* Features of Async Code

→ Clean & consize

→ Better error handeling

→ Easier debugging

→ Improved performance

* Promise

The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

A Promise is in one of these states:

→ pending: initial ~~state~~, neither fulfilled nor rejected.

→ fulfilled: when operation is completed sucessfully.

→ rejected: when operation is failed.

Promises provide two main methods to handle their states: `then()` and `catch()`. The `then()` method is used to handle a fulfilled promise and receives the resulting value as its arguement. The `catch()` method is used to handle a rejected promise and receives the reason for the rejection as its arguement.

The constructor syntax for a promise object is:

```
let promise1 = new Promise(function(resolve, reject)
{
        // executor
});
```

The function passed to new Promise is called executor. When it is created, the executor runs autometically.

* Async

The keyword 'async' before a function makes the function return a promise

Ex

```
   async function Hello(){
          return "xyz";
   }
```

Some scenarios when 'async' is used:

→ Network request

→ Timers

→ File I/O

* Await

'await' is a keyword in JS that is used to wait for a Promise to resolve or reject before executing the next line of code. It can only be inside 'async' function, which are functions that are marked as asynchronous using the 'async' keyword.

Ex

```
let value = await promise1;
```

] used inside async function

## * API

Application Programming Interface

An API is a way for different software applications to talk to each other. It defines the rules for how one application can interact with another application, allowing them to share data and functionality.

## * Fetch API

The Fetch API interface allows web browser to make HTTP requests to web servers.

GET – To receive data,

POST – To send data, etc

[explore more]

## * JSON

JavaScript Object Notation

→ It is a text format for storing and transporting data.

→ Here, data is in key value pair.

→ Data is separated by commas

→ Curly braces holds objects

→ Square brackets hold arrays

Ex.- {"name": "Rishabh", "car": null}

## * Closure

A closure is a feature of the JS that allows a function to access variable (through reference not by copy) and function from its out lexical scope, even after the outer function has returned.

In other words, a closure is a function that has access to variables in its own scope, as well as variables in the scope of the function that created it.