FACULDADE DE TECNOLOGIA DO PIAUÍ - FATEPI
COORDENAÇÃO DE SISTEMAS DE INFORMAÇÃO
BACHARELADO EM SIST. DE INFORMAÇÃO
DISCIPLINA: WEB DESIGN
PROFESSOR: MSC. SHALTON VIANA
PERÍODO: V
GRUPOS:

GRUPO1: Luis Lima e Matheus
GRUPO2: Renan e Jose Augusto
GRUPO3: Hudson e Roberto
GRUPO4: Sebastiao e Fabio

NOTA 5,0

O trabalho consiste em desenvolver uma aplicação simples usando biblioteca JQuery com as seguintes funcionalidades:

1. Contando as palavras da frase e atualizando o seu contador
2. Contando através de eventos
3. Implementando a lógica de Game Over
4. Criando o botão reiniciar
5. Organizando o código
6. Aplicando CSS
7. Usando ícones
8. Verificando a digitação
9. Armazenando a pontuação do usuário
9.1 Salvando a pontuação do usuário
10. Chamando a inserePlacar
11. Removendo linhas do placar
12. Removendo a linha
13. Organizando o nosso código

O trabalho poderá acontecer em grupo de no Maximo dois alunos. A atividade deverá ser desenvolvida usando ferramenta de gerenciamento de projeto, e esta deverá ser liberada através de link para que o professor tenha acompanhamento da mesma. Ainda assim, o grupo poderá utilizar uma plataforma de hospedagem de código-fonte e arquivos para controle de versão, como git e github. E assim, deverá ser enviado também o acesso ao versionamento do código ao professor.

Nesse trabalho, cada membro deverá apresentar os resultados da implementação, explicando os efeitos de cada implementação de código nos módulos correspondentes, e enviar o código fonte conforme arvore estabelecida acima.

Para praticar o conhecimento que iremos adquirir de jQuery, iremos desenvolver a aplicação ProjetoTyper. O ProjetoTyper é um jogo que tem como objetivo medir a velocidade de digitação de seus usuários, e salvar seu recordes em um placar. Esta aplicação parece simples, mas irá envolver quase toda a biblioteca de funções do jQuery, nos fazendo passar desde o básico de manipulação de elementos até requisições assíncronas com AJAX.

```
projeto-typer/
├── public
│   ├── css
│   │   └── estilos.css
│   ├── fonts
│   ├── img
│   ├── js
│   │   └── jquery.js
│   └── principal.html
└── servidor
```

1. Contando as palavras da frase e atualizando o seu contador

Agora podemos começar o desenvolvimento da página **principal.html**, o primeiro passo é escrever um HTML inicial, com o título da nossa aplicação, uma frase , e uma `<ul>` que conterá o número de caracteres e de palavras da frase, e algumas classes para manipular os elementos mais adiante:

```
<body>
    <h1>Projeto Typer</h1>
    <p class="frase">Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua.</p>

    <ul class="informacoes">
        <li>19 palavras</li>
        <li>15 segundos</li>
    </ul>
</body>
```

Para atualizar os contadores de acordo com a frase que está no HTML, teremos que começar a utilizar o Javascript com jQuery. Então vamos criar um arquivo que conterá nosso código, o arquivo **main.js** dentro da pasta **public/js**, e vamos importá-lo na página **principal.html**, como último elemento da tag `<body>`.

Vamos aproveitar para importar o jQuery também, como queremos usar as funções suas dentro no **main.js**, vamos importá-lo como primeiro script:

```
<--! Resto do código HTML -->
    <script src="js/jquery.js"></script>
    <script src="js/main.js"></script>

</body>
</html>
```

No **main.js**, vamos acessar a frase utilizando o atalho para a função `jQuery`, acessando-a através da classe do seu elemento:

```
$(".frase");
```

Vamos acessar o seu conteúdo, o seu texto, através da função `text()` e guardá-lo em uma variável:

```
var frase = $(".frase").text();
```

Sabemos que as palavras são separadas por um espaço em branco, então vamos quebrar a frase onde houver espaços, utilizando a função `split()`:

```
var frase = $(".frase").text();
frase.split(" ");
```

O retorno dessa função é um array com as palavras separadas, então se acessarmos o seu tamanho (**length**), temos o número de palavras da frase. Vamos guardar esse resultado em uma variável também:

```
var frase = $(".frase").text();
var numPalavras = frase.split(" ").length;
```

Já conseguimos contar a quantidade de palavras, mas ainda falta atualizar o contador no HTML. Primeiramente devemos selecionar o elemento HTML que contém a contagem de palavras. Como queremos somente o número, vamos envolvê-lo em um `span` e colocar um id nele:

```
<!-- Página principal.html -->
<!-- Restante do código HTML -->
    <ul class="informacoes">
        <li><span id="tamanho-frase">19</span> palavras</li>
        <li>15 segundos</li>
    </ul>
<!-- Restante do código HTML -->
```

