Tutorial para o laboratório jQuery 1

Pessoal, para ambientá-los com a programação utilizando jQuery, estou preparando um passo a passo das coisas que vocês precisam ter atenção e como organizar as atividades ao longo da resolução de problemas dessa natureza.

Fiquem tranquilos! Toda tecnologia nova traz desafios para incorporá-las. Não se cobre muito caso encontre erros durante a realização da tarefa. O objetivo dos laboratórios é testar as possibilidades e **errar**. Errar e aprender com os erros. Por isso, sintam-se a vontade de me procurar para tirar dúvidas. Consultem seus colegas também se preferirem, mas NUNCA, NUNCA copiem a resposta. Você só perde com isso.

A primeira dica para qualquer problema de programação é fazer as coisas aos poucos. Não tente resolver todo o problema de uma só vez. Chamamos esta estratégia de **baby steps** - seus primeiros passos em uma tecnologia nova não precisam ser firmes. Lembre-se que você está aprendendo, há muita coisa a ser entendida para que você execute as tarefas de maneira direta. Então devagar e sempre!

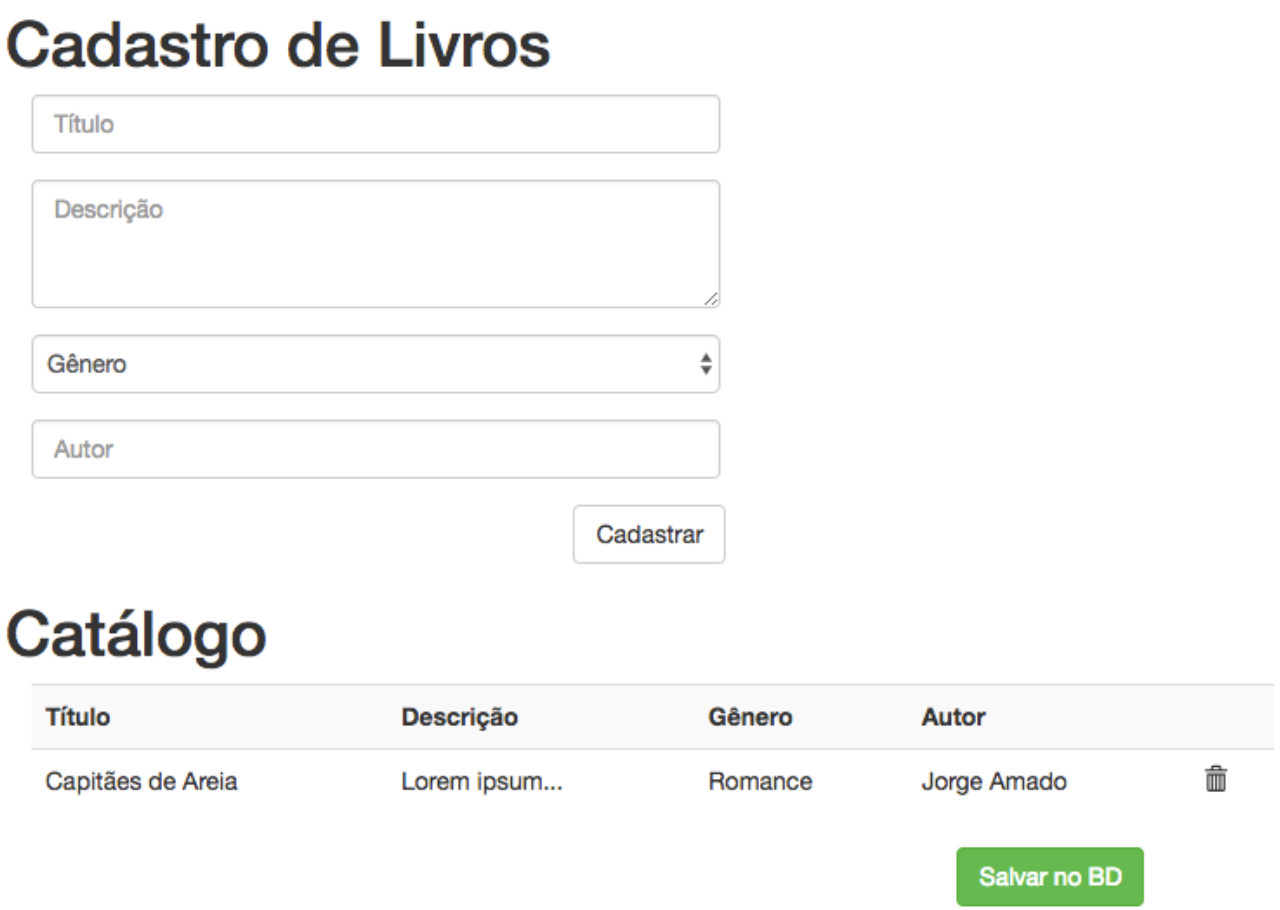
A segunda dica é: **depure seu código o tempo todo!** Suas primeiras tentativas devem estar cheias de **console.log()** para que você compreenda o que está acontecendo a cada linha de código. Obviamente, você deve apagar todas as linhas de depuração antes de submeter a atividade mas vá se cercando deste recurso na medida em que avança.

