02

Gabarito

Transcrição

Vamos começar pela função `guardaPalavraSecreta` :

```
var guardaPalavraSecreta = function () {

    // passa para o jogo o chute digitado pelo usuário
    jogo.setPalavraSecreta($entrada.val().trim());

    // limpa o campo de entrada

    $entrada.val('');

    // exibe para o usuário as lacunas de jogo.getLacunas();
    exibeLacunas();

    // muda o texto do placeholder para `chute`.
    mudaPlaceHolder('chute');
};
```

Fica evidente que precisamos implementar as funções `exibeLacunas()` e `mudaPlaceHolder()` . Vamos implementar a última primeiro, devido a sua brevidade:

```
var mudaPlaceHolder = function (texto) {

    $entrada
        .val('')
        .attr('placeholder', texto);
};
```

Usamos `val('')` para limpar o campo e `attr()` para mudarmos o valor de um atributo através do jQuery.

Agora, vamos para a implementação de `exibeLacunas()` .

Sabemos que `$lacunas` é o jQuery Object que aponta para o elemento `<ul class="lacunas">` de `index.html` . Precisamos adicionar dinamicamente as lacunas, mas antes de qualquer operação, faremos `$lacunas.empty()` . A função `empty()` remove todos os elementos filhos de um elemento pai. Estamos realizando este processo para garantirmos que toda vez que `exibeLacunas` for chamado, ele removerá todos os elementos já existentes para exibir elementos atualizados:

```
var exibeLacunas = function () {

    $lacunas.empty();

};
```

Agora, vamos iterar em `jogo.getLacunas()` e para cada lacuna criaremos dinamicamente uma `<li>` com a classe `lacuna` e com texto o valor da lacuna que estamos iterando:

```
var exibeLacunas = function () {

    $lacunas.empty();
    jogo.getLacunas().forEach(function (lacuna) {
        $('<li>')
            .addClass('lacuna')
            .text(lacuna)
            .appendTo($lacunas);
    });
};
```

Vamos entender essas instruções. Através de `forEach`, estamos iterando na lista retornada por `jogo.getLacunas()`. Para cada lacuna, criamos uma `<li>` dinamicamente através de `$('<li>')`. Vejam, não é um seletor CSS, mas uma tag passada como parâmetro para a função do jQuery e essa mudança já é suficiente para que o JQuery entenda que estamos criando um novo elemento. Em seguida, encadeamos as funções `addClass` e `text` para adicionarmos a classe `lacuna` e o texto da lacuna que estamos iterando. Por fim, através de `appendTo` indicamos em qual elemento do DOM desejamos adicionar o novo item que criamos, no caso, queremos adicionar em `$lacunas`, aquele elemento do DOM que referencia `<li class="lacunas">` que inicializamos logo no início da chamada de `iniciaController`.

Excelente, isso já é suficiente para entrarmos com a palavra secreta e vermos as lacuans em branco sendo exibidas, inclusive a alteração do valor do place holder.

Nosso código final ficará assim:

```
var criaController = function (jogo) {

    var $entrada = $('#entrada');
    var $lacunas = $('.lacunas');

    var exibeLacunas = function () {

        $lacunas.empty();
        jogo.getLacunas().forEach(function (lacuna) {
            $('<li>')
                .addClass('lacuna')
                .text(lacuna)
                .appendTo($lacunas);
        });
    };

    var mudaPlaceHolder = function (texto) {

        $entrada
            .val('')
            .attr('placeholder', texto);
    };

    var guardaPalavraSecreta = function () {

        jogo.setPalavraSecreta($entrada.val().trim());
```

```
            $entrada.val('');
            exibeLacunas();
            mudaPlaceHolder('chute');
        };

        var inicia = function () {

            $entrada.keypress(function (event) {
                if (event.which == 13) {
                    switch (jogo.getEtapa()) {
                        case 1:
                            guardaPalavraSecreta();
                            break;
                        case 2:
                            leChute();
                    }
                }
            });
        }

        return { inicia: inicia };
    };
```

Menos é mais

Por incrível que parece, utilizamos muito pouco e pontualmente o jQuery. Quando o assunto é manipulação de DOM, menos é sempre mais!