01

Introdução ao AJAX

Transcrição

Continuando a construção da página da Aparecida Nutricionista, a última *feature* que vamos desenvolver é a capacidade de integração do site a um sistema externo, que também contém pacientes e está em outro computador. A Aparecida quer trazer esses pacientes para que eles fiquem cadastrados no mesmo sistema.

O sistema externo de pacientes pode ser acessado no navegador pelo endereço [api-pacientes.herokuapp.com/pacientes (https://api-pacientes.herokuapp.com/pacientes)](https://api-pacientes.herokuapp.com/pacientes). Nele, há uma lista de pacientes que devem ser integrados ao sistema da Aparecida.

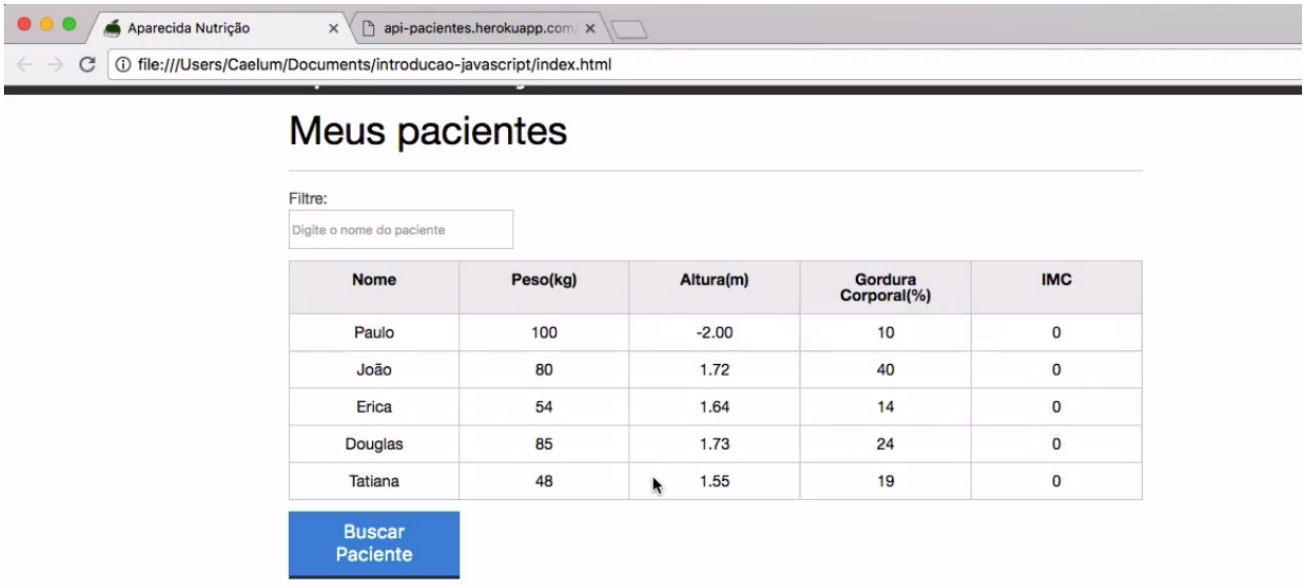
```
[
    {
        "nome": "Jéssica",
        "peso": 47,
        "altura": 1.54,
        "gordura": 17,
        "imc": 19.82

    },
    {
        "nome": "Flavio",
        "peso": 70,
        "altura": 1.7,
        "gordura": 17,
        "imc": 20.76
    }
//...
```

Nós queremos que esses pacientes sejam importados com o clique de um botão para a tabela do site de modo que, quando o usuário clicar, os dados dos pacientes serão pesquisados no sistema externo e depois adicionados ao site da Aparecida.

A primeira coisa que devemos fazer é adicionar o botão na página `index.html`. É nele que clicaremos para buscar os pacientes externos. Adicionaremos o botão abaixo do fechamento da tag `<table>`:

```
//...
   <button id="buscar-pacientes" class="botao bto-principal">Buscar Pacientes</button>

</section>
```

No browser já veremos o novo botão, que por enquanto ainda não funcionará:



Precisaremos criar uma nova *feature*, e para isto vamos gerar um novo arquivo JavaScript na pasta `js`, que receberá o nome de `buscar-pacientes.js`. Importaremos este novo arquivo à página `index.html`:

```
<script src="js/calcula-imc.js"></script>
<script src="js/form.js"></script>
<script src="js/remover-paciente.js"></script>
<script src="js/filtra.js"></script>
<script src="js/buscar-pacientes.js"></script>
```

