**ECMAScript 6, uma breve introdução à POO**

Veja como ficou a programação orientada à objetos na nova especificação do JavaScript. Construiremos uma mini loja virtual usando classes com ECMAScript 6!.

O uso de classes na ECMAScript 6, porém, foram exemplos bem superficiais. Então decidi criar um exemplo mais prático de como utilizá-la.

**Entendendo o projeto**

A ideia inicial foi criar uma mini (e põe mini nisso) loja virtual, utilizando dados em JSON.

O que teremos, nada mais é que uma variável contendo dados em JSON e estes dados serão transformados em objetos… Advinha? Objetos de uma classe da ES6.

**Tá, mas pra quê?**

Bem, hoje em dia temos muitas ferramentas no mercado que utilizam dados em JSON o tempo todo para o front-end. O AngularJS, o backbone.js, e vários outros, são frameworks que utilizam de dados JSON parar gerar views.

**Estrutura dos arquivos**

Os arquivos do projeto seguem a seguinte estrutura:

**/**

\*\*    -/css\*\*

\*\*        bootstrap.min.css\*\*

\*\*    -/img\*\*

\*\*        01.jpg\*\*

\*\*        02.jpg\*\*

\*\*        03.jpg\*\*

\*\*    -/js\*\*

\*\*        script.js\*\*

**index.html**

Como você já deve ter percebido, na pasta css temos o bootstrap, na pasta img, temos algumas imagens, temos também o arquivo index.html na raiz do projeto, e por fim, na pasta js temos o arquivo script.js, que é onde acontece a mágica.

No final do artigo deixarei um link com o repositório deste projeto no github.

**Finalmente, mãos à obra**

Vamos começar pelo arquivo index.html, que tem uma estrutura bem simples, veja:

Bem simples não?

Vamos usar uma estrutura bem básica, pois a nossa mini loja só vai mostrar alguns produtos e pronto. Como eu já disse, o objetivo é usar a ES6.

No final teremos o seguinte resultado:

Bem, vamos estudar um pouco o HTML acima.

Nas linhas 6 a 10, apenas criamos um css para “riscar” o preço antigo no caso de promoção usando a classe .old-price.

Na linha 13 criamos o container e na linha 14 uma div com o atributo id setado como “lista”, é nesta div onde carregaremos os produtos.

**Preparado? Vamos ao JavaScript.**

O JavaScript será estudado mais a fundo, então não vou simplesmente colar o código e explicar as linhas como fiz com o HTML, vou fazer um passo à passo e no final mostro o resultado do arquivo.

**Qual o primeiro passo?**

Habilitar a ES6 no seu navegador, é claro!

Como a ES6 ainda não está funcionando totalmente nos navegadores e ainda não foi adotada como padrão, ela está por padrão desabilitada.

Neste exemplo utilizei o Google Chrome Canary, indico que você o utilize também, mas nada contra o firefox.

Para habilitar a ES6 no Chrome Canary, basta você abrir uma nova aba e acessar a url: chrome://flags

Após acessar esta url, você vai procurar algo parecido com a imagem abaixo, basta apenas habilitar e pronto:

Feito isso, podemos começar :).

O primeiro passo, é habilitar o strict mode do JavaScript, pois objetos não podem ser utilizados sem o modo strict. Você pode ver mais sobre o strict mode, sobre o uso de classes na ECMAScript 6, porém, foram exemplos bem superficiais. Então decidi criar um exemplo mais prático de como utilizá-la.

**Entendendo o projeto**

A ideia inicial foi criar uma mini (e põe mini nisso) loja virtual, utilizando dados em JSON.

O que teremos, nada mais é que uma variável contendo dados em JSON e estes dados serão transformados em objetos… Advinha? Objetos de uma classe da ES6.

mas pra quê?

Bem, hoje em dia temos muitas ferramentas no mercado que utilizam dados em JSON o tempo todo para o front-end. O AngularJS, o backbone.js, e vários outros, são frameworks que utilizam de dados JSON parar gerar views.

**Estrutura dos arquivos**

Os arquivos do projeto seguem a seguinte estrutura:

**/**

\*\*    -/css\*\*

\*\*        bootstrap.min.css\*\*

\*\*    -/img\*\*

\*\*        01.jpg\*\*

\*\*        02.jpg\*\*

\*\*        03.jpg\*\*

\*\*    -/js\*\*

\*\*        script.js\*\*

index.html

Como você já deve ter percebido, na pasta css temos o bootstrap, na pasta img, temos algumas imagens, temos também o arquivo index.html na raiz do projeto, e por fim, na pasta js temos o arquivo script.js, que é onde acontece a mágica.

No final do artigo deixarei um link com o repositório deste projeto no github.

**Finalmente, mãos à obra**

Vamos começar pelo arquivo index.html, que tem uma estrutura bem simples, veja:

Bem simples não?

