



jquery



Este material é exclusivo do canal (youtube): [canalfessorbruno](https://www.youtube.com/c/canalfessorbruno) e do site www.cfbcursos.com.br
não pode ser distribuído ou comercializado por outra fonte.

Apostila: versão 1.0

21/07/2016

www.youtube.com/canalfessorbruno

www.cfbcursos.com.br

canalfessorbruno@gmail.com

www.facebook.com/canalfessorbruno

twitter: @fessorBruno

Sumário

Introdução.....	7
Baixando a biblioteca jQuery	7
Anexando a biblioteca jQuery em suas páginas	9
Entendendo o DOM	10
Sintaxe e Seletores jQuery	11
Primeira demonstração de jQuery.....	13
Executando código jQuery	15
Evento ready	17
Mais eventos	18
click	18
dblclick	19
mousedown e mouseup.....	19
mouseenter e mouseleave	19
mousemove	20
hover	20
keypress	21
keydown e keyup	22
submit	22
change.....	22
focus e blur.....	23
focusin e focusout.....	23
select.....	24
ready	24
resize	25
scroll.....	25
on	25
off.....	27
trigger.....	28
event.pageX e event.pageY.....	28
event.target	28
event.which.....	29
event.stopPropagation	29
event.type	30
Referenciando elementos, como o jQuery busca e encontra os elementos.....	30
Referências a elementos pai (parent).....	31

parent()	31
parents()	32
parentsUntil()	32
Referências a descendentes	34
children()	34
find()	34
Referências aos elementos irmãos (Siblings)	36
next()	37
nextAll()	38
nextUntil()	39
prev(), prevAll() e prevUntil()	40
Filtrando elementos (filtering)	40
first() e last()	41
eq()	41
filter() e not()	42
Manipulando Elementos e atributos	43
text()	43
val()	44
html()	44
attr()	45
Efeitos	46
hide/show	46
toggle	47
animate	48
Fila de animação	49
clearQueue	51
toggle	52
delay	52
fadeIn e fadeOut	53
fadeTo	53
fadeToggle	54
finish	54
queue	55
slideUp e slideDown	55
slideToggle	56
stop	56

Métodos para adicionar novos conteúdos	57
append()	57
appendTo()	58
prepend()	58
prependTo()	59
before() e after()	59
Métodos para remover elementos	59
remove()	59
empty()	60
css()	61
Multiplas propriedades css	62
Adicionando ou removendo classes CSS	62
addClass() e removeClass()	62
toggleClass	63
Mais métodos	63
clone()	63
detach()	64
empty()	64
hasClass()	64
position()	65
offset()	65
offsetParent()	66
removeAttr()	66
removeClass()	67
replaceAll()	67
replaceWith()	67
scrollLeft() e scrollTop()	68
unwrap()	68
wrapAll()	69
wrapInner()	69
each()	70
get()	70
index()	70
noConflict()	71
length()	71
toArray()	72

jquery	72
Métodos para definir ou obter alturas e larguras dos elementos.....	72
Considerações finais	75

Introdução

jQuery é uma biblioteca javascript bastante interessante com o objetivo de facilitar o uso em determinados recursos em javascript, torna mais simples a criação de animações, efeitos, manipulação de eventos, seleção de elementos.

Os principais objetivos de jQuery são redução de código javascript, utilização de plug-ins prontos de diversos desenvolvedores encontrados na web, resolver alguns problemas de incompatibilidade entre browsers.