Se queremos procurar os pacientes ao clicarmos no botão de "Buscar Paciente", devemos selecioná-lo e atrelá-lo ao evento de `click`. Começaremos adicionando a variável `botaoAdicionar`, no arquivo `buscar-pacientes.js`:

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function(){
    console.log("Buscando pacientes");

});
```

Queremos replicar o passo que fizemos com o navegador, em que sabemos como acessar um endereço que pode ser aberto em uma nova aba, digitá-lo e pressionar na tecla "Enter". Ou seja, o navegador é o responsável pela requisição, indo até a URL e nos mostrando os dados. Mas como faremos isso dentro da nossa página?

Veremos como fazer a requisição com JavaScript de modo que ela vá até a URL, busque e retorne os dados, sem perder os que já estão no site da Aparecida Nutricionista. Desta forma, não dependeremos do navegador, e a nossa página vai recarregar.

No código JavaScript, devemos acessar o endereço api-pacientes.herokuapp.com/pacientes (https://api-pacientes.herokuapp.com/pacientes), buscar e trazer os seus dados e colocá-los na tabela. Esse endereço é uma API, uma interface de programação que disponibiliza os dados para o usuário.

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function(){
    console.log("Buscando pacientes...");
    https://api-pacientes.herokuapp.com/pacientes
});
```

Não sabemos de onde vêm os dados, como e de que maneira eles foram gerados, pois eles se encontram disponibilizados e prontos para uso, sendo esta uma característica de uma API.

Ao analisarmos os dados, eles possuem uma estrutura que já conhecemos: cada paciente fica entre chaves, dentro das quais há itens formados por um par contendo chave e valor.

Além disso, todos os dados estão armazenados entre colchetes (`[]`), característica de um array do JavaScript. Como os dados estão disponibilizados em uma estrutura que conhecemos, será bem fácil trazê-los para dentro do código JavaScript, mas teremos que encontrar uma forma de acessá-los. Ele está com uma notação bastante parecida com o JavaScript, para realizarmos a requisição sem o navegador, somente com a linguagem.

Para fazermos essa requisição, temos um objeto bastante conhecido no JS, o `XMLHttpRequest` :

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function() {
    console.log("Buscando pacientes...");
    var xhr = new XMLHttpRequest();
});
```

O `XMLHttpRequest` é um objeto do JS responsável por fazer requisições HTTP. O trecho `XML` do nome indica que ele era utilizado anteriormente para realizar o transporte de dados do tipo XML, no entanto, atualmente ele consegue trafegar outros tipos de dados, como textos.

Para que ele realize as requisições, devemos ensiná-lo e configurá-lo do jeito que queremos. Por exemplo, informaremos que uma requisição será feita para o seguinte endereço: `https://api-pacientes.herokuapp.com/pacientes` , com alguns de seus métodos.

O primeiro será `open()` , com o qual especificaremos o tipo de requisição a ser feita, no caso, `GET` . Também indicaremos para onde queremos fazê-la:

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function(){
    console.log("Buscando pacientes...");

    var xhr = new XMLHttpRequest();

    xhr.open("GET", "https://api-pacientes.herokuapp.com/pacientes");
});
```

Essa ação será equivalente a chegarmos no navegador no momento em que ainda não enviamos a requisição, apenas verificando se o endereço está correto, se existe e está fazendo as configurações da requisição. Para que ela seja realizada, precisaremos chamar o método `send()` :

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function(){
    var xhr = new XMLHttpRequest();

    xhr.open("GET", "https://api-pacientes.herokuapp.com/pacientes");

