Essa propriedade deixa reservado o espaço para a barra de rolagem e não deixa a pagina se mover dependo das novas propriedades

body{

overflow: scroll;

}

Essa propriedade serve para mostrar ou esconder algum elemento automaticamente dependendo do valor atual. (substitui o .show() e hide())

$('.placar').toggle();

A função slideDown e slideUp são efeitos de transição junto com a função show e hide

$(".placar").slideDown(600);

$(".placar").slideUp(600);

Essas funções podem ser substituidas por

$(".placar").slideToggle(600);

Função para dar um display none com efeito de transição e depois reniver o elemento

var linha = $(this).parent().parent().fadeOut(800);

setInterval(function(){

linha.remove();

}, 800);

Assim como existem o slideUp, slideDown e slideToggle, existem funções semelhantes que executam o fade, o fadeIn, fadeOut e fadeToggle, respectivamente.

Para evitar que a animação fique se repetindo caso o usuario click muitas vezes no botão pode-se usar o stop, que para a animação antes de efetuar uma nova

$(".placar").stop().slideToggle(800);

A função abaixo pega a posição top da classe placar, e a função animate faz um scroll para essa posiçaõ(A função animate precisa estar tanto no html como no body, cada navegador le de um local)

function scrollPlacar(){

var posicaoPlacar = $('.placar').offset().top;

$("html,body").animate(

{

scrollTop: posicaoPlacar + "px"

}, 1000);

}

Outras formas de esconder um elemento

var promocoes = $('.promocoes');

if(promocoes.is(':visible')) {

promocoes.hide();

} else {

promocoes.show();

}

A função toogleClass (não confundir com a função toggle) adiciona uma classe caso ela não exista no elemento. Se existir, ela remove a classe.

$('#botao-promocao').click(function() {

$('.promocoes').toggleClass('invisivel');

});

var promocoes = $('.promocoes');

if(promocoes.hasClass('invisivel')) {

promocoes.removeClass('invisivel');

} else {

promocoes.addClass('invisivel');

}

Função para buscar dados em uma rota por get. O mathfloor serve para aredondar o numero para baixo e a math.rondom busca um numero aleatório entre 0 e 1, por isso estamos multiplicando pelo tamanho do array para que busque a posição correta e não quebre

function fraseAleatoria(){

$.get('http://localhost:3000/frases', function(data){

var numAleatorio = Math.floor(Math.random() \* data.length);

var frase = $('.frase').text(data[numAleatorio].texto);

});

}

Para colocar um ícone de carregamento enquanto não carrega a requisição ajax

<img src="img/spinner.gif" alt="Spinner" id="spinner">

Dentro do método frasealeatória colocar o código abaixo para que antes que comece a requisição comece a mostrar o gif

$('#spinner').show();

Dentro do método acrescentar o Always, que sempre sera executado, mesmo quando der erro

$.get("http://localhost:3000/frases", trocaFraseAleatoria)

.fail(function(){

$("#erro").toggle();

setTimeout(function(){

$("#erro").toggle();

},1500);

})

.always(function() {

$('#spinner').hide();

})

Botão para buscar uma frase especifica

<a class="btn-floating btn-large waves-effect waves-light cyan" id="botao-frase-id">

<i class="material-icons">repete\_one</i>

</a>

<input type="number" id="frase-id" min="0">

$('#botao-frase-id').click(buscaFrase);

function buscaFrase(){

$('#spinner').show();

var dados = {

id: $('#frase-id').val()

}

$.get('http://localhost:3000/frases', dados, trocaFrase)

.fail(function(){

$("#erro").toggle();

setTimeout(function(){

$("#erro").toggle();

},1500);

})

.always(function() {

$('#spinner').hide();

});

}

function trocaFrase(data){

$('.frase').text(data.texto);

atualizaTamanhoFrase();

atualizaTempoInicial(data.tempo);

}

Botão para sincronizar os usuários da view com o servidor por post

<a class="btn-floating btn-large waves-effect waves-light yellow" id="botao-sync">

<i class="material-icons">swap\_vert</i>

</a>

$('#botao-sync').click(sincronizaPlacar);

function sincronizaPlacar(){

var placar = [];

var linhas = $('tbody>tr');

linhas.each(function(){

var score = {

usuario: $(this).find('td:nth-child(1)').text(),

pontos: $(this).find('td:nth-child(2)').text()

}

placar.push(score);

var dados = {

placar: placar

};

$.post('http://localhost:3000/placar', dados, function(){

console.log('salvou os dados no servidor');

})

})

}

Função para buscar os dados no servidor assim que abrir a tela

function atualizaPlacar(){

$.get('http://localhost:3000/placar', function(data){

$(data).each(function(){

var linha = novaLinha(this.usuario, this.pontos);

linha.find('.botao-remover').click(removeLinha);

$('tbody').append(linha);

});

})

}

Agora que aprendemos bastante sobre AJAX é hora de conversar sobre um pequeno detalhe que é preciso se preocupar quando trabalhamos com AJAX.

Lembrando que nossa aplicação Alura Typer está hospedada no servidor que levantamos na linha de comando. Ou seja, na hora de acessar <http://localhost:3000/principal.html> o servidor deveria estar em pé para servir a página ao navegador

Também já mostrei para vocês que existe um outro servidor na rede que também está rodando a aplicação. Essa aplicação é a mesma, o mesmo código rodando, no entanto em uma máquina diferente e por isso é necessário trocar localhost pelo IP de maquina, no nosso caso 192.168.0.83

**Acessando as frases com outro IP**

Se a aplicação nessa outra máquina possui as mesmas funcionalidades, também podemos acessar /frases para ver todas as frases disponíveis. Não há mistério e funcionou. Ao acesssar <http://192.168.0.83/frases> aparece a lista de frases dessa aplicação.

