Assignment Questions 5

💡 **Q.1** What’s difference between Synchronous and Asynchronous?

Ans.: In JavaScript, synchronous means doing one thing at a time. A function gets executed in sequential manner i.e. when it's turn, and it waits for others to finish their turns before getting executed. Can cause delays.

Whereas, asynchronous means functions do not wait for their turn the get executed independently. It allows you to multitask and be more efficient. Asynchronous execution is particularly useful when dealing with time-consuming operations, such as network requests, file system operations, or database queries.

For example:

In following javascript code,

console.log("Task 1");

setTimeout(function() {

console.log("Task 2");

}, 2000);

console.log("Task 3");

The system will print task 1 and task 3 but it’ll print task 2 after it’s timeout.

💡 **Q.2** Explain Web Apis.

Ans.: A web API is an application programming interface (API) that allows interaction with a web server or a web browser. It is a way for two pieces of software to talk to each other. It's a way for them to share information and work together. Web APIs are often used to get data or do specific things on a website.

When a program or app wants to talk to a website, it sends a message called a request. The website then sends a message called a response back to the program or app. These messages are usually written in a special format that the computer can understand.

For example,

* The Twitter API can be used to access Twitter data, such as tweets, users, and hashtags.
* The Google Maps API can be used to embed Google Maps in your web pages.

💡 **Q.3** Explain SetTimeOut and setInterval

Ans.: setTimeout is a function in JavaScript that allows us to execute a function or lines of code after a specific time duration only once. Its like setting a timer for some time to perform a specific task.

The syntax of setTimeout function is as follows ;

setTimeout (function(){

},time in mili seconds);

Here in the above example the function will be executed once when the specified time is passed.

setInterval is a function in JavaScript that is used to perform an action reoeatedly for certain period of time.That means it allows us to run a specific piece of code for few seconds again and again.

The syntax of setInterval function is as follows ;

setInterval (function(){

},time in mili seconds);

Here in the above example the function will be executed again and again for specified time in ms.

💡 **Q.4** What are the ways we have to handle Async Code in JS?

Ans.: There are following ways to handle async code in js as:

1. Callback: Callbacks are functions that are passed as arguments to other functions. The callback function is invoked once the asynchronous task finishes, allowing us to handle the result or perform further actions.
2. Promises : It’s the efficient way to handle the async function. Promises are objects that represent the eventual completion (or failure) of an asynchronous operation. They can be chained together to create complex asynchronous workflows.
3. Async/await: Async/await is a newer syntax that makes it easier to write asynchronous code. It allows you to write code that looks synchronous, but it actually runs asynchronously.

💡 **Q.5** What are Callbacks & Explain Callback Hell ?

Ans.: Callback: Callbacks are functions that are passed as arguments to other functions. The callback function is invoked once the asynchronous task finishes, allowing us to handle the result or perform further actions.Callbacks are functions that are passed as arguments to other functions. They are executed when the asynchronous operation completes.

Callback hell is a situation that occurs when multiple callbacks are nested within each other, creating a long chain of functions that are executed one after the other. This can make the code difficult to read and understand, and it can also be difficult to debug.

💡 **Q.6** What are Promises & Explain Some Three Method of Promise

Ans.: Promises : It’s the efficient way to handle the async function. Promises are objects that represent the eventual completion (or failure) of an asynchronous operation. They can be chained together to create complex asynchronous workflows

Promises are used instead of using callback as there can be a situation called as callback hell.

Three methods of promises are,

all(): This method is used to wait for multiple promises to resolve before executing a callback function.

race(): This method is used to wait for the first promise to resolve or reject before executing a callback function.

finally(): This method is used to execute a callback function regardless of whether the promise resolves or rejects.

💡 **Q.7** What’s Async & Await Keyword in JavaScript

Ans.: The async and await keywords are a new feature of JavaScript that allows us to write asynchronous code that looks synchronous. This can make the code easier to read and understand, and it can also help to prevent callback hell.

💡 **Q.8** Explain Purpose of Try and Catch block

Ans.: A try-catch block is a programming construct that allows you to handle errors that occur in your code. The try block contains the code that you are trying to execute, and the catch block contains the code that you want to execute if an error occurs.

The try and catch block is a way to handle errors in our code in a controlled manner. It allows us to handle errors, display meaningful error messages, and take appropriate actions.

💡 **Q.9** Whats fetch in JavaScript?

Ans.: The fetch() method in JavaScript is used to make asynchronous HTTP requests. It is a newer way to make HTTP requests in JavaScript, and it is more elegant and efficient than the XMLHttpRequest object.

The fetch() method takes a URL as an argument, and it returns a promise. The promise resolves with the response from the HTTP request, or it rejects with an error.

💡 **Q.10** How do you define an asynchronous function in JavaScript using async/await?

Ans.: To define an asynchronous function using async/await in JavaScript, we use the keyword async as shown in following example:

async function fetchData() {

try {

const response = await fetch('https://api.example.com/data');

const data = await response.json();

console.log('Data:', data);

// Perform further actions with the data

return data; // Optional: Return a value as a resolved promise

} catch (error) {

console.log('An error occurred:', error);

throw error; // Optional: Throw the error to reject the promise

}

}

In above example, the fetchData() function is defined as an asynchronous function. This means that the function can make asynchronous calls, such as making HTTP requests. The await keyword is used to wait for the asynchronous call to resolve. Once the asynchronous call resolves, the await keyword returns the value of the asynchronous call.

The await keyword is used to wait for the promise to resolve before the return statement is executed. This means that the return statement will not be executed until the HTTP request has completed.

If any error occurs during the fetch or data parsing, the catch block is executed. The catch block logs the error to the console and then throws the error. This means that the error will be propagated to the caller of the fetchData() function.