03

Removendo com fade

Transcrição

Um outra funcionalidade da nossa aplicação que precisa de uma pequena animação é a **remoção de linhas do placar**. Quando removemos alguma linha do placar, é uma remoção abrupta, a linha desaparece, e não suavemente.

Para melhor experiência do usuário, o ideal seria que a linha fosse esmaecendo aos poucos, até chegar num ponto em que ela desapareça.

A função que é responsável por remover as linhas da tabela é a `removeLinha`, dentro do **placar.js**. Logo, é nela que precisamos alterar a nossa funcionalidade.

O jQuery já possui uma função que vai diminuindo a opacidade de um elemento aos poucos, até o seu total desaparecimento, essa função é a `fadeOut`. Então vamos utilizá-la no lugar a função `remove`:

```
function removeLinha() {
    event.preventDefault();
    $(this).parent().parent().fadeOut();
}
```

Podemos reparar agora que o elemento vai desaparecendo aos poucos, mas será que ele é removido mesmo? Podemos inspecionar o HTML da tabela, e para nossa surpresa a linha continua lá, o que mudou foi que o seu `display` está com valor `none`. Ou seja, o `fadeOut` vai removendo a opacidade do elemento até um ponto e depois modifica a sua propriedade `display`, fazendo com que o elemento desapareça da tela, mas continue no HTML.

Não é exatamente isso que queremos, queremos que o elemento também seja removido do HTML, então vamos chamar a função `remove` logo após a função `fadeOut`. Para melhorar a semântica do nosso código, vamos também exportar a linha a ser removida para uma variável:

```
function removeLinha() {
    event.preventDefault();
    var linha = $(this).parent().parent();

    linha.fadeOut();
    linha.remove();
}
```

Podemos testar novamente o nosso código, mas parece que o `fadeOut` não está funcionando! Na verdade, ele está acontecendo, só que o JavaScript imediatamente chama a função `remove`, não esperando o término do `fadeOut` acontecer. O certo é só chamarmos a função `remove` após a função *`fadeOut`* terminar. E nós conseguimos fazer isso!

Primeiro, passamos um tempo em milissegundos por parâmetro para a função `fadeOut`, esse será o tempo de execução da função:

```
function removeLinha() {
    event.preventDefault();
    var linha = $(this).parent().parent();
```

```
    linha.fadeOut(1000);
    linha.remove();
}
```

E aguardamos 1 segundo para executar a função `remove`. Nós fazemos isso através da função , que recebe dois parâmetros: a função que desejamos executar após determinado tempo, e o próprio tempo que será aguardado para executar a função:

```
function removeLinha() {
    event.preventDefault();
    var linha = $(this).parent().parent();

    linha.fadeOut(1000);
    setTimeout(function() {
        linha.remove();
    }, 1000);
}
```

Então temos sempre que ter em mente que devemos remover um elemento após realizar o `fadeOut`, já que essa função só faz com que o elemento desapareça da tela, mas ainda fique no HTML da página.

Assim como existem o `slideUp`, `slideDown` e `slideToggle`, existem funções semelhantes que executam o *fade*, o `fadeIn`, `fadeOut` e `fadeToggle`, respectivamente.

O que aprendemos?

- A função `fadeOut`, que vai esmaecendo o elemento até o ponto dele desaparecer, mas ele continua no HTML.