08

Ajax e jQuery

O que é jQuery?

O jQuery (https://jquery.com/) é uma biblioteca bem leve que também é escrita em JavaScript, assim como o Google Charts.

O foco do jQuery é oferecer funções prontas para uso, simplificando a programação e o número de linhas de código em JavaScript.

Então, ao invés de escrevemos uma função para trocar a cor de fundo de uma página em JavaScript como

```
<script>
function mudarCorDeFundo(cor) {
        document.body.style.background = cor;
}
</script>
<body onload="mudarCorDeFundo('red');">
```

ao invés de usarmos 3 linhas, fazemos o mesmo com jQuery usando apenas uma linha de código:

```
<script>
$("body").css("background", "red");
</script>
```

Essa é uma das maiores vantagens dessa biblioteca.

O que é o AJAX do jQuery?

AJAX é um acrônimo de *Asynchronous JavaScript And XML*, ou, em português, JavaScript e XML Assíncronos.

O AJAX é uma junção feita com JavaScript de algumas tecnologias da *web* para fazer com que as páginas fossem recarregadas em partes, sem a necessidade de atualizar a página inteira. Isso acontece pois o AJAX permite que as partes das páginas se atualizem sem travar o navegador, ou de modo assíncrono, ao trocar informações com um servidor.

De acordo com *Jesse James Garrett*, que foi o criador do termo AJAX, temos 6 tecnologias unidas pelo JavaScript (e, se contarmos com ele, 7); são elas o HTML (ou XHTML) e CSS para mostrar adequadamente os dados na página, o Modelo de Documento de Objetos (DOM) para gerar interações dinâmicas com os dados, mais o JSON ou XML para a troca ou transporte de dados, junto com XSLT para a manipulação dos dados, e o objeto XMLHttpRequest para possibilitar a comunicação assíncrona.

Usando o AJAX dentro da biblioteca do *jQuery* como fizemos no curso, podemos acessar arquivos e passar essas informações sem necessitar de uma atualização da página inteira na qual os nossos gráficos são exibidos.

Ao fazermos isso, seguimos a documentação do *Google Charts* e passamos apenas 3 parâmetros, mas, se olharmos na documentação oficial do AJAX (https://api.jquery.com/jQuery.ajax/), percebemos que podemos especificar em torno de 34 parâmetros ao realizarmos essa comunicação.

Exemplo do uso de outros parâmetros

Por estamos tralhando com um *json*, em alguns navegadores podemos encontrar um erro ao exibir os nossos gráficos. Este erro é o `XML Parsing Error: not well-formed.` . Ele acontece porque o nosso navegador não entende que é um arquivo do tipo *json* e está esperando um *xml*, então ele acha que tem algum erro no arquivo.

Se olharmos na documentação, veremos que podemos usar um parâmetro para isso chamado tipo de conteúdo ou **contentType**, e especificar que esse conteúdo é um *json* com *"application/json"*, logo a nossa chamada ficaria assim:

```
var dadosJson = $.ajax({
    url: 'dados.json',
    contentType: 'application-json',
    dataType: 'json',
    async: false,
    }).responseText;
```

Mas ao fazermos isso, não só continuamos com o mesmo erro como temos ainda outro de pré-vôo ou em inglês *preflight* e o nosso gráfico nem é exibido. O que aconteceu?

O que acontece é que o navegador não entende esse tipo de conteúdo, ele entende apenas *application/x-www-form-urlencoded*, *multipart/form-data*, ou *text/plain*, e quando tentamos mandar um *application/json*, ele não deixa a página "decolar", ela fica em pré-vôo.

Então vamos tentar outra propriedade, uma que customize o tipo de arquivo que estamos mandando. Precisamos dizer para o navegador aceitar este arquivo novo. Logo, vamos escrever o que ele aceita com **accepts** e então passar o *application/json* como um tipo customizado ou **mycustomtype**.

```
var dadosJson = $.ajax({
    url: 'dados.json',
    accepts: {
            mycustomtype: 'application/json'
      },
    dataType: 'json',
    async: false,
    }).responseText;
```

Beleza, desse modo, demos um jeito de incluir um *json* aí, será que funciona agora?

Ainda não, continuamos com o mesmo erro, o nosso navegador ainda não entende o arquivo. Mas ainda temos uma outra propriedade para testar.

Quando eu descrevi o AJAX, disse que uma das suas partes era o objeto XMLHttpRequest. É nesse objeto que são definidos os tipos de arquivos que falamos até agora. Outros formatos além dos "tradicionais" são extendidos por algo que chama MIME (de Multipurpose Internet Mail Extensions em inglês ou Extensões Multiproposta de Email de Internet). O MIME foi criado para email, mas é usado nas nossas chamadas e protocolos da *web*.

É inclusive dentro do MIME que temos o *contentType* que acabamos de tentar explicitar. Mas, se isso não funcionou, o que iremos fazer é dar uma novo valor para esse MIME, que será o de um *json*.

Então vamos dizer que o tipo do MIME ou **mimeType** é o de *application/json*.

E agora, iremos conseguir?

```
var dadosJson = $.ajax({
    url: 'dados.json',
    mimeType: 'application-json',
    dataType: 'json',
    async: false,
    }).responseText;
```

E veja só, o erro sumiu. Ao sobrescrevermos os tipos suportados pelo protocolo, conseguimos dizer para ele qual o tipo do nosso arquivo corretamente.

Repare que para fazermos isso, fomos usando diversas propriedades existentes no AJAX do jQuery.

Cursos da alura

Caso queira saber mais sobre o jQuery, a Alura possui este curso (https://cursos.alura.com.br/course/jquery-a-biblioteca-do-mercado) sobre a biblioteca que fica dentro da carreira de desenvolvedor Front-end.