Agora, no **main.js**, vamos selecionar essa tag `span`, e modificar o seu conteúdo, passando o número de palavras por parâmetro para a função `text()`:

```
var frase = $(".frase").text();
var numPalavras = frase.split(" ").length;

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

Ou seja, a função **.text()** tem dois comportamentos, o primeiro , quando utilizamos-a sem **nenhum parâmetro**, nos é **retornado** o valor de texto do elemento, e o segundo, quando passamos um parâmetro para a função, ela **altera** o valor de texto do elemento!

Ao reiniciar sua página, você deve notar que nosso indicador atualizou-se para o valor de palavras da frase! Conseguimos fazer a contagem automática de palavras.

2. Contando através de eventos

Vamos adicionar uma `textarea` à nossa página e começar a trabalhar com eventos. Mãos à obra :

1) Abra o arquivo `public/principal.html` e adicione um `<textarea>` e um `ul`, logo após a `ul` de informações:

```
<textarea class="campo-digitacao" rows="8" cols ="40"></textarea>

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

2) Vamos adicionar um evento ao nosso `campo-digitacao`. Abra o arquivo `public/js/main.js` e coloque no final:

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


});
```

Observa-se que foi selecionado o campo pelo nome da classe e já associamos o evento `input` com ele.

3) Dentro da função anônima do evento recupere o valor (`val()`) do campo, conte as palavras usando a função `split(..)` e imprima no console.

```
var conteudo = campo.val();
var qtdPalavras = conteudo.split(/\S+/).length - 1;
$("#contador-palavras").text(qtdPalavras);
```

Você já pode testar esse código no navegador. O contador de palavras deve mostrar a quantidade de palavras.

4) Logo após do `$("#contador-palavras")` atualize também o contador de caracteres:

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

5) Salve e teste o seu código no navegador:

6) Corrigindo o Bug do espaço sendo contado como caractere:

```
    var conteudo = campo.val();

    //Retira os espaço da String
    var conteudoSemEspaco = conteudo.replace(/\s+/g,'');

    var qtdCaracteres = conteudoSemEspaco.length;
    $('#contador-caracteres').text(qtdCaracteres);
```

3. Implementando a lógica de Game Over

Assim que o usuário clicar no campo de digitação, deve começar a contagem regressiva do jogo. Como apresentado em aula, vamos implementar essa funcionalidade:

1) Abra o arquivo `principal.html` e envolva o tempo em uma tag `<span>`, colocando o id `tempo-digitacao`. Adicione o elemento `span` apenas dentro da segunda `li`:

```
<ul class="informacoes">
    <li><span id="tamanho-frase">5</span> palavras</li>
    <li><span id="tempo-digitacao">10</span> segundos</li>
</ul>
```

Repare que foi usado agora a `ul` com a classe `informacoes`.

2) Abra o arquivo `main.js` e acrescente no final o código para selecionar o elemento `span`:

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

3) Ainda no `main.js` adicione logo depois da variável `tempoRestante` o evento `focus` que fica associando com o nosso `campo`:

```
var tempoRestante = $("#tempo-digitacao").text();
campo.one("focus", function() {
    //aqui vem mais
});
```

Repare que já foi usada a função `one` que garante que o evento será associado apenas uma vez.

4) Dentro da função anônima usa o `setInterval` para diminiur o `tempoRestante` a cada segundo:

```
var tempoRestante = $("#tempo-digitacao").text();
campo.one("focus", function() {
    var cronometroID = setInterval(function() {
        tempoRestante--;
        //aqui vem mais
    }, 1000);
});
```

Perceba que a função `setInterval` devolve uma id (`cronometroID`).

5) Agora só falta atualizar o `tempo-digitacao` no DOM e verificar se o tempo já esgotou. Lembre-se de desabilitar o `campo` através da função `attr` e de limpar o intervalo (`clearInterval`):

```
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);
});
```

6) Pronto, salve tudo e teste no seu navegador!

4. Criando o botão reiniciar

1) Como primeiro passo, adicione um `<button>` na página `principal.html`, logo abaixo da `<textarea>`:

```
<button id="botao-reiniciar">Reiniciar Jogo</button>
```

2) No arquivo `main.js` atrele o evento de `click` no nosso botão. Dentro da função do evento reative o campo pela função `attr`:

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

3) Ainda na função do evento reinicie o `campo`, o `contador-palavras` e o `contador-caracteres`:

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

4) Falta ainda reinicializar o tempo inicial. Vamos guardar o `tempoInicial` em uma variável auxiliar. No início do arquivo `main.js`, logo após `tamanhoFrase.text(numPalavras);` adicione:

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

5) Agora use esssa variável dentro da função do evento `click` do nosso botão:

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

    $("#tempo-digitacao").text(tempoInicial); //novo

});
```

Isso faz que o tempo inicial volte ao elemento `tempo-digitacao`.