Erros vão acontecer o tempo todo. Não se frustem se não funcionar de primeira. O importante é aprender com os erros e ir corrigindo. Aprenda a buscar ajuda para seus problemas na Web. Há uma série de páginas dedicadas a problemas de programação, e acredite: a mesma situação que você está passando hoje já foi uma pedra no sapato de alguém há algum tempo. Recomendo o Stack overflow, w3schools, linhadecodigo.com.br, developer.mozilla.org etc. O que te diferencia como profissional de informática do CEFET é a capacidade de aprender, de pesquisar e executar tarefas dentro do prazo combinado, enfim, a capacidade de se virar sozinho(a). Você não é obrigado a decorar comandos ou sintaxe de tecnologias, o importante é conhecer o funcionamento delas. Iremos voltar a esse ponto ao longo do curso.

Por enquanto é só, hora de pôr a mão na massa!

1. **Faça um programa que catalogue livros em uma biblioteca. Para isto, você deverá implementar um formulário com os campos "Título", "Descrição", Gêneros pré-definidos (select box com as opções "Romance", "Drama", "Terror", "Ficção", "Técnico"), "Autor" e um botão submeter.**

- Ao clicar no botão "Cadastrar", o livro deverá ser inserido dinamicamente em uma tabela com a descrição do item abaixo do formulário.
- Na medida em que novos autores forem cadastrados, o campo de autor deve sugerir este novo autor (caixa de texto type search).
- Ao clicar na lixeira, o livro deve ser removido do catálogo
- Ao clicar no botão "Salvar no BD", deve enviar todos os objetos via post para o endereço "http://rafaelescalfoni.net/web/livros.php". Obs.: para facilitar, crie um array "acervo", e, ao inserir o livro na tabela "catálogo", insira também no array "acervo" um objeto com {"titulo": "algum titulo", "descricao": "Alguma descrição", "genero": "Algum gênero", "autor": "algum autor"}



Estratégia de resolução :

1. Criar uma estrutura estática da página para testes
2. Programar o preenchimento de novos livros no catálogo (tabela)
3. Programar a remoção de itens do catálogo pela lixeira.
4. Criar uma estrutura de dados que possa ser enviada ao servidor (formato esperado - no caso o JSON do enunciado)
5. Carregar o input hidden para quando o formulário for submetido ele estar com os dados.

Observação: Você não precisa saber como operacionalizar a estratégia quando estiver nessa fase. O importante é pensar em um conjunto de passos que façam sentido e que, por mais que você não tenha uma resposta imediata, possa ser executado. O plano é pense na solução como um todo, decomponha em partes e vá resolvendo as partes.

Parte 1 - Criação da página estática
- Caprichem na interface. Vocês já tiveram aulas de Design Web. Qualquer dúvida sobre como ajustar a tela, podem me procurar.

- Observação importante: seu botão “Salvar no BD” deve estar em um formulário:

```
<form action=”http://rafaelescalfoni.net/web/livros.php” method=”post”>
    <input type=”hidden” name=”acervo_post”>
    <input type=”submit” value=”Salvar no BD”>
</form>
```

Sobre o input hidden: nas próximas aulas, aprenderemos a fazer requisições via AJAX (requisições assíncronas). Por enquanto, nossa requisição depende de um formulário <form>.

