Dia 20 – Espionando o manual do Javascript, falando sobre Async e Await

 Dia 20 - 'Folheie' a documentação da sua linguagem e faça um resumo de algum conceito novo

Salve jovem padawan, final de domingo, friozinho gostoso e com chuva, bem abençoada para minimizar essa seca assustadora, inspirados pela serie Round 6, fico pensando nas desigualdades econômicas, na crise do desemprego e endividamento das pessoas nesta louca vida consumistas, onde muitas pessoas do marketing criam necessidades não necessária e supérfluos comedores de dinheiro.

Mas tiozão o que isso tem haver com Javascript, manual e async await, tudo jovem padawan, pois neste artigo veras funções assíncronas, síncronas, single thread, call back hell, syntatic sugar e promise, senta-te confortavelmente que lá vem história.

Introdução

Um dos problemas da computação é que apesar de pensarmos que o computador seja multi-thread, na verdade não é, ou melhor, não completamente. Ele utiliza-se de artifícios para processar tudo ao mesmo tempo numa enorme fila, dando a impressão de fazer várias tarefas ao mesmo tempo. Aproveitando das limitações dos humanos para aperceberem-se disso, quase na velocidade da luz.

Nisso muitas linguagens usam do mesmo artificio, enganando nossos sentidos e fazendo pequenas interações. O Javascript criado com o objetivo WEB, nos idos anos 90 para dar vida a páginas HTML, não tinha a necessidade de múltiplas tarefas e o grau de complexidade para isso inviabilizaria a linguagem.

Mas a vida é froids e surgem problemas no paraíso, a necessidade de encher a linguagem de penduricarias, funções poderosas para vender e dominar o mercado, pressionadas pelo pessoal do marketing e executivos que aspiravam por bônus mais altos, entendeu agora a referência ao Round 6?

Obrigaram a incluir funcionalidades na linguagem que gerava o caos, dando lentidão aos sistemas, bloqueando servidores, derrubando websites e dificultando front-ends com os famoso call back hell.

Para minimizar situações assim, no passado foi criado o conceito de promise, que permitia paralelizar o processamento simulando multi-thread, avançando no processamento, enquanto algumas partes ainda estavam processando, se pensou em funções síncronas e assíncronas acertou está no caminho certo, mas se chegou agora na nossa comunidade e não faz ideia do que se trata, irei ajuda-lo.

O que é uma Função Síncrona?

Imagine um programa que faz um pedido e bloqueia o processo até receber a resposta, indo para o próximo pedido, parando e esperando, indefinidamente. Isso irrita e cria pontos de lentidão.

Imagine navegar na web, é cada vez que pede uma nova página o processo bloqueia e enquanto ela não for carregada, você perde o acesso ao navegador, isso é meio absurdo nos dias atuais, mas num passado não muito remoto era assim.

O que é Call-Back?

Numa chamada síncrona mentes habilidosas, conceberam uma estratégia para evitar paradas no processamento devido há uma chamada síncrona, embutiram trechos lentos em funções e com isso o processo prosseguia linearmente sem perda de performance.

Mas nem tudo foi flores nesse jardim, outras mentes não tão brilhantes assim, começaram a embutir call-back em call-back, num spaghetti code dos infernos e com isso surgiu o terror de todo programador de front-end o Call-back Hell.

O que é Call-back Hell?

Imagine um pesadelo de chamadas entrando e saindo e você não tem a menor ideia do que se passa? O CBHELL é ainda pior, um código sujo, confuso, macarrônico e destruidor de performance.

A parte triste que isso se tornou tao comum, que os rapazes da ECMA, resolveram criar uma ferramenta que evitasse isso, com isso vamos ao próximo ponto de nossa jornada.

O que é uma Função Assíncrona?

Foi uma jogada de mestre, dividindo-se a tarefa a mínima parte e no decorrer da atividade faz o pedido, segue, faz o próximo pedido, segue e conforme vai recebendo as respostas, volta ao ponto. A web é assim, fazemos um pedido de website e ele vai se completando aos poucos, primeiro carrega todo o texto, depois gradualmente áudios e vídeos

Fica a dica, entendeu por que quando colocamos scripts de javascript embutidos em páginas web, a boa pratica recomenda colocá-lo no final da página, sendo a última coisa a ser carregada, assim auxiliamos o processamento.

O que é promisse?

Trabalhando com funções assíncrona e evitando o CBHell surgiu o comando promisse, que ajudava a efetuar processos paralelos sem confusão, porém ele era tao difícil, que poucos usaram-no e tornou-se necessário alguns estratagemas para auxiliar os programadores.

Imagine a confusão, surge uma dor, apresentam uma solução para minimizar, mas um pouco confusa e nessa confusão, foi necessário um syntatic sugar para salvar a pátria: surgiu o assyn e o await.

Que cara é essa jovem padawan? Esta muito confuso? Onde te perdi?

O que é Syntatic Sugar?

Acredito que esse termo assustou um pouco e tirou a credibilidade do texto? Em Javascript e em outras linguagens, syntatic sugar é o nosso adoçante, foi uma estratégia para facilitar a codificação, uma forma de escrever, digo codificar de forma mais simples e racional.

Ajudando o programador a trabalhar de forma mais fácil, elegante e de fácil entendimento, evitando confusões com sintaxe difícil e fugindo do spaghetti code. Neste contexto surgiram o async e o await, comando promisse vitaminados e user-friends.

Respeitável publico com muito carinho apresento o Async e o Await.