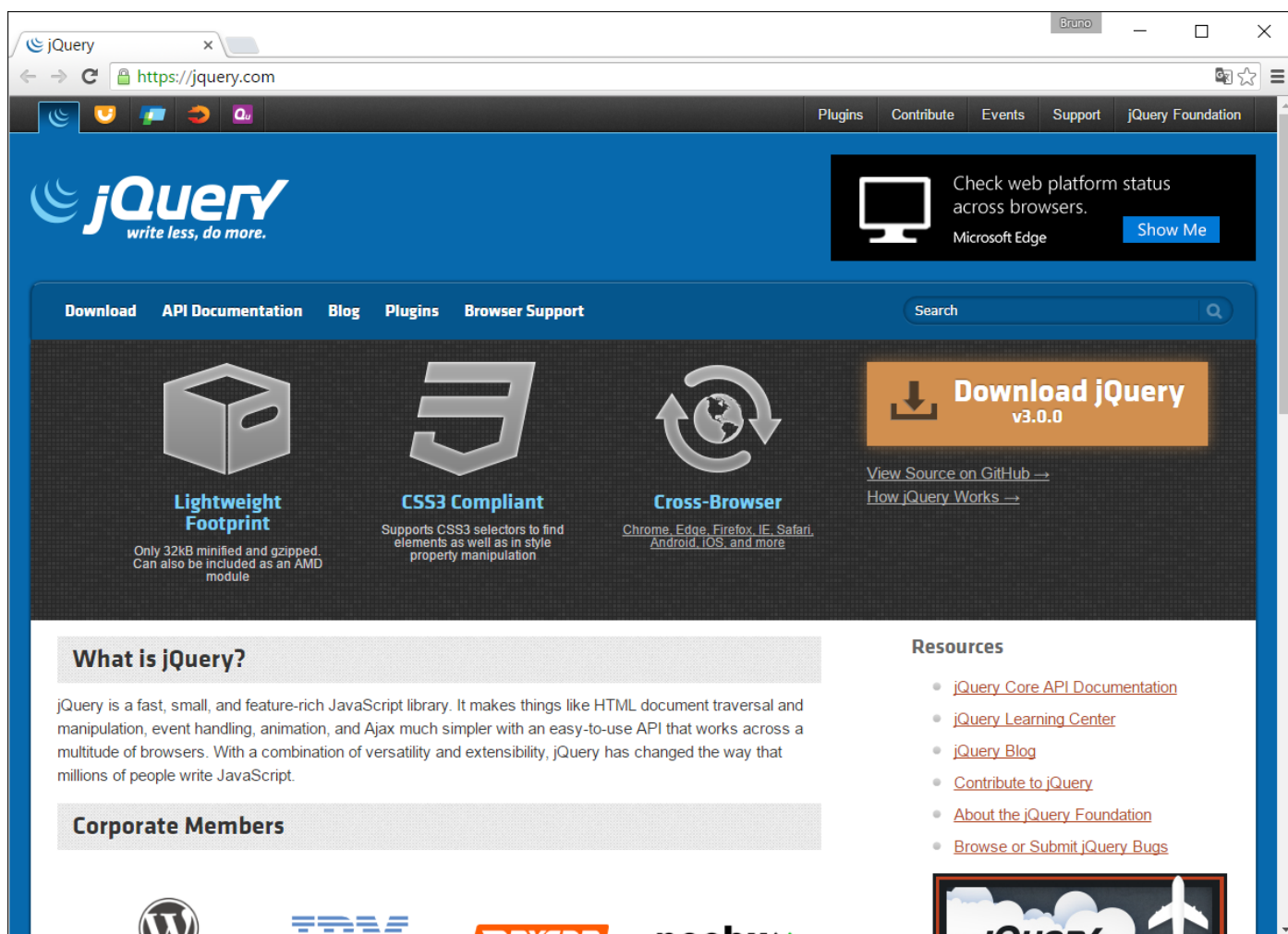
A princípio jQuery parece algo complicado de se usar, mas aos poucos vamos nos acostumando com a sintaxe e tudo vai se tornando simples.

Você consegue informações adicionais oficiais sobre a linguagem no site <https://jquery.com/>

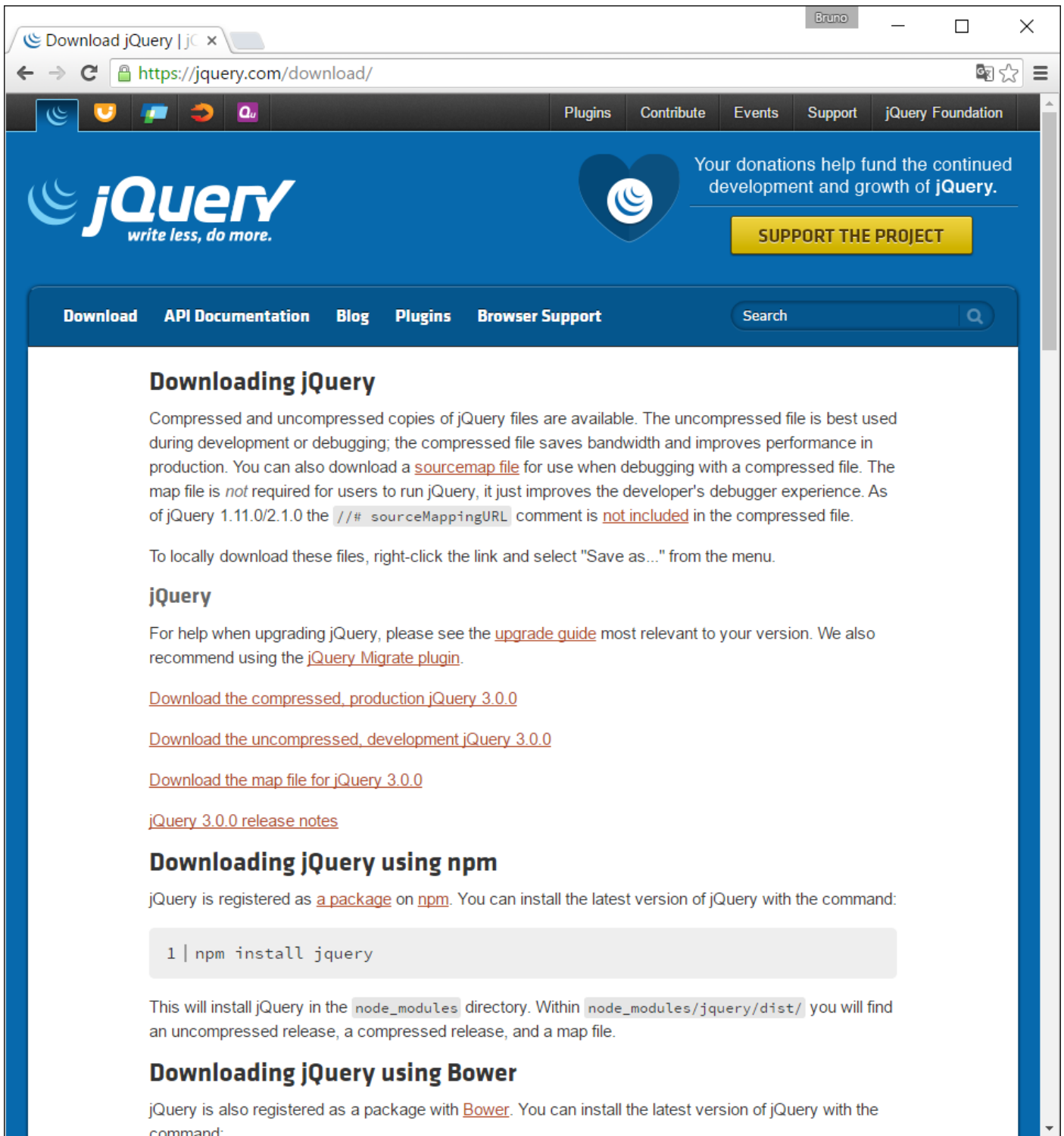
Então vamos ao trabalho aprender a usar a biblioteca javascript mais usada no mundo.