Como vocês devem lembrar de Design Web, um <form> agrupa todos os campos de entrada que serão enviados ao serviço indicado pelo atributo **action.** Essa ação será enviada de acordo com o método definido (no caso post, que envia através do corpo da requisição os dados).

Ao apertar o botão “Salvar no BD”, o formulário irá enviar os dados contidos nele (no caso, apenas o input type=”hidden” (que não aparecerá na tela mas terá todos os valores atribuídos ao longo do programa).

Hora de fazer o primeiro teste. Aqui está o objeto de exemplo:

```
{"titulo": "algum titulo"
  , "descricao": "Alguma descrição"
  , "genero": "Algum gênero"
  , "autor": "algum autor"}
```

O servidor espera um array desses objetos aí, logo um array com esse formato:

```
[
  {"titulo": "algum titulo"
    , "descricao": "Alguma descrição"
    , "genero": "Algum gênero"
    , "autor": "algum autor"}
]
```

TESTE SEU CÓDIGO: SUBSTITUA O `<input type=”hidden” name=”acervo_post”>`
por

```
<input type=”hidden” name=”acervo_post” value=’[{"titulo": "O Sumiço da Santa", "descricao": "Lorem Ipsum", "genero": "Romance", "autor": "Jorge Amado"}]‘>
```

Salve, dê um refresh no site e aperte o Botão Salvar no BD. Ele irá apresentar esses dados no servidor.

Relembrando as aulas sobre estruturas de dados em JavaScript, o primeiro elemento aí de cima é um objeto JavaScript. Um objeto é uma coleção que possui um conjunto de chaves e valores associados. Se chamarmos a estrutura aí de cima de **livro**, podemos acessar essas propriedades da seguinte forma:

```
livro.titulo //retornará “algum titulo”
livro.descricao // retornara “Alguma descrição” etc
```

Como criar objetos? É bem simples! Basta atribuir {} a uma variável:

```
let livro = {}; //criado um objeto vazio chamado livro
livro.titulo = “Tenda dos Milagres”; // livro passa a ser {titulo:”Tenda dos Milagres”};
livro.autor=”Jorge Amado”;  // livro passa a ser {titulo:”Tenda dos Milagres”, autor: “Jorge Amado”}
```

Uma outra forma mais robusta de criar objetos é através de uma função construtora:

```
function Livro (titulo, desc, genero, autor){
      this.titulo = titulo;
      this.descricao = desc;
      this.genero = genero;
      this.autor = autor;
}
```

para usar a função, basta usar a palavra reservada new seguida da chamada da função:

```
var meuLivro = new Livro(“O sumiço da santa”, “Lorem Ipsum”, “Romance”, “Jorge Amado”);

console.log(meuLivro)
>> {"titulo": "O Sumiço da Santa", "descricao": "Lorem Ipsum", "genero": "Romance", "autor": "Jorge Amado"}
```

Já um Array é uma coleção cujo os índices são numéricos (não são chaves). É nativo do JavaScript, não precisa ser criado.

Formas de criar um array:

```
var turma = []; // ou turma = new Array();
turma.push("Ana");
turma.push("João");
```

ou

```
var turma = ["Ana", "João"];
```

Vamos precisar de arrays e objetos mais adiante.

2. Programar o preenchimento de novos livros no catálogo (tabela)

Ao clicar no botão cadastrar (método click()) devemos recuperar os valores das caixas de entrada e criar uma linha na tabela#acervo.

Programe as atividades 1, 2, 3 e 4. Teste cada uma delas utilizando o console.log.

```
$("#add_acervo").click(function(){
// 1. recuperar os valores digitados nas caixas de entrada (função val())

// 2. criar um TR e adicionar uma classe livro (consultar o item 4 para entender porque uma classe)
// dentro deste TR, criar 5 TD - para título, descrição, gênero, autor e lixeira.
// 3. adicionar esse TR à tabela
// 4. apagar as caixas de texto
});
```

