01

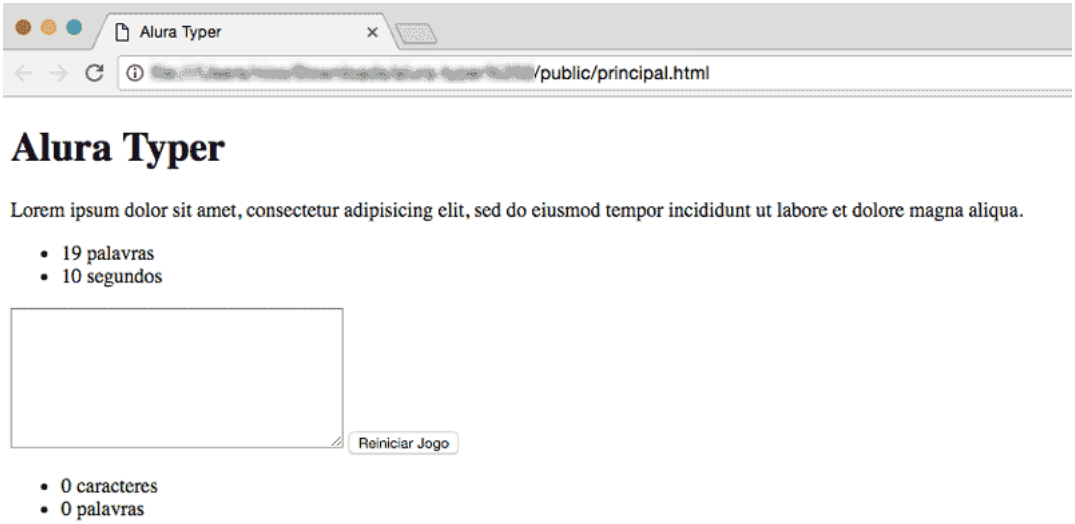
Reiniciando o jogo

Transcrição

Queremos dar a opção do nosso usuário reiniciar o nosso jogo sem ter que recarregar a página. Para isso, implementaremos um pequeno botão, que ao ser clicado, irá zerar os nossos campos e contadores, permitindo o usuário iniciar uma nova rodada no jogo.

Em `principal.html` adicionaremos um `<button>`, abaixo da `<textarea>`:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UFT-8">
    <title>Alura Typer</title>
</head>
<body>
    <h1>Alura Typer</h1>
    <p class="frase">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod ter

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

    <textarea class="campo-digitacao" rows="8" cols="40"></textarea>
    <button id="botao-reiniciar">Reiniciar Jogo</button>

    <ul>
            <li><span id="contador-caracteres">0</span> caracteres</li>
        <li><span id="contador-palavras">0</span> palavras</li>
    </ul>

    <script src="js/jquery.js"></script>
    <script src="js/main.js"></script>

</body>
</html>
```



Agora precisamos atrelar um evento de click neste botão recém criado. Até agora, vimos como fazer isto através da função `on("click",function(){...})` do jQuery, porém vamos aprender um novo modo de ter acesso a este evento.

Um atalho para função on("click"), a shorthand click()

Os eventos do Javascript que são mais comuns, como o **click**, **blur**(evento de sair de um campo), **dblclick** (clique duplo) tem funções próprias no jQuery, que são funções de atalho, evitando precisar chamar a função `on()` e chamando diretamente a função do próprio evento.

Veja no *exemplo* abaixo, onde capturamos o evento de click em um título e exibimos uma mensagem no console como resposta:

```
var titulo = $("#titulo");
titulo.on("click",function(){
    console.log("Titulo foi clicado!");
});
```

Poderíamos executar este mesmo evento e ação com a função de atalho **click()**:

```
var titulo = $("#titulo");
titulo.click(function(){
    console.log("Titulo foi clicado!");
});
```

Podemos ser mais sucintos ainda e escrever tudo em uma linha só:

```
$("#titulo").click(function(){
    console.log("Titulo foi clicado!");
});
```

Como o jQuery é uma grande ajuda na produtividade, ele disponibiliza algumas dessas funções de atalho que são mais curtas e simples, agilizando o trabalho do programador. Ao clicar no `.click`, queremos que *o campo seja liberado novamente* e que tudo o que havia dentro dele, que *seja apagado*.

Em `main.js` iremos atrelar um evento de click no nosso botão com esta nova função.

Reiniciando o jogo: tratando do campo

Agora que estamos detectando o evento de click no botão reiniciar, vamos começar a retornar o jogo para o estado inicial, como se o usuário estivesse acabado de entrar na página. O primeiro passo é habilitarmos o campo novamente, pois quando colocamos o atributo **disabled** nele na hora do *Game Over*, impedimos o usuário de digitar.

Para removê-lo, vamos nos aproveitar da função `.attr()` do jQuery no arquivo `main.js` :

