Código síncrono

Código síncrono explicado a bem grosso modo é basicamente o código que é executado linha por linha, vamos dar exemplo para ficar melhor o entendimento.

Temos abaixo um código **síncrono**:

```
const showAddress = (address) => {
    console.log(address)
}
let nowAddress = {
    street: 323,
    district: 'Brooklyn'
}
showAddress(nowAddress)

let nameUser = 'John Doe';
let yearsOld = 41;
```

Naturalmente o Javascript vai ler a primeira linha (nameUser), depois yearsOld e address, executado de maneira síncrona. Veja que eu defino uma função para exibir um endereço qualquer, defino uma váriavel com endereço, executo a função e depois crio mais duas váriaveis quaisquer. Quando eu chamo a função **showAddress** ela é executada completamente para só depois passar o controle à seguinte, definindo a váriavel nameUser e yearsOld.

Código **assíncrono**: Contexto: Temos que resgatar do nosso banco de dados informações do usuário e para isso sabemos que ao fazer uma request o servidor tem um determinado tempo para processar a requisição, o banco também tem um tempo para resgatar os dados e depois de tudo isso nos voltar o resultado da request (response).

```
const getUserById = id => {
    try {
        // vou tentar pegar meus dados no banco aqui
    }.catch(error) {
        // caso tenha algum erro, vou exibi-lo aqui para poder tratar disso.
    }
}

let welcomeMessage = 'Welcome to Javascript async/sync, '

// Suponde que fizemos uma req para pegar o nome do usuário mandando o ID dele (ID =
1).
Callbacks e sua solução, as Promises
const username = getUserById(1)

// Supondo que também a função ocorreu com sucesso e nos retornou o seguinte objeto:

{
    id: 1,
    username: 'John Doe',
    yearsOld: 41,
    address: {
```

```
        street: 323,
        district: 'Brooklyn'
    }
}

welcomeMessage += user.username
console.log(welcomeMessage)
```

Primeiro criamos a função de resgatar os dados do nosso database, depois criamos uma váriavel que terá um complemento depois, criamos uma váriavel que armazena o resultado da requisição feita e complementados a váriavel welcomeMessage no final.

Todos os comandos são executados linha por linha, porém, a nossa função faz uma comunicação externa com o database e não sabemos o quanto tempo ela irá demorar para nos trazer a resposta, nesse caso ela é uma função assíncrona. Se você entendeu um pouco do que os termos significam, **async await** não será problema para você.

Declarando uma função **assíncrona**:

```
async function nameFunction(){} ou const nameFunction = async () => {}

// A palavra async já é auto-explicativa, estamos dizendo que nossa função será assíncrona.

const loadArticlesBlog = async () => {
    const req = await fetch('https://gabrielvalin-blog.herokuapp.com/articles')
    const response = await req.json()
    console.log(response)
}

loadArticlesBlog()

// A palavra await vai dizer para nossa aplicação que para ela executar qualquer instrução abaixo, ela deverá esperar o término da execução da instrução atual para passar o controle para a instrução seguinte.
```

Um exemplo que vai te fazer entender bem isso é quando você entra em um site e aquela tela de "loading" fica rodando na sua tela, isso quer dizer que o site está processando/buscando/acessando informações para montar a tela que vai ser visualizada.