Baixando a biblioteca jQuery

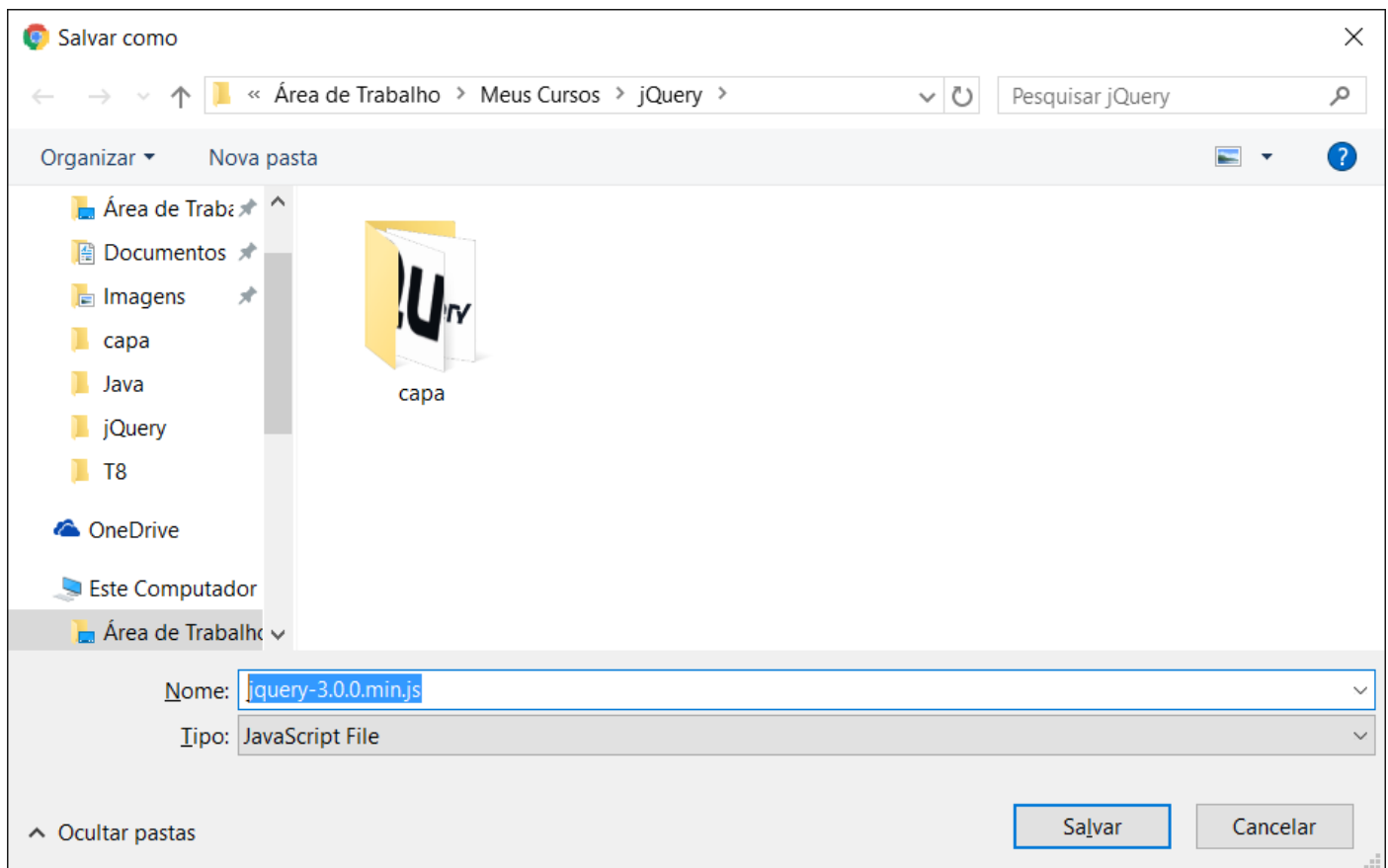
Acesse o site <https://jquery.com/> e clique no botão laranja “Download jQuery”.



Ao clicar no botão para download a janela a seguir será mostrada, clique no link “Download the compressed, production jQuery 3.00”.