6) Salve tudo e teste no navegador. O botão já deve reinicializar os valores.

5. Organizando o código

Objetivo desse exercício é separar cada bloco de código em uma função e resolver o problema de reinicialização do jogo. **Muito cuidado nesse exercício para realmente usar o mesmo código dentro das funções.**

1) Envolva as variáveis `frase`, `numPalavras` e a inicialização da `tamanhoFrase` dentro de uma função `atualizaTamanhoFrase`:

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

2) Envolva o evento `input` do campo dentro de uma função

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

        var qtdPalavras = conteudo.split(/\S+/).length - 1;
        $("#contador-palavras").text(qtdPalavras);

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

    });
}
```

Cuidado, repare que a inicialização da variável `campo` não ficou nessa função!

3) Agora envolva a variável `tempoRestante` *junto* com o evento `focus` do `campo` dentro de uma função `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);
    });
}
```

4) O nosso botão para reiniciar vai receber na função `click` o nome da função:

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

Todo o código que estava dentro da função do evento `click` deve estar dentro da função `reiniciaJogo`:

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

Repare que já estamos chamando a função `inicializaCronometro` dentro da função `reiniciaJogo`.
Faça as alterações!

5) Como agora todo nosso código está isolado dentro de funções, precisamos que alguém invoque estas funções para que elas sejam executadas!

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

Para tal, adicione no ínicio da página logo após da declaração das variaveis já existentes:

```
//as duas vars já devem existir
var campo = $(".campo-digitacao");
var tempoInicial = $("#tempo-digitacao").text();

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

//outras funções omitidas
```

6) Salve e teste o seu código!

6. Aplicando CSS

6. 1 Melhorando o visual

O visual do nosso jogo ainda está bastante simples razão suficiente para aplicar um CSS, não? Usaremos o framework Materialize para estilizar a página que auxilia muito nessa tarefa.

Para tal:

1) Abra o arquivo `principal.html` e *adicione* dentro do `<head>` logo abaixo da tag `<title>`:

```
<link rel="stylesheet" href="css/libs/materialize.min.css">
```

Recarregue a página dentro do navegador e repare a diferença. Para sua comparação o `<head>` inteiro ficou como:

```
<head>
    <meta charset="UTF-8">
    <title>Projeto Typer</title>
    <link rel="stylesheet" href="css/libs/materialize.min.css"> <!--
novo -->
</head>
```

Obs: *O arquivo `materialize.min.css` já está dentro da pasta `public/css/libs` do seu projeto.*

2) Agora, na mesma página `principal.html`, coloque todo o conteúdo do `body` (exceto os scripts) dentro de uma `div` com a classe `container`. Adicione apenas o `div`:

```
<body>
    <div class="container">
        <!-- conteúdo da página aqui -->
    </div>

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

Novamente, recarregue a página no navegador para ver a diferença

3) Após isso, vamos aumentar a altura do campo, alinhar a frase à esquerda, e aumentar as suas fontes. Abre o arquivo `public/css/estilos.css` e coloque esse código abaixo:

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

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

4) No arquivo `principal.html` importe o `estilo.css` dentro da tag `<head>`:

```
<head>
    <meta charset="UTF-8">
    <title>Projeto Typer</title>
    <link rel="stylesheet" href="css/libs/materialize.min.css">
```

```
    <link rel="stylesheet" href="css/estilos.css"> <!-- novo -->
</head>
```

5) Para deixar mais claro que o jogo terminou, deixe o campo de digitação cinza quando o tempo esgotar. Dentro do arquivo `public/css/estilo.css` adiciona uma nova clase:

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

6) No arquivo `main.js`, quando o tempo se esgotar, devemos adicionar essa classe **`campo-desativado`** ao campo usando jQuery. Quando o usuário clicar no botão reiniciar, o campo deveria ser reativado. Para não espalhar as funções `addClass` e `removeClass` do jQuery, vamos aproveitar a função `toggleClass` que ativa e desativa respectivamente.
Abra o arquivo `main.js` e procure a função `inicializaCronometro`:

```
//dentro do arquivo main.js, dentro da função inicializaCronometro

//dentro desse if adicione apenas a linha com toggleClass
if (tempoRestante < 1) {
    campo.attr("disabled", true);
    clearInterval(cronometroID);
    campo.toggleClass("campo-desativado"); //novo
}
```

Adicione a mesma funcionalidade no final da função `reiniciaJogo`:

```
function reiniciaJogo() {
    //código omitido e não alterado
    campo.toggleClass("campo-desativado"); //novo
}
```

7) Salve os arquivo e teste no navegador. Após o tempo esgotar, o campo de digitação deve ficar cinza. Ao reiniciar volta a ser ativado.