3. Programar a remoção de itens do catálogo pela lixeira.

Também precisamos programar o excluir linha. Como vimos, como as linhas não existem quando o dom é carregado (document.ready) então não podemos usar diretamente métodos como click(). Neste caso, precisamos nos referenciar a um elemento que esteja presente na página quando o documento for carregado. O método on() nos permite a referenciar filhos de um elemento e programar eventos para eles, mesmo que eles ainda não existam quando a página for carregada:

```
$("#acervo").on("click", "img", function(){
      // para o elemento com ID="acervo", considere que em algum momento ele terá um descendente
      // img e que queremos programar o evento "click" dele. Execute esta função quando isso ocorrer.
});
```

4. Criar uma estrutura de dados que possa ser enviada ao servidor (formato esperado - no caso o JSON do enunciado)

Os dados dos livros serão enviados através do formulário, pelo input name="acervo_post".

O primeiro passo é organizar a lista de elementos a serem enviados.
Duas estratégias são possíveis:

1. Ao inserir um item no catálogo, criar o objeto livro e adicionar na array. Neste caso, seu programa de exclusão deve remover o item do array quando for excluído.
2. Quando o usuário clicar no Salvar no BD, recuperar os livros e salvá-los numa lista para então carrega-los no input hidden.

Seguem os passos para a segunda opção:

programando o

```
$("input[type=submit]").click(function(){
      // ao clicar no botão "Salvar no BD"
      // recuperar todos os elementos tr da tabela#acervo que possuam tds
      // você precisará recorrer aos recursos para percorrer a árvore DOM
      // para encontrar os valores de cada um dos TDs - título, descrição,
      // gênero e autor

      // no meu teste, optei por adicionar uma classe nas tr's referentes à livros
      // (a primeira tr tem o cabeçalho da tabela e precisa ser descartada). Criei
      // uma classe livro.
```

Se você também adicionou essa classe .livro, irá recuperar um array de trs utilizando um simples

```
var linhaLivros = $(".livro");
      // array de:
      // <tr>
      //    <td>titulo</td>
      //    <td>desc</td>
      //    <td>gênero</td>
      //    <td>autor</td>
      //    <td><img src="lixeira.png"></td>
      // </tr>
```

Teste!!! use um **console.log(linhaLivros)** e veja o resultado.

o passo seguinte é iterar esse array linhaLivros pegando cada tr e buscando nos seus nós filhos (<td>) os seus elementos textuais

Existe um comando de iteração poderosíssimo no jQuery - each(). Ele funciona assim:

```
$.each(umArrayQualquer, function(indice, objeto){ // não esqueça do "." entre $ e each
      // seu código aqui
      // a cada iteração em umArrayQualquer,
      // ele atribui a índice a posição do array
      // (use o nome que preferir e ao objeto = umArrayQualquer[índice]
      // sem que tenhamos que nos preocupar com o tamanho do array.

});
```

No nosso caso, ficaria algo do tipo:

```
$.each($(".livro"), function(idx, linha){
      let tituloLinha = $(linha).children(selector para o primeiro filho).text();
      let descricaoLinha = $(linha).children(selector para o segundo filho).text();
      let generoLinha = $(linha).children(selector para o terceiro filho).text();
      let autorLinha = $(linha).children(selector para o quarto filho).text();
```

//substitua **seletor para xxxxx filho** por um seletor CSS que referencie o filho.

```
      let livro = new Livro(tituloLinha, descricaoLinha, generoLinha, autorLinha);
      acervoArray.push(livro);
});

// após iterar e carregar o array acervoArray, basta enviar para o input hidden
// mas antes, é preciso transformar o array em string para que ele possa ser enviado via http post:
```

```
var acervoStr = JSON.stringify(acervoArray);

//JSON é um objeto JavaScript para manipulações json
//agora é só mudar o valor do acervo_post para acervoStr. ;-) e está pronto!!!

}); //fim do input[type=submit]
```

- Na medida em que novos autores forem cadastrados, o campo de autor deve sugerir este novo autor (caixa de texto type search).