A screenshot of a web browser displaying the jQuery download page. The browser's address bar shows 'https://jquery.com/download/'. The page has a blue header with the jQuery logo and the tagline 'write less, do more.'. To the right of the logo is a heart icon and the text 'Your donations help fund the continued development and growth of jQuery.' with a yellow button labeled 'SUPPORT THE PROJECT'. Below the header is a navigation bar with links: 'Download', 'API Documentation', 'Blog', 'Plugins', 'Browser Support', and a search bar. The main content area has a section titled 'Downloading jQuery' with a paragraph explaining that compressed and uncompressed copies are available, and that the compressed file is best for production. It also mentions a 'sourcemap file' for debugging. Below this is a paragraph about locally downloading files. Then, there's a section titled 'jQuery' with a paragraph about upgrading and a link to the 'jQuery Migrate plugin'. This is followed by three links: 'Download the compressed, production jQuery 3.0.0', 'Download the uncompressed, development jQuery 3.0.0', and 'Download the map file for jQuery 3.0.0'. Then, there's a link to 'jQuery 3.0.0 release notes'. Next is a section titled 'Downloading jQuery using npm' with a paragraph stating that jQuery is registered as a package on npm and can be installed with the command 'npm install jquery'. Below this is a code block containing the command '1 | npm install jquery'. Then, there's a paragraph explaining that this will install jQuery in the 'node_modules' directory and that within 'node_modules/jquery/dist/' there will be an uncompressed release, a compressed release, and a map file. Finally, there's a section titled 'Downloading jQuery using Bower' with a paragraph stating that jQuery is registered as a package with Bower and can be installed with a command (partially visible).

O download do arquivo “jquery-3.0.0.min.js” irá iniciar, baixe este arquivo para a pasta onde irá criar sua(s) página(s) que irá(ão) usar jQuery.



Ótimo, biblioteca já baixada.

Anexando a biblioteca jQuery em suas páginas

É muito simples anexar a biblioté jQuery em nossas página, podemos inclusive anexar a biblioteca que baixamos ou a biblioteca online.

Para anexar a biblioteca que baixamos basta usar a tag `<script>` com o parâmetro "scr" apontando para o arquivo.

Veja a seguir uma página com código HTML básico onde anexamos a biblioteca jQuery que baixamos.

```
<html>
<head>
  <script src="jquery-3.0.0.min.js"></script>
</head>
<body>

</body>
</html>
```

Caso você não queira baixar e prefira usar a biblioteca online basta usar um do paths a seguir:

Google CDN → <https://ajax.googleapis.com/ajax/libs/jquery/3.0.0/jquery.min.js>

Microsoft CDN → <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.0.0.min.js>

*CDN (Content Delivery Network).

Veja o exemplo do código com o uso da biblioteca online.

```
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
```

