```
API
----------
- Application programming interface
- Mechanism that enables front-end and back-end to communicate each other
- or back-end to any other service
- Api must follow some set of rules

Two Types of APIs
---------------------
1) Public APIs:
        freely available, and we can directly use interface
2) Private Apis:
        Internal to a company or an organization
        we cant use it directly
        Some token autherization and authentication needed

Authentication and Authorization
-----------------------------------
Authentication:
- It is the process of verifying the identity of a user
- It is generally done using username and password

Authorization:
- process of determining , what an authenticated user is allowed to done
- something like, whether a logged user can access particluar page  or service
- authorization is mainly implemented uisng user roles

URL
----
Uniform resource location:
    adress of a location where specific resource is stored in internet/cloud

DNS
---------
Domain name server: used for domain name to ip address mapping

Protocols: Set of rules that need to be consider while sending and receiving data

Different Types of communication protocols
-------------------------------Ḥ
HTTP:  Hypertext Transfer Protocol (most commonly used)
FTP: File Transfer Protocol
HTTPS: HTTP secure (data is send and received in encrypted form, it uses secure
socket layer)

HTTP methods
----------------
GET: to get data form resource
POST: to send data to resource
DELETE: to delete data from rersource
```

PUT: to update data in resource

HTTP status codes

1XX - informational resposne
2XX - success
3XX - redirectional
4XX - client error (problem with request)
5XX - Server side error

JSON

- Javascript object notation
- commonly used data format to send data through internet(request/response)
- data is send as key value pairs
- here key must be in string format

```
        eg: {
                "name":"john",
                "age":27
            }
```

- simple format, make communication easy.
- both FE and BE can easily understand the data

XML

-Extinsible markup language
- more complex
- it uses tags

API architecture Types

1) REST api architecture
    - representational state Transfer
    - most popular architecture for creating an API
    - it uses set of rules for building api
    - it uses HTTP for basic CRUD operations

    CRUD: Create, Read, Update and Delete

2) SOAP

    Simple object access protocols

3) RPC

    Remote procedure call

API testing tools

1) Thunder client : vs code extension
2) Postman

Synchronous functions and Asynchronous functions

----
1) Synchronous functions
- Synchronous functions are executed in sequence
- Each operation must complete before next one begin
- if Synchronous function contains long-running task(like loop, complex calculations, api fetch),
it will block the execution of subsequent code, until the task is finished

2) Asynchronous functions
- it takes some time to execute
- it allow the program to continue executing other tasks while waiting for
asynchronous operation to complete

Is Javascript single threaded or Multi threaded

- Primarily Javascript is single threaded,
- it can execute only one operation at a time
- Callstack in Javascript engine is executing the code,
- Javascript consists of only single callstack

How Javascript handles asynchronous operations
----

1) callstack
2) microtask queue
3) callback queue
4) Event loop

****
during the execution phase of Javascript, js first adds all the
synchronous operations to callstack, calls stack is actually
executing the js code, when js finds any asynchronous functions
like timer methods(eg: setInterval, setTimeout) or Api calls,
file read or write, js taken away that asynchronous fns from
normal execution flow and add to Queue, then continue with the
next line,
There are 2 Queues
1) Microtask queue
2) callback queue
microtask queue hold api calls, callback queue holds callback
functions and timer functions
Microtask queue has high priority

Event loop

It controls all this operations, It is an infinite loop
that runs in the background, continously check, whether callstack
is empty or not, if empty, it first check the microtask queue, if any items
in microtask queue, event loop will add that item into callstack
when microtask queue becomes empty, eventloop check callback queue

```
Different Api Calling methods
------------------------------
1) Ajax
2) fetch
3) Async await

1) Ajax
------------
- Asynchronous javascript XML
- it make use of a inbuilt class in javascript : XMLHttpRequest
- 1) first we create an object of the above class to access methods and properties
of that class
      const http= new XMLHttpRequest()
- 2) call open() for connection establishment
        http.open('method','url')
- 3) send request
        http.send()

Ready State
-----------------
- Ready state are important concept while working on Ajax request
- It indicates the state of ajax request
- its value ranges from 0 to 4
- it allows user to track the progress of Ajax request
        0 : unsent state
        1 : open state
        2 : header set state
        3 : loading state
        4 : done state

2) fetch
        promise
        ---------
        - it make use of promise concept
        - promise is used to manage asynchronous functions/operations
        - promise represents an eventual completion of an asynchronous operation
        - it has three states
                1) pending - in progress
                2) resolve - fulfilled/done
                3) reject - failure state
        - We are calling asynchronous function inside promise,
        after completion of that asynchronous function, if it is success
        it uses resolve() method to send back the response, if it is failure,
        it uses reject() method to send back the error
        .then() method is used to used to access the completed promise function
        if promise rejected error, we can access that error in .catch() method

        const promise_name = new Promise((resolve, reject)=>{
        })

        promise_name.then(()=>{

        }).catch((error) => {
             console.log(error)
        })

      fetch('api url')
        .then((response)=>{
            console.log(response)
            // here resposne is in diffrent format, like streams, we have to
convert it into js object
            // so .json() method is used
            response.json().then((data)=>{

            })
        })

fetch post method
-------------------
fetch("api_url",{
        method:'POST',
        headers:{
                'Content-Type':'application/json'
                'Bearer Toke':'hgsdnvsndvnsdb'
        },
        body:{
                "key1":value,
                "key2":value
        }
})
.then((response)=>response.json())
.then((data)=>{
        console.log("success", data)
})
.catch((error)=>{
        console.log('Error', err)
})

3) Asyc await
---------------------

- We can store the result to a variable
- await must be called inside async function
syntax:
 const data = await
fetch(`https://restcountries.com/v3.1/name/${countryName}?fullText=true`);
            data.json().then((result) => {
                console.log(result)
```

```
})
```