**Assignment 5 Solutions**

**Q.1 What’s difference between Synchronous and Asynchronous?  
Ans. Synchronous** as the name suggests synchronous means to be in a sequence, i.e. every statement of the code gets executed one by one. So, basically a statement has to wait for the earlier statement to get executed.  
**Asynchronous** code execution allows to execution next instructions immediately and doesn't block the flow because of previous instructions.

**Q.2 What are Web Apis ?  
Ans. API** stands for Application Programming Interface. API is actually some kind of interface which is having a set of functions. These set of functions will allow programmers to acquire some specific features or the data of an application.  
Web API is an API as the name suggests, it can be accessed over the web using the HTTP protocol. It is a framework that helps you to create and develop HTTP based RESTFUL services.

**Q.3 Explain SetTimeOut and setInterval ?  
Ans.** The **setTimeout** Javascript method is used to call a function after a certain period of time. The time after which the function will be called is given by the user in milliseconds.  
**let timerID = setTimeout(func, delay)**The **setInterval** Javascript method is used to call a function repeatedly at a specified interval of time. This time interval at which the function will be called is provided by the user in milliseconds.  
**let intervalID = setInterval(func, delay)**setTimeout() is cancelled by clearTimeout() method, and setInterval() is cancelled by clearInterval() method.

**Q.4 How can you handle Async code in JavaScript ?  
Ans.** We can handle Async code in javascript either using callbacks or by using promises.  
console.log("Program Starts......");  
 setTimeout(() => {  
 console.log("setTimeout execution....");  
 }, 0);  
 new Promise((resolve, reject) => {  
 resolve("Promise resolved.....");  
 })  
 .then((res) => console.log(res))  
 .catch((error) => console.log(error));  
 console.log("Program Ends.....");  
Output:   
Program Starts......  
Program Ends.....  
Promise resolved.....  
setTimeout execution....

**Q.5 What are Callbacks & Callback Hell ?  
Ans.** Callbacks are functions that are passed as arguments to another function which will be executed later within the outer function.  
The phenomenon which happens when we nest multiple callbacks within a function is called a callback hell. The shape of the resulting code structure resembles a pyramid and hence callback hell is also called the “pyramid of the doom”. It makes the code very difficult to understand and maintain.

**Q.6 What are promises? What are the different states of a promise?   
Ans.** The **Promise** is an object which represents eventual completion or failure of an asynchronous operation. It allows us to associate handlers with the asynchronous operation’s eventual success value or failure reason.  
**States of a Promise: -   
- pending: -** it is the initial stage **- fulfilled: -** means operation is successfully completed **- rejected: -** means the operation is failed  
Example: -   
const myPromise = new Promise(function (resolve, reject) {

    const a = 20;

    const b = 20;

    if (a === b) {

        resolve();

    }

    else {

        reject();

    }

})

myPromise

    .then(function () {

        console.log("Promise resolved!");

    })

    .catch(function () {

        console.log("Promise rejected!!");

    })

Output –   
Promise resolved!  
We might need to create our own promise when we want to fetch data from an api, we don’t know if the api will always return us data and don’t know after how much time we will receive our data so here we can use promise.

**Q.7 What’s async & await Keyword in JavaScript  
Ans. Async/Await** is used for writing asynchronous javascript code. It is another way of writing promises.   
Async: It simply allows us to write promises-based code as if it was synchronous and it checks that we are not breaking the execution thread. It operates asynchronously via the event loop. Async functions will always return a value. It makes sure that a promise is returned and if it is not returned then JavaScript automatically wraps it in a promise which is resolved with its value.  
const getData = async() => {

let data = "Hello World";

return data;

}  
 getData().then(data => console.log(data));

Output:   
Hello World  
Await: Await function is used to wait for the promise. It could be used within the async block only. It makes the code wait until the promise returns a result. It only makes the async block wait.  
const getData = async() => {

let y = await "Hello World";

console.log(y);

}

console.log(1);

getData();

console.log(2);  
Output:  
1  
2  
Hello World

**Q.8 Explain Purpose of Try and Catch Block & Why do we need it?  
Ans.** The **try** statement allows you to define a block of code to be tested for errors while it is being executed.  
The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.  
When an error occurs, JavaScript will normally stop the execution of code and generate an error message. In this way we can handle errors in much better way.

**Q.9 Explain fetch.  
Ans.** The fetch() method in JavaScript is used to request data from a server. The fetch() method requires one parameter, the URL to request, and returns a promise.

**Q.10 How do you define an asynchronous function in JavaScript using async/await?  
Ans.** async function fetchMovies() {

const response = await fetch('/movies');

// waits until the request completes...

console.log(response);

}  
We can simply define a function using async await like above.