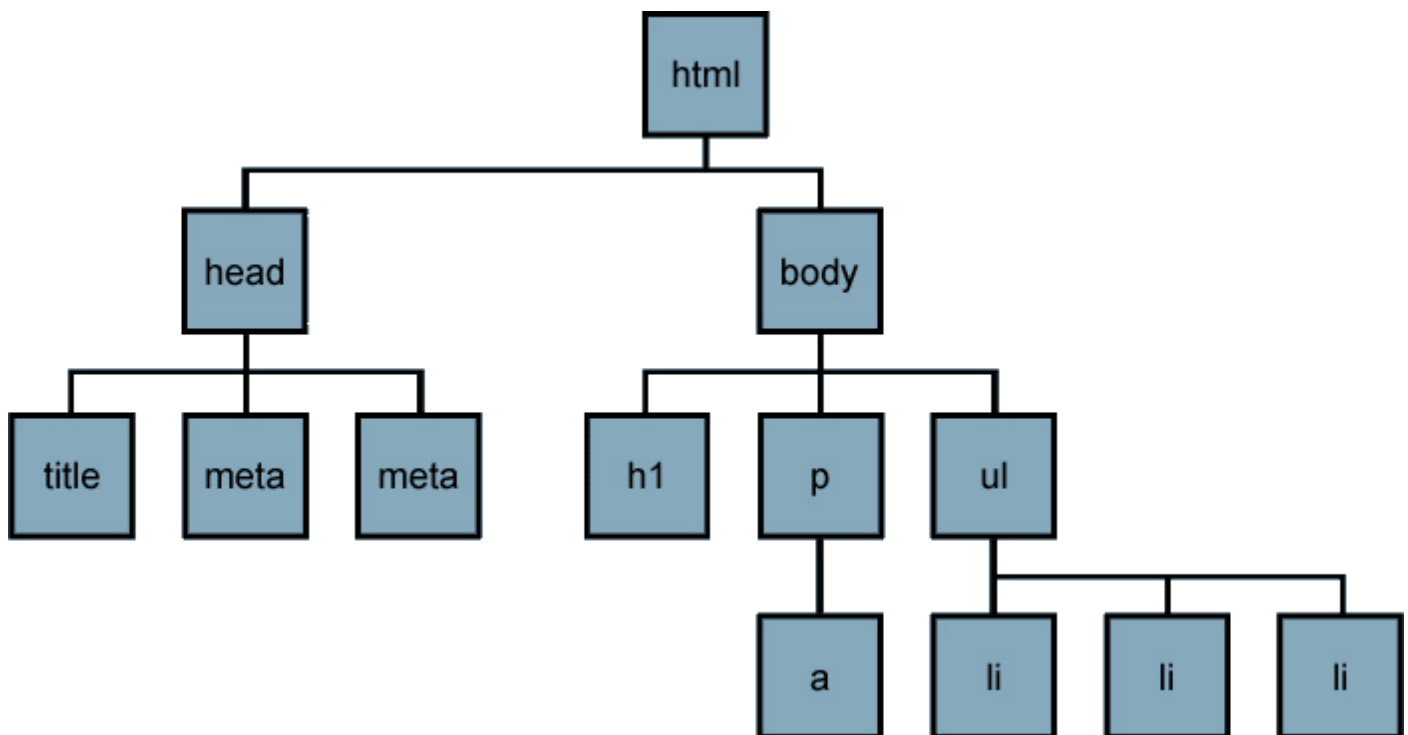
```
<body>  
</body>  
</html>
```

OBS: Antes de prosseguir você precisa saber trabalhar com HTML, CSS e Javascript, portanto se você não tem estes conhecimentos básicos necessários sugiro que assista as aulas no canal e baixe as apostilas pelo site cfbcursos.com.br.

Entendendo o DOM

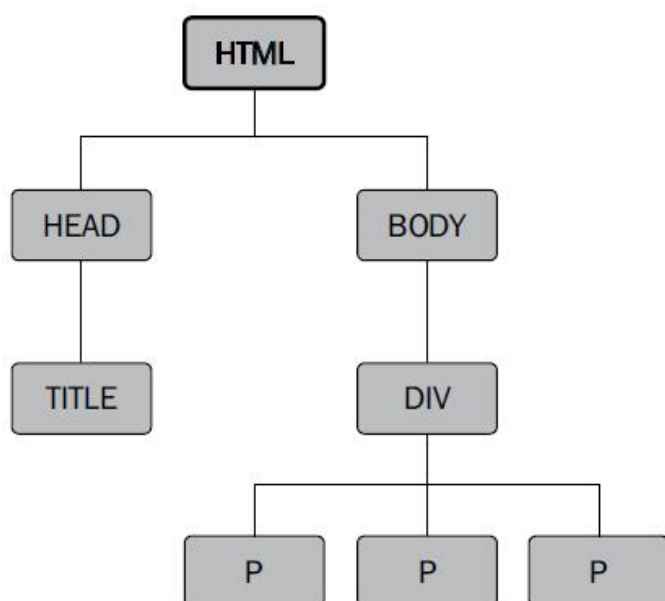
Um dos grandes aspectos do jQuery é sua poderosa capacidade de selecionar os elementos no DOM que serão manipulados de forma relativamente fácil e direta. Basicamente DOM é uma plataforma que representa como as tags em HTML, XHTML e XML são organizadas e lidas pelo browser, estas marcações ou tags são organizadas em uma “árvore” que manipulamos via API, que é exatamente o que é feito quando usamos javascript para alterar as funcionalidades de uma página, de forma bem simples pode ser compreendido como uma interface entre o javascript e as páginas web.

A ilustração a seguir mostra um exemplo de uma árvore DOM.



Veja o exemplo do código a seguir e logo após a ilustração de sua árvore DOM.

```
<html>  
<head>  
  <title>Canal Fessor Bruno</title>  
</head>  
<body>  
  <div>  
    <p>Canal Fessor Bruno</p>  
    <p>Curso de jQuery</p>  
    <p>cfbcursos.com.br</p>  
  </div>  
</body>  
</html>
```



Sintaxe e Seletores jQuery

jQuery tem uma sintaxe bem específica e bem definida, fácil de identificar e se acostumar.

Veja a sintaxe básica: **\$(seletor).ação()**

Onde:

\$ = jQuery

seletor = Elemento que será manipulado

ação = Uma ação a ser executada com o elemento (Evento, Função)

É muito simples a forma de selecionar os elementos pelo jQuery, podemos selecionar tags HTML, propriedades CSS e elementos específicos através do ID, observe a tabela a seguir.

Tipo de seletor	CSS	jQuery	Descrição
<tag>	p{ }	\$('p')	Seleciona todas as tags <p> do documento.
ID	#id_do_elemento{ }	\$('#id_do_elemento')	Selecione um elemento específico no documento, selecione pelo ID.
Classe	.nome_da_classe{ }	\$('.nome_da_classe')	Selecione todos os elementos no documento que usem a classe indicada.

Podemos entender o operador \$ como uma forma simples de dizer jQuery, assim vamos considerar que uma expressão jQuery será formada de duas partes, o que será manipulado (selecionado) e como será manipulado, na tabela acima vimos três formas de informar o que será manipulado, selecionar elementos por tag \$('tag'), selecionar elementos por ID \$('ID') e selecionar elementos por classe CSS \$('classe').

Assim as duas linhas de códigos a seguir são exatamente iguais, portanto geram o mesmo resultado final.

```

$("p").css("background-color", "#F88");
jQuery("p").css("background-color", "#F88");
  
```

A tabela a seguir mostra mais alguns seletores usados em jQuery.

