Fluxo de execução assíncrono em Javascript

Síncrono – é executado sequencialmente, sendo necessário esperar a execução linha a linha do código.

Assíncrono – é executado num fluxo diferente do principal, desta forma não há bloqueio executado na pilha principal.

* Callbacks – 2005
* Promises – ES6
* Async/Await – ES8

Por que o código assíncrono é tão importante no Front-end?

O javascript é executado em apenas um thread que também executa código de outros itens, o que varia de navegador para navegador.

Na fila onde o javascript é executado, o ‘painting’ e ‘rendering’ da página também acontece. Este é um grande problema, já que a página só será renderizada após a execução do código javascript.

Já pensou se fosse necessário esperar as requisições da sua API para que a página fosse mostrada?

É por isso que o código assíncrono é tão importante. Tarefas que demandam tempo, tais como requisições a recursos externos, carregamento de imagens, processamento massivo, entre outras devem ser executadas de forma assíncrona.

Toda essa brincadeira começou em 2005 com o famoso Ajax.

Isso proporcionou uma verdadeira revolução nas páginas web. A partir do Ajax, foi possível criar aplicações web interativas e complexas.

No ES6 tivemos a implementação nativa das promises, uma evolução das callbacks. As promises deixaram a leitura de fluxos assíncronos muito mais elegante.

No ES8 foi adicionado o que há de mais novo para escrever códigos assíncronos. Async/Await. Se a sintaxe já estava elegante com a Promises, com Async/Await ficou melhor ainda!