```
$("#botao-reiniciar").click(function(){
    campo.attr("disabled", false);
});
```

A função `.attr()` nos permite **colocar, retirar ou modificar** valores de atributos de elementos HTML.

Com o campo digitável, precisamos zerá-lo, pois ele ainda contém os resquícios da última jogada do nosso usuário. Queremos que o **conteúdo** da `<textarea>` seja zerado, fique vazio.

Sabemos que para manipular o conteúdo de um elemento de input do usuário, como a `<textarea>` , temos que utilizar a função `.val()` , e como queremos que ela fique vazia, faremos assim:

```
$("#botao-reiniciar").click(function(){
    campo.attr("disabled", false);
    campo.val("");
});
```

Colocando o conteúdo `.val()` do campo como vazio(""), ele ficará zerado.

Agora nosso campo está limpo e pode ser digitado novamente.

Reiniciando o jogo: zerando os contadores

Nosso próximo passo é *retornar os contadores para seu estado inicial*, ou seja, com zero palavras.

Este é mais fácil, basta selecionar os dois contadores e substituir o **texto** dos `<span>` por **0**.

```
$("#botao-reiniciar").click(function(){
    campo.attr("disabled", false);
    campo.val("");
    $("#contador-palavras").text("0");
    $("#contador-caracteres").text("0");
});
```

Com os contadores zerados, fica faltando apenas *ajustar o tempo inicial*!

Reiniciando o jogo: restaurando o tempo inicial

Para obtermos o tempo inicial, temos que guarda-lo numa variável assim que iniciamos a aplicação, pois como alteramos o valor do tempo depois, acabamos perdendo-o. Em `main.js` criaremos uma variável chamada `tempoInicial` e capturaremos o valor assim que o script for carregado, pois atualmente ele é calculado a partir do `tempoRestante`, e assim podemos pegar o valor inicial salvo para reiniciar o jogo:

```
var tempoInicial = $("#tempo-digitacao").text();

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

Agora que temos o valor do tempo inicial, podemos reatribuí-lo ao marcador de tempo quando clicarmos no botão de reiniciar:

```
$("#botao-reiniciar").click(function(){
    campo.attr("disabled", false);
    campo.val("");
    $("#contador-palavras").text("0");
    $("#contador-caracteres").text("0");
    $("#tempo-digitacao").text(tempoInicial);
});
```

Feito isso, o tempo inicial é restaurado corretamente.

Reiniciando o jogo: atrelando o evento de focus ao campo

Nosso botão reiniciar está quase funcionando completamente, ele já zera os campos e contadores, e restaura o tempo inicial. No entanto, ao testar o botão, percebemos que o jogo não será reinicializado.

O evento `input` do `campo` foi associando apenas uma vez (`one`) pelo nosso código. Para resolver o problema, bastaria atrelar um novo evento de `focus` ao `campo`, que fará o trabalho do cronômetro. O nosso evento de `focus` deve ser atribuído novamente com a função `one` do jQuery, pois só queremos que ele seja executado uma única vez.

Para fazer isto, poderíamos simplesmente copiar e colar o código que implementa o cronômetro, porém isto seria uma má prática de programação, pois estaríamos colando código repetido em dois lugares diferentes.

Sabemos que podemos reaproveitar este pedaço de código caso ele estivesse mais modularizado e organizado em funções. Como nosso código está crescendo muito, agora é uma boa hora de fazer esta separação de responsabilidades.

Organizando o nosso código em funções

Vamos separar cada bloco de código em uma função com um nome bem semântico que especifique o que aquele bloco de código fará. Vamos começar pelo topo do arquivo `main.js`, em que temos este bloco de código:

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

Vemos que ele é responsável por *atualizar o tamanho da frase*, logo vamos envolvê-lo numa função de mesmo nome:

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

Continuando a olhar o arquivo `main.js`, vemos o trecho abaixo:

```
campo.on("input",function(){
    var conteudo = campo.val();

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

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

Este pedaço de código é inteiramente responsável por *inicializar os contadores* de palavras e caracteres. Vamos envolver este bloco em uma função chamada `inicializaContadores()`:

