alura

Requisições Ajax

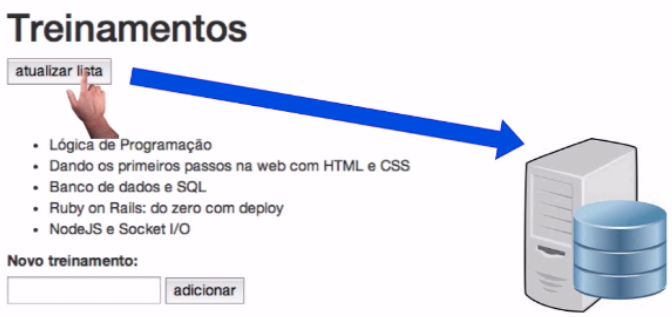
Revisão

Bem-vindo ao treinamento introdução ao jQuery da Alura. No capítulo anterior, vimos como utilizar o jQuery para alterar propriedades CSS, inclusive como animar a exclusão do item de uma lista.

Agora, a listagem será atualizável através de um botão. O problema é que se submetermos a página, perderemos os itens que acabamos de incluir.

O que é AJAX?

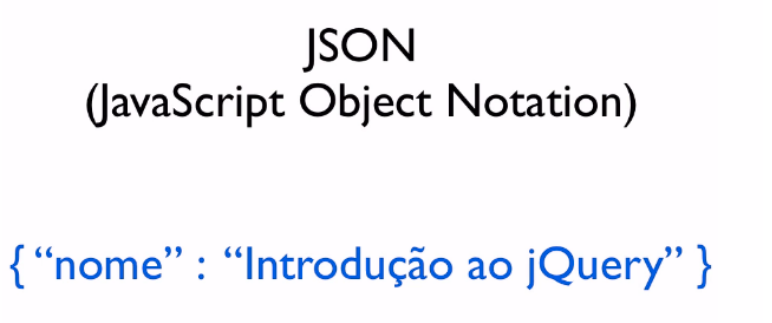
Podemos resolver este problema submetendo uma requisição AJAX, tema deste vídeo. AJAX significa Asynchronous Javascript and XML e nada mais é do que uma técnica que nos permite enviar requisições assíncronas, ou seja, manter a página que estava aberta intacta, e recuperar a resposta dessa requisição para fazermos qualquer processamento com ela.



JSON

Essas respostas costumam ser XML, HTML ou um formato de transferência de dados chamado JSON (::Javascript Object Notation::).

JSON (JavaScript Object Notation) é um formato de troca de dados. A sua estrutura facilita sua manipulação e criação tanto para seres humanos quanto para máquinas. Há ganho de performance em seu processamento se comparado com o XML. Este formato nada mais é do que um objeto JavaScript, com a diferença de seus atributos virem sempre entre aspas:



A função $.ajax()

Para realizarmos uma requisição AJAX, precisamos utilizar Javascript. E mais uma vez o jQuery irá nos ajudar.

Vamos começar adicionando um evento clique ao botão responsável pela atualização. Como sempre, passamos a função que queremos executar quando o evento for disparado.

```
1. $('#botao-atualiza').click(function() {
2. });
```

Agora, dentro da função, vamos executar uma requisição ajax. Isso é feito chamando a função de mesmo nome diretamente do jQuery. Esta função necessita de alguns parâmetros para funcionar. Eles são passados através de um objeto do JavaScript, onde cada propriedade é um parâmetro de configuração.

```
1. $('#botao-atualiza').click(function() {
2.     $.ajax({ url : ? ,
3.                dataType : ?,
4.               success: ?
5.     }});
6. });
```

Vamos começar pela propriedade url. Nela indicamos o endereço do serviço na web de onde queremos obter nossos dados. A segunda propriedade indica o tipo de dados que receberemos, em nosso caso, um JSON. Por fim, temos a propriedade success. Nela definimos uma função que recebe um parâmetro.

A ideia é a seguinte: caso a requisição Ajax seja realizada com sucesso, o jQuery chamará a função definida em success, inclusive nos passará através do parâmetro da função os dados que foram trazidos. Neste caso, receberemos uma lista com objetos que representam treinamentos e cada treinamento possui um nome.

```
1. $('#botao-atualiza').click(function() {
2.     $.ajax({
3.         url : 'http://mirrorfashion.caelum.com.br/treinamentos',
4.         dataType: 'json',
5.         success: function(retorno) {
```

```
6.
7.      }
8.      });
9.  });
```

Precisaremos varrer a lista, por exemplo, utilizando um laço for, mas podemos fazer a mesma coisa com jQuery.

```
1.  $('#botao-atualiza').click(function() {
2.      $.ajax({
3.          url : 'http://mirrorfashion.caelum.com.br/treinamentos',
4.          dataType: 'json',
5.          success: function(retorno) {
6.                $.each(retorno.treinamentos, function() {
7.                      var treinamento = this;
8.                      alert(treinamento.nome);
9.            });
10.     }
11.     });
12. });
```

O jQuery possui a função each(). Ela recebe dois parâmetros: o primeiro, a lista que queremos varrer, o segundo, uma função que nos dará acesso ao elemento de cada iteração através de this. Para deixar mais claro o código, podemos guardar this em uma variável com o nome treinamento e logo em seguida imprimir através de um alerta seu nome.

Abrindo e testando no Chrome, nada acontece. O Chrome possui um depurador de erros que para ser exibido basta apertar simultaneamente as teclas CONTROL + SHIFT + C ou F12. Dentro dele, há uma aba chamada console. Nela, vemos uma mensagem de erro do Chrome.

O problema é que o Chrome não aceita realizar requisições Ajax localmente. Poderíamos até configurá-lo, mas podemos evitar isso testando nosso código no Firefox. Caso você não tenha o Firefox instalado, esta é uma boa hora para tê-lo.

Mesmo com o Firefox nada acontece, e pior, não recebemos nenhuma mensagem de erro.

Isso acontece porque o Ajax, por padrão, só pode realizar requisições para o mesmo domínio da página que carregamos. Como estamos localmente e nosso serviço em outro lugar na Web, não conseguiremos testar nosso código.

Apesar disso, podemos trocar o dataType para "jsonp", um formato mais seguro suportado por outros servidores e que permite requisições para um domínio diferente do qual a página foi carregada.

Alteramos isso facilmente na função Ajax:

```
$('#botao-atualiza').click(function() {
    $.ajax({
        url : 'http://mirrorfashion.caelum.com.br/treinamentos',
        dataType: 'jsonp',
        success: function(retorno) {
              $.each(retorno.treinamentos, function() {
                    var treinamento = this;
                    alert(treinamento.nome);
          });
    }
    });
});
```

Pronto, testando mais uma vez, vemos que nosso alerta exibe os dados que recebemos. Isso ainda não é suficiente. Precisaremos criar dinamicamente um item da lista com o nome do treinamento. Já aprendemos isso em exercícios anteriores. A solução envolve a função $ para criar elementos, a CSS para alterar propriedades e a Text, para alterar seu texto:

```
$('#botao-atualiza').click(function() {
    $.ajax({
        url : 'http://mirrorfashion.caelum.com.br/treinamentos',
        dataType: 'jsonp',
        success: function(retorno) {
              $.each(retorno.treinamentos, function() {
                    var treinamento = this;
                   $('<li>').css('color', 'green').text(treinamento.nome).appe
          });
    }
    });
});
```

Por fim, adicionamos o item em nossa lista. Verificando o resultado, tudo sai conforme o esperado.

Ver video | Começar exercícios