02

Salvando placar

Transcrição

Se o nosso objetivo é enviar o placar via `POST` para o servidor, para que ele sobreescreva o placar antigo e grave o atual, temos que enviar os dados em formato de dados correto, conforme é o esperado pelo servidor.

No nosso caso, devemos enviar um **array** de **objetos** Javascript, cada objeto contendo o **nome de usuário** e o **número de palavras** no placar. Vamos começar obtendo estes dados da nossa tabela do placar.

Vamos começar criando o array vazio dentro da função **sincronizaPlacar()** ;

```
function sincronizaPlacar(){
    var placar = [];
}
```

Lendo a tabela placar

Queremos ler todas as linhas do placar, para então colocá-las dentro do array, então vamos começar buscando todas as linhas da tabela:

```
function sincronizaPlacar(){
    var placar = [];

    var linhas = $("tbody>tr");
}
```

Já sabemos que a função seletora do jQuery aceita qualquer seletor CSS, e desta vez utilizamos o seletor de filho direto para obter **todos** os `<tr>` 's que são filhos de um `<tbody>` .

Percorrendo as linhas de modo inteligente: a função $.each()

Agora que temos todas as linhas, precisamos percorrer cada linha obtendo o nome de usuário e de palavras. Poderíamos utilizar um **for** tradicional do Javascript para percorrer o array `linhas` , porém vamos utilizar um recurso do jQuery que é a função **each()**:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){

    });
}
```

A função **each()** executa a ação da função passada por parâmetro para cada item do array em que ela foi chamada, no caso, o array linhas. Uma vantagem dela é que não precisamos saber o tamanho do array que vamos percorrer, e ela também nos dá acesso ao próprio elemento que ela vai executar a função através do **this**.

Como o **this** neste caso é uma das linhas da tabela, queremos ter acesso ao conteúdo do **primeiro e segundo** `<td>` **'s**, pois são estes que guardam os nomes do usuário e o número de palavras.

Vamos criar variáveis para salvar os dois dados:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario =
        var palavras =

    });
}
```

Para buscar o `<td>` que é o primeiro filho do `<tr>` da iteração que estamos, podemos utilizar uma função conhecida nossa do jQuery, a função **.find()**, que faz as buscas nos elemento filhos. Porém não podemos utilizar o comando `this.find()`, pois o **this** neste caso é um objeto do Javascript que representa o elemento do HTML da linha, o `<tr>`. Precisamos empoderar este objeto Javascript e transformá-lo em um objeto jQuery atráves da função jQuery, assim ela terá acesso a função `.find()`:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find();
        var palavras = $(this).find();

    });
}
```

Buscando nos filhos com seletores avançados

Mas pelo o que devemos buscar com a `.find()`? Pelo **primeiro e segundo** `<td>`, que contêm os dados de usuário e número de palavras, respectivamente. Podemos utilizar qualquer seletor CSS nas funções de jQuery, tanto na função seletora ($) quanto na função `.find()`, e vamos nos aproveitar deste benefício para utilizar um seletor que nos retorna o `td` que é e**n**ésimo filho da linha: O seletor `nth-child()`:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)");
        var palavras = $(this).find("td:nth-child(2)");
```

```
    });
}
```

Com isto vemos a importância de ter um bom domínio de CSS até mesmo para trabalhar com Javascript e jQuery, já que com isto conseguimos tirar maior proveito das nossas conhecidas funções de seleção.

Como queremos obter o **texto** e não o elemento em si, vamos utilizar a função `.text()` para obter o texto dos `<td>` 's.

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

    });
}
```

Montando o objeto a ser enviado

A estrutura que devemos enviar para o servidor é um array de objetos, então precisamos salvar os dados que obtemos de cada linha dentro um novo objeto:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

        var score = {

        };

    });
}
```

Os objetos tem que ter duas propriedas, a primeira **usuario**, com o nome do usuario, e a segunda **pontos** com a quantidade de palavras digitadas:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

        var score = {
```

```
            usuario: usuario,
            pontos: palavras
        };

    });
}
```

Com um objeto por linha criado, basta adicioná-lo dentro do array placar utilizando a função push do Javascript:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

        var score = {
            usuario: usuario,
            pontos: palavras
        };

        placar.push(score);
    });
}
```

Fazendo um POST com Ajax: enviando o placar

Com os dados obtidos do placar, podemos começar a enviá-los para o nosso servidor. Precisamos realizar um AJAX,para que a página do usuário não recarregue, e tem que ser um POST para que os dados sejam gravados definitivamente.

Para realizar um post AJAX com jQuery , podemos utilizar a função **$.post**:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

        var score = {
            usuario: usuario,
            pontos: palavras
        };

        placar.push(score);

        $.post("http://localhost:3000/placar");
    });
}
```

Passamos o endereço para qual queremos fazer o POST, mas também precisamos passsar os **dados** que queremos enviar, neste caso o nosso placar, e uma função para ele executar após o sucesso da requisição.

O dado que queremos enviar é o array *placar*, porém já sabemos que só podemos enviar como dados das funções de AJAX do jQuery um **Objeto** ou uma **String**, por isso vamos colocar o nosso array placar dentro de um objeto antes de passá-lo para a função **$.post**:

```
function sincronizaPlacar(){
    var placar = [];
    var linhas = $("tbody>tr");

    linhas.each(function(){
        var usuario = $(this).find("td:nth-child(1)").text();
        var palavras = $(this).find("td:nth-child(2)").text();

        var score = {
            usuario: usuario,
            pontos: palavras
        };

        placar.push(score);

    });

    var dados = {
        placar: placar
    };

    $.post("http://localhost:3000/placar", dados, function(){

    });
 }
```

Vamos adicionar uma mensagem de sucesso na função de conclusão da $.post:

```
function sincronizaPlacar(){
        ... //restante da função
        });
    $.post("http://localhost:3000/placar", dados, function(){
        console.log("Placar sincronizado com sucesso");
    });
}
```

Agora jogue algumas vezes o jogo para que se crie um placar, clique no botão sincronizar e acesse a URL `http://localhost:3000/placar` para ver se o seu novo placar foi salvo com sucesso!

O que aprendemos:

- A importancia de dominar o CSS quando estamos trabalhando com JS e jQuery
- Utilizando seletores avançados com a função find
- A função $.each() para percorrer um array
- O formato correto de enviar dados com as funções de AJAX

- Enviando dados com a função $.post()