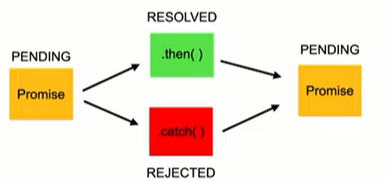
Aqui trataremos dos termos de assincronidade e APIs.

**Assincronicidade** aquilo que não ocorre ou não se efetiva ao mesmo tempo, e fazer as ações sem seguir uma sincronia como ter um turma indo estudar na escola na hora x (sincronia) ou ter cada aluno estudando em seu devido horario desejado (assincronia).

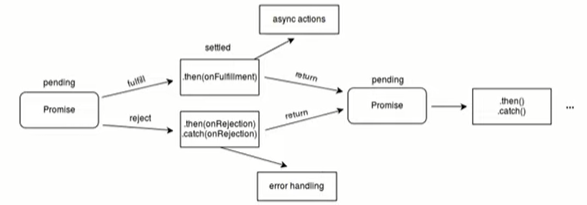
O javascript ele vai rodar por padrão em **sincronia**, seguindo **uma ordem padrão e fixa entre suas atividades**, a assincronicidade permite fazer instruções enquanto outra ocorre.

**Promisses** é um objeto de processamento assíncrono, de valor desconhecido, podendo ser resolvida ou rejeitada. Um dado que num momento que voce recebe ele voce não sabe o valor mas, quando este chegar, decidirá se vai trabalhar com ele ou não. Tem três estados:

**Pending** quando esta em aguardo de receber o valor.

**Fulfilled** quando ela é completada e faz sua instrução.

**Rejected** quando ela é rejeitada e **volta** a ser pendente e/ou retorna um erro.



o **then** visto ao lado é um subcomando que permite continuar a promise, neste exemplo a promisse vai esperar 2 segundos e mandar o primeiro then, esperar mais dois segundos e mandar o segundo then.

*Cast myPromise=new Promise((resolve,reject)=>{*

*window.setTimeout(()=>{*

*resolve(console.log(‘resolvida!’));}, 2000);});*

***await*** *myPromise*

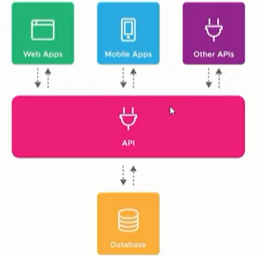
*.****then****((result)=> result + ‘ passando pelo then’)*

*.****then****((result)=>result + ‘e agora acabou!’)*

*.catch((err)=>console.log(err.message));*

Colocando a palavra **async** antes de uma função, torna a função assincrona, permitindo o uso das promisses e o comando **await**, o await serve para fazer o sistema **esperar a resolução da mypromise**, em outras palavras, **é o subcomando que da ouvidos à promise**.

**API**¸application programming interface, é uma forma de intermediar os resultados do back-end com o que é apresentado pelo front-end, sendo acessados por meio de URLs.

É comum vermos API no formato .json, o Javascript Object Notation, portando precisa-se tratar esses dados quando recebido ou enviado ressaltando que ele tem um formato de objeto, como o nome sugere.

*{“description”:”schema POST bank”,*

*“type”:”object”,*

*“id”:{“type”:”number”,”minimum”:0},*

*“code”:{“type”:”string”},*

*“name”:{“type”:”string”}}}*

O método **fetch permite o consumo de APIs**.

*Fetch(url,options)*  como pode ver pelo do then

*.then(response=>response.json())* o método fetch retorna uma

*.then(json=>console.log(json))* **promise**!

**