Sintaxe	Descrição
<code>\$("*")</code>	Seleciona todos os elementos
<code>\$(this)</code>	Seleciona o elemento atual
<code>\$("p.cfb")</code>	Seleciona todos os elementos <code><p></code> que usem a classe <code>cfb</code>
<code>\$("p.first")</code>	Seleciona o primeiro elemento <code><p></code>
<code>\$("p.last")</code>	Seleciona o último elemento <code><p></code>
<code>\$("table tr:first")</code>	Seleciona o primeiro elemento <code><tr></code> do primeiro elemento <code><table></code>
<code>\$("ul li:first-child")</code>	Seleciona o primeiro elemento <code></code> de qualquer elemento <code></code>
<code>\$("p:first-of-type")</code>	Seleciona todos os elementos <code><p></code> que sejam os primeiros no seu grupo
<code>\$("p:last-of-type")</code>	Seleciona todos os elementos <code><p></code> que sejam os últimos no seu grupo
<code>\$("p:nth-child(2)")</code>	Seleciona todos os elementos <code><p></code> que sejam os segundos no seu grupo
<code>\$("p:nth-last-child(2)")</code>	Seleciona todos os elementos <code><p></code> que sejam os penúltimos no seu grupo
<code>\$("p:only-child")</code>	Seleciona todos os elementos <code><p></code> que sejam únicos em seu grupo
<code>\$("[href]")</code>	Seleciona todos os elementos que usam o atributo <code>href</code>
<code>\$("[href]='index.html'")</code>	Seleciona todos os elementos que tenham o atributo <code>href</code> com valor <code>index.html</code>
<code>\$("a[target='_blank']")</code>	Seleciona todos os elementos <code><a></code> com o atributo <code>target</code> com o valor <code>_blank</code>
<code>\$("a[target!='_blank']")</code>	Seleciona todos os elementos <code><a></code> com o atributo <code>target</code> diferente do valor <code>_blank</code>
<code>\$(":button")</code>	Seleciona todos os elementos <code><button></code> e <code><input></code> do tipo <code>button</code>
<code>\$("input:text")</code>	Seleciona todos os elementos <code><input></code> que são do tipo <code>text</code>
<code>\$(".cfb")</code>	Seleciona todos os elementos que usam a classe <code>.cfb</code>
<code>\$(".cfb,.curso")</code>	Seleciona todos os elementos que usam a classe <code>.cfb</code> ou a classe <code>.curso</code>
<code>\$("h1,h2,h3")</code>	Seleciona todos os elementos <code>h1</code> , <code>h2</code> e <code>h3</code>
<code>\$("tr:even")</code>	Seleciona todos os mesmos elementos <code><tr></code>
<code>\$("tr:odd")</code>	Seleciona todos os elementos <code><tr></code> diferentes
<code>\$("div > span")</code>	Seleciona todos os elementos <code>span</code> que estão dentro de algum <code><div></code>
<code>\$("div p")</code>	Seleciona todos os elementos <code>p</code> que são descendentes de um elemento <code><div></code>
<code>\$("div + p")</code>	Seleciona todos os elementos <code>p</code> que vêm logo após um elemento <code><div></code> , somente o primeiro do grupo que está abaixo de uma <code><div></code>
<code>\$("div ~ p")</code>	Seleciona todos os elementos <code><p></code> que vêm logo abaixo de um elemento <code><div></code>
<code>\$("p:eq(1)")</code>	Seleciona o segundo elemento <code><p></code> , lembrando que o índice inicia em zero
<code>\$("p:lt(3)")</code>	Seleciona os 3 primeiros elementos <code><p></code>
<code>\$("p:gt(3)")</code>	Seleciona todos os elementos <code><p></code> , menos os 3 primeiros
<code>\$("p:not(.cfb)")</code>	Seleciona todos os elementos <code><p></code> que não usem a classe <code>.cfb</code>
<code>\$(":header")</code>	Seleciona todos os elementos <code>h1</code> , <code>h2</code> , <code>h3</code> , <code>h4</code> , <code>h5</code> e <code>h6</code>
<code>\$(":animated")</code>	Seleciona todos os elementos animados, que usam o método <code>animate</code>
<code>\$(":focus")</code>	Seleciona o elemento que está em foco
<code>\$(":contains('cfb')")</code>	Seleciona todos os elementos que contenham o texto <code>cfb</code>
<code>\$("div:has(p)")</code>	Seleciona todos os elementos <code><div></code> que contenham elementos <code><p></code>
<code>\$(":empty")</code>	Seleciona todos os elementos que são vazios (<code>empty</code>)
<code>\$(":parent")</code>	Seleciona todos os elementos que são parentes de outro elemento, que estão dentro de outro elemento
<code>\$("p:hidden")</code>	Seleciona todos os elementos <code><p></code> que estejam ocultos (<code>hidden</code>)
<code>\$("p:visible")</code>	Seleciona todos os elementos <code><p></code> que estejam visíveis (<code>visible</code>)
<code>\$(":root")</code>	Seleciona o elemento raiz corrente
<code>\$("[href\$='.org']")</code>	Seleciona todos os elementos que tenham o atributo <code>href</code> com valor que termine com <code>.org</code>

Primeira demonstração de jQuery

Vamos partir para nossa primeira rotina em jQuery, não se assuste com os códigos mostrados, este primeiro exemplo servirá para mostrar de forma geral como jQuery funciona e para podermos comparar javascript convencional de jQuery em uma aplicação simples.

Vamos criar uma pequena calculadora de soma, onde iremos somar os valores de dois campos input de textos.

