01

Game Over com eventos

Transcrição

Nosso jogo está começando a criar forma, mas está faltando algo característico: o fim do jogo (*Game Over*). Nós temos um tempo limite para o usuário digitar a frase, e esse tempo deve decrescer.

Quando o tempo se esgota, queremos que seja feita a contagem de quantas palavras foram digitadas. Dessa forma, criamos uma dificuldade para o jogador e o desafio fica mais interessante.

Vamos começar a implementar o contador de tempo.

Assim que o usuário clicar no campo de digitação, o contador inicia contagem regressiva.

Primeiramente, devemos detectar a ação do usuário "entrar no campo" do jogo, ou seja, clicar no *box* de digitação. Quando isso acontecer, o tempo deve começar a decrescer automaticamente. O usuário pode iniciar o jogo ao clicar no box de digitação, mas essa não é a única maneira, existe a opção da tecla "Tab".

Pensando nisso, há um evento específico para quando entramos dentro de um campo, que é o evento `focus`, que é quando o campo ganha o foco no programa, ou seja, é selecionado de alguma maneira. Logo, no **main.js**, podemos começar a implementar o nosso código:

```
var frase = $(".frase").text();
var numPalavras = frase.split(" ").length;
var tamanhoFrase = $("#tamanho-frase");
tamanhoFrase.text(numPalavras);

var campo = $(".campo-digitacao");
campo.on("input", function() {
var conteudo = campo.val();

var qtdPalavras = conteudo.split(/\S+/).length;
 $("#contador-palavras").text(qtdPalavras);

var qtdCaracteres = conteudo.length;
$("#contador-caracteres").text(qtdCaracteres);

campo.on("focus", function() {

});
```

Se queremos que o tempo decresça, temos que saber o seu valor. Para isso, na página `principal.html`, vamos envolver o tempo em uma tag `<span>`, e colocar o id `tempo-digitacao`:

```
    <ul class="informacoes">
        <li><span id ="tamanho-frase">19</span> palavras</li>
        <li><span id ="tempo-digitacao">10</span> segundos</li>
    </ul>
```

Voltando ao **main.js**, vamos pegar o conteúdo do `span` e salvá-lo em uma variável denominada `tempoRestante` :

```
var tempoRestante = $("#tempo-digitacao").text();
campo.on("focus", function() {

});
```

Feito isso, temos que implementar a lógica. A cada segundo que se passar, temos que subtrair **1** do nosso tempo restante. Para tal, vamos utilizar a função `setInterval()` do JavaScript, que faz com que uma determinada ação (passada como primeiro parâmetro) seja executada em um intervalo de tempo (passado como segundo parâmetro, no nosso caso, 1 segundo, ou 1000 milissegundos):

```
var tempoRestante = $("#tempo-digitacao").text();
campo.on("focus", function() {
    setInterval(function() {

    },1000);
});
```

Dentro de `setInterval` podemos subtrair 1 do nosso tempo restante a cada segundo que passe, logo:

```
campo.on("focus", function() {
    setInterval(function() {
        tempoRestante--;
    }, 1000);
});
```

Falta agora atualizarmos o contador com o tempo restante, assim ele subtrairá de `tempoRestante` e atualizará o valor do campo, bem semelhante ao que já fizemos anteriormente:

```
    setInterval(function() {
        tempoRestante--;
        $("#tempo-digitacao").text(tempoRestante);
    }, 1000);
```

Nosso tempo já está decrescendo, só que ainda não influencia em nada no jogo, já que o usuário consegue ficar digitando mesmo com o tempo zerado. Vamos então desabilitar o campo para que o usuário não consiga mais digitar nada quando o tempo zerar.

A `textarea` possui um atributo `disabled` , que faz com que não consigamos digitar nada na mesma (justamente o que queremos). Quando o tempo chegar a 0, o JavaScript colocará o atributo `disabled` na `textarea` .

Como queremos **adicionar um atributo**, o jQuery nos auxilia disponibilizando a função `attr` .

Essa função funciona de maneira semelhante à função `text` , podendo pegar o valor de um atributo ou modificá-lo. Por exemplo, para pegar o valor do atributo `rows` do nosso campo, fazemos:

```
var campo = $(".campo-digitacao");
campo.attr("rows");
```

E para modificar o mesmo, passamos mais um parâmetro para a função, que é o novo valor do atributo, por exemplo:

```
var campo = $(".campo-digitacao");
campo.attr("rows", 500);
```

Só que o atributo `disabled` não possui nenhum valor, só queremos colocá-lo na tag. Nesse caso, nós temos que informar isso passando o valor `true` (verdadeiro) para a função, assim estaremos "habilitando" o atributo:

```
campo.on("focus", function() {
    setInterval(function() {
        tempoRestante--;
        $("#tempo-digitacao").text(tempoRestante);
        campo.attr("disabled", true);
    }, 1000);
});
```

Como queremos desabilitar o campo somente quando o tempo chegar a 0, vamos envolver essa linha em uma condição `if`:

```
campo.on("focus", function() {
    setInterval(function() {
        tempoRestante--;
        $("#tempo-digitacao").text(tempoRestante);
        if (tempoRestante < 1) {
            campo.attr("disabled", true);
        }
    }, 1000);
});
```

Testamos e vemos que assim que o tempo chega a 0, o campo é travado. Mas ainda temos um bug, porque o tempo continua decrescendo depois do 0, ou seja, ele fica negativo. Temos que fazer com que a função `setInterval` pare quando o tempo for 0, mas como?

Para isso, existe a função `clearInterval`, que recebe o id do `setInterval` como parâmetro. Vamos colocá-la dentro do nosso `if`:

```
if (tempoRestante < 1) {
    campo.attr("disabled", true);
    clearInterval(id);
}
```

Mas atualmente, não temos acesso a essa id da função `setInterval()`, como consegui-lo? Toda função `setInterval()` **retorna o seu próprio id**, logo, basta guardarmos esse id em uma variável e passá-lo para a função `clearInterval`:

```
campo.on("focus", function() {
    var cronometroID = setInterval(function() {
        tempoRestante--;
        $("#tempo-digitacao").text(tempoRestante);
```

```
            if (tempoRestante < 1) {
                campo.attr("disabled", true);
                clearInterval(cronometroID);
            }
        }, 1000);
    });
```

Iremos testar nosso código e analisar se tudo funciona como o esperado.

Escutando um evento uma única vez

O que acontece se entrarmos vários vezes no nosso campo? Repare que toda vez que entramos no campo, o tempo decresce mais rápido, chegando até a ficar negativo! Isso acontece porque toda vez que o campo é focado, a nossa lógica é executada. O ideal seria que esse código só fosse executado **uma única vez**.

O problema é que a função `on` fica escutando o evento o tempo todo, e para que ela funcione somente na primeira vez, existe a função `one`, que funciona exatamente como a função `on`, só que só escuta o evento uma única vez:

```
var tempoRestante = $("#tempo-digitacao").text();
campo.one("focus", function() {
    var cronometroID = setInterval(function() {
        tempoRestante--;
        $("#tempo-digitacao").text(tempoRestante);
        if (tempoRestante < 1) {
            campo.attr("disabled", true);
            clearInterval(cronometroID);
        }
    }, 1000);
});
```

Resolvemos o bug, feito isso, implementamos a função de decrescer o tempo.

O que aprendemos?

- O evento de focus do jQuery
- Como implementar um cronômetro de tempo
- Relembrando as funções de tempo do Javascript: setInterval e clearInterval
- Manipulando atributos com a função .attr() do jQuery
- Escutando eventos com a função .one() do jQuery