Para esta funcionalidade, você deve criar um <datalist> que gera um autocompletar. Funciona assim: as opções existentes no datalist serão sugeridas ao usuário:

referência: https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/datalist

```
<input list="browsers" />
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Internet Explorer">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Então, sua lista de autores deve estar vinculada a um datalist. A cada novo autor, você deve inserí-lo como um novo option no datalist:

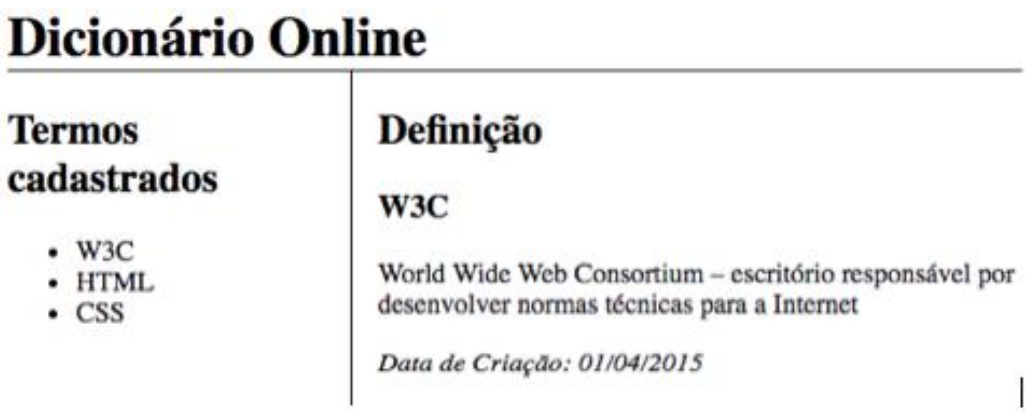
```
<input list="autores" id="autor">
<datalist id="autores">
<!-- começa vazio, mas na medida em que novos livros vão sendo incluídos, seus autores vão sendo criados como novas options -->
</datalist>
```

2. Um site disponibiliza um glossário online onde pessoas do mundo todo podem cadastrar e consultar termos. Tais termos são armazenados em uma coleção com formato similar ao trecho abaixo.

Implemente uma função que, ao clicar no item de lista dos termos cadastrados, exiba a definição na seção à direita.

Para carregar seu dicionário, utilize o array abaixo:

```
var glossario = [{"id":1
        , "termo":"W3C"
        ,  "definicao": "World Wide Web Consortium – escritório responsável por desenvolver normas técnicas para a Internet"
        , "autor": "José Borges"
        , "dataCriacao": "2015-04-01"}
      , {"id":2
        , "termo":"HTML"
        , "definicao": "HyperText Markup Language – linguagem de marcação utilizada para estruturar páginas web"
        , "autor": "Pedro Silva"
        , "dataCriacao": "2017-05-12"}
      , {"id":3, "termo": "CSS"
        , "definicao": "Cascade Style Sheet – Folhas de estilo utilizadas para configurar a visualização de  páginas web"
        , "autor": "Maria Machado"
        , "dataCriacao": "2018-10-11"}
      ];
```



Nesta atividade, a página deve ser carregada dinamicamente a partir de um array com definições. Quando o usuário clicar em um item, a definição deve aparecer na seção da direita. É um exercício simples, que usa uma estrutura de dados bastante limitada (um array com 3 itens). Mas poderia ser algo mais sofisticado, buscando itens em uma base complexa, como por exemplo a da Wikipedia. O funcionamento seria o mesmo.

Estratégia de resolução :

1. Criar uma estrutura estática da página para testes
2. Programar o preenchimento dinâmico da lista não ordenada da esquerda. Cada item deve guardar seu id para facilitar a recuperação do objeto no array.
3. Programar o evento de clique nos itens de lista para que carregue a seção da direita com o título, definição, e data de criação.

1. Estrutura estática

Caprichem na página. Nosso esquema ficou bastante pobre...

2. Preenchimento dinâmico

Com base na estrutura estática, é hora de iterar sobre o array e inserir cada um dos itens de lista presente em glossário.

Atenção ao fato de que os itens de lista não existem quando o documento é carregado, então não poderemos programar no passo 3 $(“li”).click() como já vimos no exercício anterior.

3. Programar evento clique

Ao clicar em um item, identificar o id do li que sofreu a ação e buscar no array o item correspondente. Daí basta carregar os valores dos atributos na seção da direita, de acordo com a sua estrutura de html.