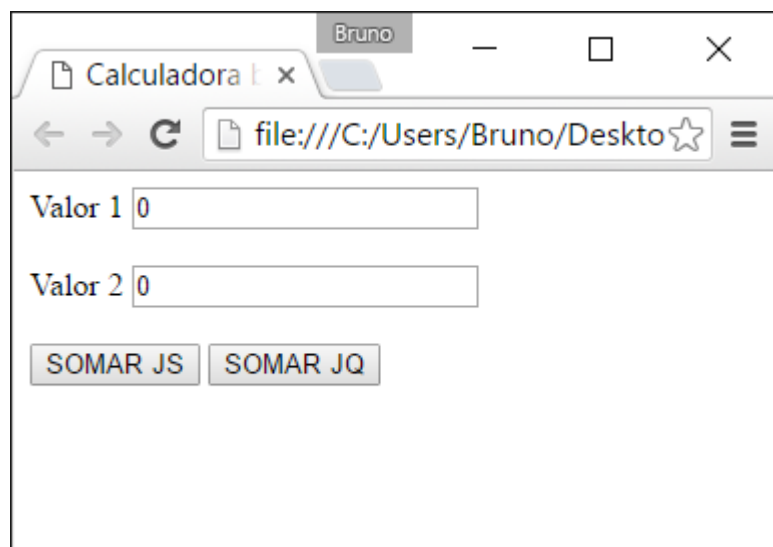
Primeiramente vamos ao código HTML.

```
<html>
<head>
  <title>Calculadora básica de soma</title>
</head>
<body>

  <form id="f_calc">
    <label>Valor 1</label>
    <input type="text" id="f_v1" value="0"/><br/><br/>
    <label>Valor 2</label>
    <input type="text" id="f_v2" value="0"/><br/><br/>
    <input type="button" id="btSomaJS" value="SOMAR JS"/>
    <input type="button" id="btSomaJQ" value="SOMAR JQ"/>
  </form>
  <div id="res"></div>

</body>
</html>
```

Com este código HTML obtemos o seguinte resultado.



Veja que temos dois botões o da esquerda irá realizar a soma pela rotina javascript convencional e o da esquerda pela rotina em jQuery.

Vamos ao javascript convencional destacado em vermelho.

```
<html>
<head>
  <title>Calculadora básica de soma</title>
  <script>
    document.addEventListener("load",function(){
      document.getElementById("btSomaJS").addEventListener("click",function(){
        var v1=document.getElementById("f_v1").value;
        var v2=document.getElementById("f_v2").value;
        var soma=parseInt(v1)+parseInt(v2);
        document.getElementById("res").innerHTML=soma;
      });
    });
  </script>
</head>
<body>
```

```

</script>
</head>
<body>

    <form id="f_calc">
        <label>Valor 1</label>
        <input type="text" id="f_v1" value="0"/><br/><br/>
        <label>Valor 2</label>
        <input type="text" id="f_v2" value="0"/><br/><br/>
        <input type="button" id="btSomaJS" value="SOMAR JS"/>
        <input type="button" id="btSomaJQ" value="SOMAR JQ"/>
    </form>
    <div id="res"></div>

</body>
</html>

```

É uma rotina simples que adiciona um evento de clique ao botão com ID “btSomaJS” com a seguinte rotina.

Cria a variável “v1” e adiciona o valor inserido no campo de texto com ID “f_v1”.

```
var v1=document.getElementById("f_v1").value;
```

Cria a variável “v2” e adiciona o valor inserido no campo de texto com ID “f_v2”.

```
var v2=document.getElementById("f_v2").value;
```

Cria a variável soma e adiciona a soma das variáveis v1 e v2, note que foi necessário o uso da função “parseInt” isso porque o valor retornado dos campos de texto são do tipo texto “string”, tornando necessário a conversão para numeral inteiro “int”.

```
var soma=parseInt(v1)+parseInt(v2);
```

Por último, inserimos o resultado como um texto dentro da div com ID “res”.

```
document.getElementById("res").innerHTML=soma;
```

Agora vamos criar uma rotina em jQuery que faça o mesmo trabalho da rotina anterior.

```

<html>
<head>
    <title>Calculadora básica de soma</title>
    <script src="jquery-3.0.0.min.js"></script>
    <script>
        document.addEventListener("load",function(){
            document.getElementById("btSomaJS").addEventListener("click",function(){
                var v1=document.getElementById("f_v1").value;
                var v2=document.getElementById("f_v2").value;
                var soma=parseInt(v1)+parseInt(v2);
                document.getElementById("res").innerHTML=soma;
            });
        });

        $(document).ready(function(){
            $("#btSomaJQ").click(function(){
                var v1=parseInt($("#f_v1").val());
                var v2=parseInt($("#f_v2").val());
                var soma=v1+v2;
                $("#res").text(soma);
            });
        });
    </script>
</head>
<body>

    <form id="f_calc">
        <label>Valor 1</label>
        <input type="text" id="f_v1" value="0"/><br/><br/>
        <label>Valor 2</label>
        <input type="text" id="f_v2" value="0"/><br/><br/>
        <input type="button" id="btSomaJS" value="SOMAR JS"/>
        <input type="button" id="btSomaJQ" value="SOMAR JQ"/>
    </form>
    <div id="res"></div>

```

```
</body>
</html>
```

Note que a sequência de código é a mesma.