Vamos pensar que devemos puxar as frases do outro servidor que não é o local. Isso é algo muito comum na web. Em outras palavras, queremos pegar dados de mais de um servidor! A nossa primeira ideia é simples: No arquivo frase.js, na função fraseAleatoria vamos usar o endereço do outro servidor:

//frase.js, na função fraseAleatoria

$.get("http://192.168.0.83/frases". trocaFraseAleatoria)

//resto omitido

Como tem no outro servidor a mesma aplicação com mesmo código, o esperado é que a requisição funcione! Para nossa surpresa ela falha! O que funciona em uma aplicação não funciona na outra! Como assim?

**Entendendo a Same Origin Policy**

Ao analisar o console percebemos que o navegador simplesmente não permitiu a requisição para outro servidor. Isto é pois o navegador possui uma proteção que é chamada de **Same Origin Policy**.

*Same Origin* significa que por padrão o navegador não permite chamar um outro servidor que não é da mesma origem. Ou seja, se a aplicação foi carregada pelo localhost:3000, o navegador só permite requisições para localhost na porta 3000. Até o protocolo importa, por exemplo o navegador não permite mudar a origem do http para https.

Isso tudo existe para evitar fraudes e é uma forma de impedir que o usuário use um site que na verdade não representa a origem.

**CORS - Cross-Origin Resource Sharing**

Voltando para nossa aplicação, então é impossível carregar as frases do outro servidor? Não é, mas nesse caso é preciso configurar as outras origens no servidor. Aqui o outro servidor (não a origem que já funciona com AJAX) adiciona um cabeçalho na resposta HTTP e baseado nessa resposta o navegador permitir uma requisição para *outra origem*.

O cabeçalho é bem simples e faz parte do protocolo HTTP:

Access-Control-Allow-Origin: http://localhost:3000, http://192.168.0.83:3000

Essa forma de permitir chamar uma outra origem também é chamado de **Cross-Origin Resource Sharing** ou CORS.

Repare então que isso não é uma configuração do jQuery ou AJAX em geral. Isso é algo que o servidor precisa se preocupar e adicionar na resposta HTTP. Como nosso foco aqui é jQuery não vamos dar muitos detalhes pois depende muito da linguagem e framework utilizado no lado do servidor.

No nosso caso escrevemos a aplicação servidora em node.js (JavaScript) usando o framework Express. Para habilitar CORS com Express basta utilizar:

app.use(function(req, res, next) {

res.header("Access-Control-Allow-Origin", "http://localhost:3000, http://192.168.0.83:3000");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

next();

});

Mais infos sobre Express no link abaixo e nos nossos cursos sobre node.js!

<http://enable-cors.org/server_expressjs.html>

Para colocar um slider na pagina com jquery usando o plugin slick

Para colocar a div com as fotos que irão ficar passando, é só colocar esse código htm na pagina

<div class="slider">

<img src="img/slide1.png" class="imagem-jogo">

<img src="img/slide2.png" class="imagem-jogo">

<img src="img/slide3.png" class="imagem-jogo">

</div>

Importar os css e js na pagina

<link rel="stylesheet" type="text/css" href="slick/slick.css"/>

<link rel="stylesheet" type="text/css" href="slick/slick-theme.css"/>

<script src="slick/slick.min.js"></script>

Para chamar o slider e definir algumas propriedades personalizáveis. O dot é para os pontinhos

//arquivo js/slider.js

$(function(){

$(".slider").slick( {

dots: true,

infinite: true,

speed: 300,

slidesToShow: 1,

adaptiveHeight: true

});

});

Para dar cor a seta que por padrão vem branca e deixar a altura automática

.slider {

height: auto;

}

.slick-prev:before, .slick-next:before{

color: black;

}

Biblioteca selectsize para melhorar o select da pagina <http://selectize.github.io/selectize.js/>

Baixar o plugin selectsize salvar arquivos js e css na pasta do projeto

<div class="select-usuarios">

<select class="usuarios">

<option value="Douglas">Douglas</option>

<option value="Flavio">Flavio</option>

<option value="Nico">Nico</option>

</select>

</div>

Para incorporar o plugin ao select e poder criar um option na própria view

$('.usuarios').selectize({

create: true,

sortField: 'text'

});

Esse plugin facilita dar feedback através desses pequenos balões que mostram alguma info:

<http://iamceege.github.io/tooltipster/>

Para usar basta colocar a classe tooltip e definir um titulo para que apareça na tela

<a id="botao-sync" class="btn-floating btn-large waves-effect waves-light yellow tooltip" title="Sincronizado com sucesso">

<i class="material-icons">swap\_vert</i>

</a>

Para que a mensagem não abra exatamente quando passarmos o mouse sobre o botão de sincronismo e sim quando terminar de ser executada a chamada ajax, o tooltip tem um trigger que podemos utilizar.

Para isso definimos ele como customizado ao iniciar a pagina

$(".tooltip").tooltipster({

trigger: "custom"

});

Depois colocamos no código ajax que é quando queremos que ele abra fazemos o seguinte

$.post('http://localhost:3000/placar22', dados, function(){

$(".tooltip").tooltipster("open"); //novo

}).fail(function(){

$(".tooltip").tooltipster("open").tooltipster("content", "Falha ao sincronizar");

}).always(function(){ //novo

setTimeout(function() {

$(".tooltip").tooltipster("close");

}, 1200);

})

Dessa forma ele vai abrir e depois de um tempo fechar. Caso o sincronismo não aconteça ele seta uma outra mensagem que tbm sera fechada