03

Marcação da borda

Transcrição

O nosso aplicativo *Alura Typer* está ganhando forma, mas há pontos a melhorar. Primeiramente, iremos alinhar o texto à esquerda da tela no arquivo `estilos.css` :

```
.campo-digitacao{
    font-size: 20px;
    height: 130px;
}

.frase{
    font-size: 20px;
    text-align: left;
}

.campo-desativado{
    background-color: lightgray;
}

.icones{
    vertical-align: middle;
}
```

Outra melhoria a ser feita no jogo, é uma verificação visual se o usuário realmente acerta o texto a ser digitado. Queremos que o jogador perceba facilmente seus erros no texto.

Enquanto o jogador acerta as letras, mostraremos uma borda verde no campo de digitação, assim que um erro for cometido alteraremos a borda para a cor vermelha.

Evento input

Para comparar o texto digitado com a frase proposta, devemos associar o evento `input` ao nosso elemento `campo` . Já aprendemos isso nas aulas anteriores, basta usar a função `on` do jQuery.

```
campo.on("input",function()){

});
```

Dentro do `main.js` , por enquanto fora de qualquer outra função, vamos pegar a frase pelo seletor de classe do elemento e adicionar o evento ao `campo` :

```
var frase = $(".frase").text();
campo.on("input", function() {
    console.log(frase);
});
```

A frase completa está sendo exibida no navegador, e podemos digitá-la no campo. O próximo passo é comparar se o texto a ser digitado é compatível com o proposto pelo jogo.

Vamos capturar o texto digitado: basta usar a função `val()` do `campo`:

```
var frase = $(".frase").text();
campo.on("input", function() {
    var digitado = campo.val();
    console.log(frase);
    console.log(digitado);
});
```

Comparando o texto digitado

Sabemos qual é a frase e o que o jogador digitou. Falta verificar se o texto digitado realmente faz parte da frase para saber se o jogador errou ou não.

Repare que não podemos usar a condição `digitado == frase` pois isso verificaria sempre a frase inteira com o valor digitado. Mesmo se o jogador tenha escrito apenas uma palavra, o jogo consideraria errado, pois estaria comparando com a frase inteira.

Para saber se o jogador está certo ou não, usaremos apenas a parte inicial da frase que possui a mesma quantidade do valor digitado. Isso pode ser feito pela função `substr`:

```
var comparavel = frase.substr(0 , digitado.length);
```

A função `substr` devolve uma outra string com o tamanho definido nos parâmetros. O primeiro parâmetro é o inicio (`0`), ou seja, sempre a partir do primeiro char. O segundo define o fim que é justamente o tamanho do valor digitado.

Assim podemos realmente comparar o valor digitado com a `substring` correta da frase:

```
if(digitado == comparavel) {
    console.log("Está certo");
} else {
    console.log("Está errado");
}
```

Agrupando os códigos no evento, temos a seguinte forma:

```
var frase = $(".frase").text();
campo.on("input", function() {
    var digitado = campo.val();
    var comparavel = frase.substr(0 , digitado.length);

    if(digitado == comparavel) {
        console.log("Está certo");
    } else {
        console.log("Está errado");
    }
});
```

Ao testar o programa no navegador, poderemos observar no console que a comparação estará funcional.

Melhorando o feedback

Iremos melhorar o feedback para o usuário mostrando uma borda verde ou vermelha.

O primeiro passo é adicionar duas classes que representam a borda verde ou vermelha no nosso arquivo `estilo.css` :

```
.campo-digitacao{
    font-size: 20px;
    height: 130px;
    border: 3px solid black;
}

.frase{
    font-size: 20px;
    text-align left;
}

.campo-desativado{
    background-color: lightgray;
}

icones{
    vertical-align: middle;
}


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

});

.borda-verde{
    border: 3px solid green;
}

.borda-vermelha{
    border: 3px solid red;
}
```

Em `main.js` , falta adicionar a classe programaticamente no evento dentro do nosso `if` :

```
var frase = $(".frase").text();
campo.on("input", function() {
    var digitado = campo.val();
    var comparavel = frase.substr(0 , digitado.length);

    if(digitado == comparavel) {
        campo.addClass("borda-verde");
    } else {
        campo.addClass("borda-vermelha");
```

```
    }
});
```

Removendo a classe

Ao testar já aparece a borda verde como esperado, e se errarmos se torna vermelha. Porém, existe um problema: se corrigimos o texto digitado, a borda não volta para a cor verde, e sim continua vermelha.

Isso se deve a inserção da classe no campo, uma vez adicionada ela continua lá e, consequentemente, continua vermelho ou verde, dependendo da última classe adicionada.

Para resolver basta remover a outra classe dentro do `if` :

```
var frase = $(".frase").text();
campo.on("input", function() {
    var digitado = campo.val();
    var comparavel = frase.substr(0 , digitado.length);

    if(digitado == comparavel) {
        campo.addClass("borda-verde");
        campo.removeClass("borda-vermelha");
    } else {
        campo.addClass("borda-vermelha");
        campo.removeClass("borda-verde");
    }
});
```

Novamente vamos testar no navegador e a troca de cores funciona perfeitamente.

Encapsulando o código

Vamos seguir as boas práticas e isolar o código JavaScript dentro de uma função para não ficar solto dentro do arquivo `main.js` . Chamaremos a função de `inicializaMarcadores()`

```
function inicializaMarcadores() {
    var frase = $(".frase").text();
    campo.on("input", function() {
        var digitado = campo.val();
        var comparavel = frase.substr(0 , digitado.length);

        if(digitado == comparavel) {
            campo.addClass("borda-verde");
            campo.removeClass("borda-vermelha");
        } else {
            campo.addClass("borda-vermelha");
            campo.removeClass("borda-verde");
        }
    });
}
```

E assim, podemos adicionar `inicializaMarcadores()` na função que é chamada ao carregar a página:

```
$(function(){
    atualizaTamanhoFrase();
    inicializaContadores();
    inicializaCronometro();
    inicializaMarcadores(); //novo
    $("#botao-reiniciar").click(reiniciaJogo);
});
```

Inicializando sem cor

A marcação da borda já funciona, mas ainda podemos melhorar um detalhe: ao jogar e reinicializar o jogo através de nosso botão, percebemos que a borda continua pintada e não volta ao estado inicial. Faz todo sentido começar com uma borda preta, e a cor vir apenas quando realmente começamos a digitar.

Para reiniciar o jogo corretamente basta remover as classe de borda do campo na função `reiniciaJogo` :

```
function reiniciaJogo(){
    campo.attr("disabled",false);
    campo.val("");
    $("#contador-palavras").text("0");
    $("#contador-caracteres").text("0");
    $("#tempo-digitacao").text(tempoInicial);
    inicializaCronometro();
    campo.toggleClass("campo-desativado");

    campo.removeClass("borda-vermelha"); //novo
    campo.removeClass("borda-verde"); //novo
}
```