    xhr.send();
});
```

Podemos testar o nosso botão somente com esse código! Porém, ao clicarmos no botão, nada acontecerá. Por quê? Com o código atual, é como se o JavaScript estivesse abrindo uma nova aba no navegador, em que digitamos o endereço e clicamos em "Enter". Ficou faltando a parte final, de exibição dos dados para o usuário.

Obtendo e exibindo a resposta da requisição

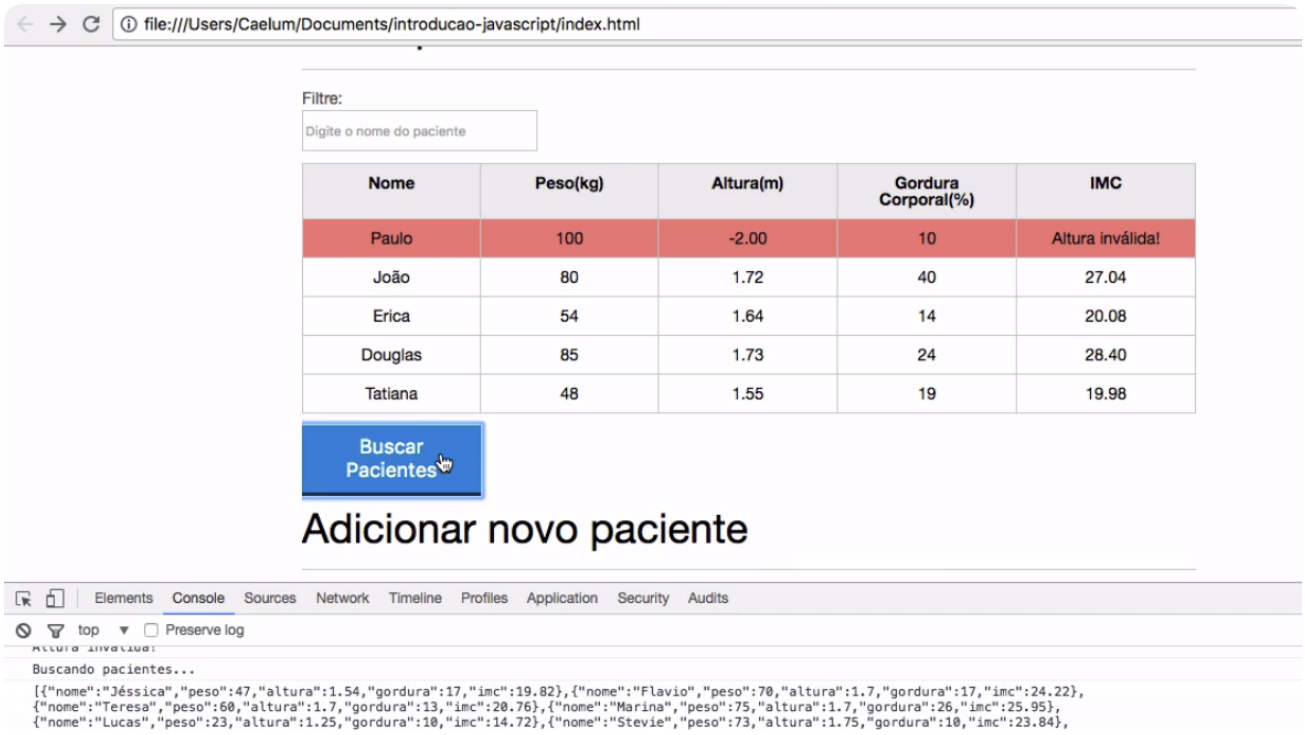
Para os dados serem exibidos, após o envio da requisição, devemos escutar um evento específico que é acionado quando a requisição termina e a sua resposta é carregada. Ao escutarmos o evento, carregaremos a resposta da requisição - que no caso, serão nossos dados. Esse evento é o `load` , característico do `XMLHttpRequest` :

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function() {
    var xhr = new XMLHttpRequest();

    xhr.open("GET", "https://api-pacientes.herokuapp.com/pacientes");

    xhr.addEventListener("load", function(){

    });

    xhr.send();
});
```

E para acessarmos os dados da resposta, usaremos a propriedade `responseText` do `XMLHttpRequest` . Para testarmos, podemos guardá-la em uma variável, e depois imprimi-la no console do navegador:

```
var botaoAdicionar = document.querySelector("#buscar-pacientes");

botaoAdicionar.addEventListener("click", function() {
    var xhr = new XMLHttpRequest();

    xhr.open("GET", "https://api-pacientes.herokuapp.com/pacientes");

    xhr.addEventListener("load", function() {
        console.log(xhr.responseText);
    });

    xhr.send();
});
```

Agora, o `xhr.responseText` será exibido no console:



Ao clicarmos no botão, os dados serão impressos no console - nós conseguimos trazer os dados de outro site para o "Aparecida Nutricionista"! O próximo passo será exibi-los na nossa tabela, criando os pacientes de acordo com os dados. Faremos isso a seguir.