 1 second interval

```
function  countDownPromise( max ){


}
```