```
function inicializaContadores() {
    campo.on("input",function(){
        var conteudo = campo.val();

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

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

Continuando a olhar o `main.js`, vamos encontrar mais um bloco que já é necessário que se torne uma função: o bloco que inicializa o cronômetro para marcar o tempo:

```
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);
});
```

Vamos envolvê-lo em uma função chamada `inicializaCronometro()` :

```
function inicializaCronometro() {
    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);
    });
}
```

Por último o nosso código recém escrito, que tem como função *reiniciar o nosso jogo*. Vamos separar a função seletora da função de reiniciar e também separá-lo em uma função com nome semântico:

```
$("#botao-reiniciar").click(reiniciaJogo);

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

Com isso, nosso código está bem mais organizado e modularizado, com blocos de função e responsabilidades bem definidas.

Podemos agora chamar a função `inicializaCronometro` dentro da `reiniciaJogo` , já que agora são códigos reaproveitáveis:

```
$("#botao-reiniciar").click(reiniciaJogo);

function reiniciaJogo(){
    campo.attr("disabled",false);
    campo.val("");
    $("#contador-palavras").text("0");
    $("#contador-caracteres").text("0");
    $("#tempo-digitacao").text(tempoInicial);
    // Adicionamos aqui \/
    inicializaCronometro();
}
```

Graças a nossa separação, podemos fazer o reuso de uma mesma função em diversos pedaços do código.

Código organizado...mas funcionando?

Conseguimos separar nosso código em diversas funções, com nomes bem claros que especificam suas funcionalidades, porém se você abrir a página, verá que o nosso jogo Alura Typer não está mais funcionando como deveria! Ele não conta palavras, os eventos não são atribuídos e tudo que foi feito até agora parou de funcionar.

Como agora todo nosso código está isolado dentro de funções, precisamos que alguém invoque estas funções para que elas sejam executadas. Afinal, nossos eventos só serão atrelados aos elementos HTML quando nossas funções forem chamadas.

Para fazer isto, vamos utilizar uma função do jQuery que aguarda a página ser carregada e depois executa seu conteúdo: a função `$(document).ready()` :

Sabemos que para que o Javascript manipule uma página com segurança ele deve aguardar todos os seus elementos serem carregados, e é exatamente ai que a função `$(document).ready()` vai entrar. Vamos colocá-la no nosso código e assim que a página for carregada ela irá executar nossas funções para nós:

```
var tempoInicial = $("#tempo-digitacao").text();

$(document).ready(function(){
    atualizaTamanhoFrase();
    inicializaContadores();
    inicializaCronometro();
    $("#botao-reiniciar").click(reiniciaJogo);
});

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


var = campo = $(".campo-digitacao");

function inicializaContadores() {
    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);
    });
}

function inicializaCronometro() {
    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);
    });
}

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

Vamos aproveitar e atrelar o evento da função "iniciar" dentro dela também.

Como esta é uma função bastante utilizada do jQuery, ela também tem um atalho, que é a função chamada: `$(function() { ... });` . Quando passamos uma função dentro da função **$()** , estamos na verdade utilizando a função `$(document).ready()` . Como é mais prático utilizar este segundo modo, vamos alterar nosso código:

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

Conferindo a organização

Depois de separado e organizado, nosso código deve estar deste modo:

```
var tempoInicial = $("#tempo-digitacao").text();
var campo = $(".campo-digitacao");

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

function atualizaTamanhoFrase(){
    var frase = $(".frase").text();
    var numPalavras  = frase.split(" ").length;
    var tamanhoFrase = $("#tamanho-frase");
    tamanhoFrase.text(numPalavras);
}

function inicializaContadores() {
    campo.on("input",function() {
        var conteudo = campo.val();

        var qtdPalavras = conteudo.split(" ").length;
```

```
        $("#contador-palavras").text(qtdPalavras);

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

function inicializaCronometro() {
    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);
    });
}

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

Apesar deste pequeno trabalho inicial de organizar o código, daqui pra frente mantê-lo será muito mais fácil, pois agora podemos usar e abusar de nossas funções para reaproveitar bastante as funcionalidades.

E nossa função de reiniciar funciona corretamente, fazendo com que nosso usuário possa jogar o AluraTyper repetidas vezes sem precisar utilizar o botão F5.

O que aprendemos ?

- A função `.click()`
- Remover atributos com a função `.attr()`
- Como esperar o carregamento da página com $(document).ready();
- O atalho $(function(){})