Vamos usar uma estrutura bem básica, pois a nossa mini loja só vai mostrar alguns produtos e pronto. Como eu já disse, o objetivo é usar a ES6.

No final teremos o seguinte resultado:

Bem, vamos estudar um pouco o HTML acima.

Nas linhas 6 a 10, apenas criamos um css para “riscar” o preço antigo no caso de promoção usando a classe .old-price.

Na linha 13 criamos o container e na linha 14 uma div com o atributo id setado como “lista”, é nesta div onde carregaremos os produtos.

**Preparado? Vamos ao JavaScript.**

O JavaScript será estudado mais a fundo, então não vou simplesmente colar o código e explicar as linhas como fiz com o HTML, vou fazer um passo à passo e no final mostro o resultado do arquivo.

Qual o primeiro passo?

Habilitar a ES6 no seu navegador, é claro!

Como a ES6 ainda não está funcionando totalmente nos navegadores e ainda não foi adotada como padrão, ela está por padrão desabilitada.

Neste exemplo utilizei o Google Chrome Canary, indico que você o utilize também, mas nada contra o firefox.

Para habilitar a ES6 no Chrome Canary, basta você abrir uma nova aba e acessar a url: chrome://flags

Após acessar esta url, você vai procurar algo parecido com a imagem abaixo, basta apenas habilitar e pronto:

Feito isso, podemos começar :).

O primeiro passo, é habilitar o strict mode do JavaScript, pois objetos não podem ser utilizados sem o modo strict. Você pode ver mais sobre o strict mode][2](https://tableless.com.br/javascript-strict-mode/) do Fabiano de Lima Abreu.

Para fazer isto é simples, vamos inserir na primeira linha do arquivo o seguinte:

Simples, não?

Agora vamos criar a nossa classe Produto e seu método construtor, veja abaixo:

Não precisa ser um gênio para entender esta parte, e se você chegou a ler o artigo que deixei no início deste, já estará familiarizado com o assunto.

Criamos uma classe com seu método construtor, que possui os atributos, código, nome, imagem, promoção, preço e desconto.

Bacana, e agora?

Vamos criar um método para listagem de produtos, que receberá uma variável em JSON e organizará os produto no HTML.

Criaremos o método lista, que receberá como parâmetro uma variável contendo o JSON:

Em seguida criaremos uma variável lista pegando a div onde a lista será inserida pelo id:

Agora, vamos criar um laço de repetição que percorrerá nossa lista em JSON e criará seus elementos:

Dentro deste laço, criaremos e preencheremos os elementos HTML de cada produto, utilizaremos os thumbnails do bootstrap, como você pôde ver na foto com o resultado final.

**A estrutura básica será a seguinte:**

Para cada produto lido através do JSON, teremos esta estrutura para exibi-lo.

Primeiramente vamos instanciar a classe produto a cada repetição deste laço, utilizando o método construtor para atribuir às suas propriedades:

Quando criamos um objeto à partir de uma classe, é necessário a utilização da palavra reservada let.

Em seguida, vamos criar os elementos HTML dos produtos, começando pelas variáveis:

Em seguida, atribuímos o HTML nessas variáveis:

Em seguida, vamos adicionar as classes aos elementos para receberem a formatação do bootstrap:

Para adiantar, já colocamos o innerHTML do botão com o texto “Comprar”.

Agora o que temos que fazer é inserir nos elementos os seus valores:

Eu coloquei por enquanto, apenas o nome do produto, e sua imagem. Pois precisamos pensar na promoção.

O atributo promoção, é um valor booleano, que nos mostrará se o produto vai ou não ter desconto.

Caso promoção seja verdadeiro, colocaremos o preço normal do produto com a classe .old-price que criamos lá em cima para que ele fique “riscado”, e o preço real será um cálculo do valor do produto subtraído do desconto. Caso ocorra o contrário, o elemento old\_price receberá um display:none e o preço não sofrerá alteração:

E por fim, para finalizarmos o nosso laço de repetição, iremos dar um appendChild() onde for necessário, criando a hierarquia do HTML:

Não vou postar o código final do nosso método, pois o artigo já está bem grande. Basta dar uma olhadinha no repositório no final do artigo e você verá a estrutura completa.

Beleza, mas e agora?

Bom, criei uma variável após o final de nossa classe Produto, com os dados em formato JSON para testarmos os produtos. Em um sistema real, estes dados viriam de fontes externas.

Abaixo o código:

Por fim, basta apenas instanciar a classe Produto e chamarmos o método lista, passando como parâmetro nossa variável contendo os dados em JSON:

**Conclusão**

Este foi só um exemplo do que podemos fazer com as novas features da ECMAScript 6 e as mudanças (pra melhor) que ela nos proporciona.