JavaScript Avançado III: ES6, orientação a objetos e padrões de projetos

- Esse curso capacita a:
  - Aprender a armazenar dados off-line com IndexedDB;
  - Encantar-se com a Fetch API;
  - Usar Babel e garantir compatibilidade máxima do código;
  - Usar e entender as vantagens do ECMASCRIPT 2015 (ES2015) modules;
  - Tornar o código ainda mais elegante com novos padrões de projeto;

Aulas:

1. **Guardando negociações off-line com IndexedDB:**
   - **IndexedDB**: banco de dados presente dentro do navegador;
   - Quando o banco é a aberto pela primeira vez, três eventos são disparados, onupgradeneeded(sempre chamado quando o banco é criado pela primeira vez, mas nas vezes seguintes, não mais), onsucess e onerror;

```
let transaction = connection.transaction(['negociacoes'],'readwrite');

let store = transaction.objectStore('negociacoes');

let negociacao = new Negociacao(new Date(), 200, 1);

let request = store.add(negociacao);

request.onsuccess = e => {
  alert('Adicionado com sucesso!');
};

request.onerror = e => {
  alert('Não foi possível adicionar');
};
```

   - A partir da connection, obtemos uma transação através do método **transaction**. Ele recebe como primeiro parâmetro um array com a object store que desejamos criar uma transação, e como segundo o tipo de acesso à store. No caso, ter acesso de leitura e escrita. Com a transação, tem-se acesso a uma store transacional, através do método objectStore, que recebe como parâmetro o nome da store. É através da store que podemos realizar operações de persistência, como inclusão ou listagem;

     Chamar simplesmente **store.add** pode ou não adicionar efetivamente um objeto dentro de uma store, mas sempre ficaremos na dúvida se a operação foi realizada com sucesso. É por isso que o método **add** retorna uma requisição de abertura e no callback passado para seu evento onsuccess, quando ele for chamado, temos certeza de que o objeto foi adicionado. Caso um erro aconteça, o callback passado para onerror será chamado;

2. **Gerenciando nossa conexão com o pattern Factory:**

3. **Padronizando acesso aos dados com o pattern DAO:**
    - **Padrão DAO**: Apresenta a vantagem de que tem a capacidade de isolar todo o código que acessa seu repositório de dados em um único lugar. Assim, toda vez que o desenvolvedor precisar realizar operações de persistência ele verá que existe um único local para isso, seus DAO's;

4. **Lapidando um pouco mais nossa aplicação:**
    - A função **some** itera sobre o array, assim como forEach, filter e map. No entanto, seu retorno é true ou false. Ela retorna true logo assim que encontrar o primeiro elemento que for condizente com o critério de comparação utilizado. Quando dizemos, "logo assim", significa que a função parará de iterar nos elementos da lista, porque já encontrou pelo menos algum (some) que atenda ao critério;

5. **Simplificando requisições Ajax com a Fetch API:**

6. **Tornando nosso código ainda mais compatível usando Babel:**
    - Babel é um transcompilador muito famoso no cenário open source;

7. **Trabalhando com módulos do ES2015!**