Preparamos os eventos.

```
$(document).ready(function() {
    $("#btSomaJQ").click(function() {
```

Criamos a variável “v1” e adicionamos o valor inserido no campo de texto com ID “f_v1”.

```
var v1=$("#f_v1").val();
```

Criamos a variável “v2” e adicionamos o valor inserido no campo de texto com ID “f_v2”.

```
var v2=$("#f_v2").val();
```

Criamos a variável soma e adicionamos a soma das variáveis v1 e v2, também convertidos para numerais inteiros.

```
var soma= parseInt(v1)+ parseInt(v2);
```

Assim como no javascript convencional, inserimos o resultado como um texto dentro da div com ID “res”.

```
$("#res").text(soma);
```

Esta simples tarefa tem o objetivo de comparar jQuery com javascript convencional, não se preocupe se você não entendeu algumas partes do código, eu vou explicar tudo isso ao longo do curso.

Executando código jQuery

Parece muito lógico que os comandos serão executados a medida que vão sendo lidos pelo browser, e realmente é, na prática é assim que funciona, mas devemos observar alguns detalhes importantes que podem nos confundir e fazer parecer que nosso código possui algum erro por não ter gerado nenhum resultado.

Veja o exemplo a seguir.

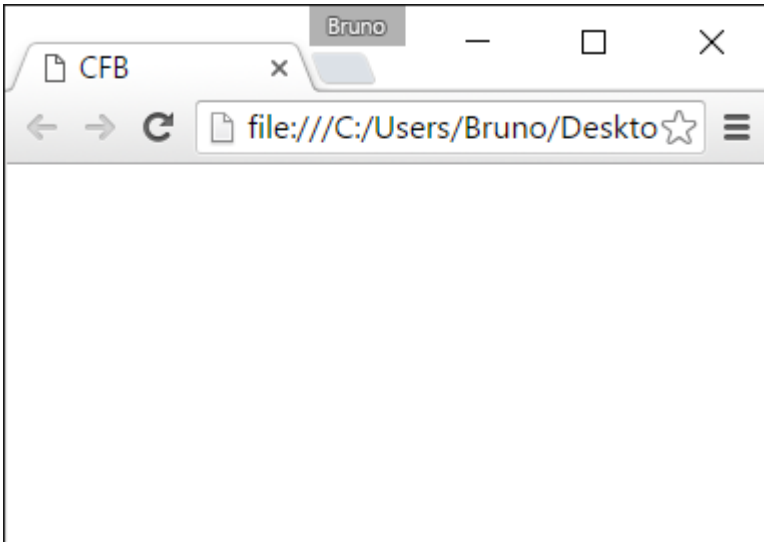
```
<html>
<head>
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js"></script>
    <script>
        $("#p1").text("Canal Fessor Bruno");
    </script>
</head>
<body>

    <p id="p1"></p>

</body>
</html>
```

É um código simples que adiciona um texto ao elemento com id “p1” no caso um elemento <p>.

Mas veja o resultado deste código.



Onde está o texto “Canal Fessor Bruno”? Existe um erro em nosso código?

Na verdade nosso código não está errado, só está incompleto para o resultado que desejamos, a rotina jQuery simplesmente não encontrou o elemento com ID “p1”. Como assim? Este elemento está na página!

Sim está! Mas o que acontece é que o javascript foi executado primeiro e quando foi executado o elemento com id “p1” ainda não havia sido criado.

Uma forma de resolver este problema seria inserindo o código jQuery após o elemento “p1”, veja.

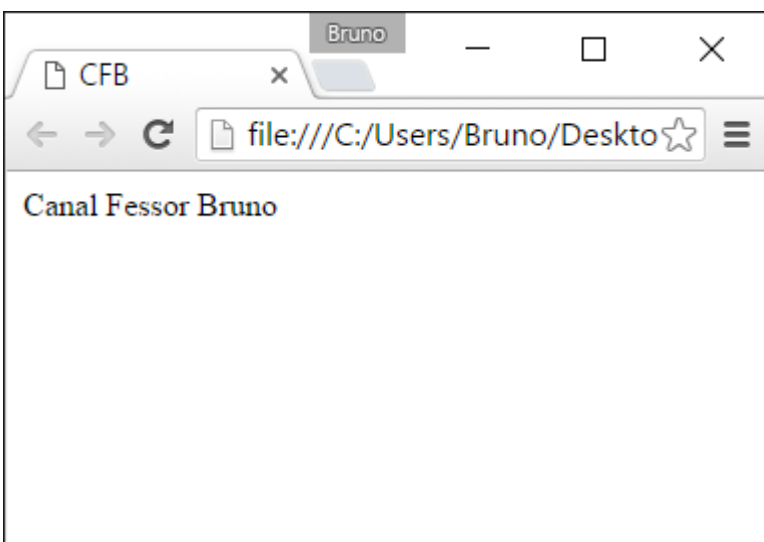
```
<html>
<head>
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js"></script>
</head>
<body>

  <p id="p1"></p>

  <script>
    $("#p1").text("Canal Fessor Bruno");
  </script>

</body>
</html>
```

Desta maneira o código jQuery consegue encontrar o elemento “p1”, veja o resultado.



Mas esta não é uma maneira muito elegante e organizada de se trabalhar, o correto é que todo código de script fique no início do documento e dentro da <head>, então o que podemos fazer?

A resposta é simples, precisamos interceptar o evento “ready” de jQuery que é semelhante ao evento “load” em javascript convencional e adicionar as rotinas neste evento.

Ou seja, quando a página for carregada e estiver pronta “ready” o código jQuery será executado, assim garantimos que os elementos serão encontrados, pois já estarão carregados.

Acompanhe o próximo capítulo onde irei mostrar o evento ready e iremos “arrumar” o código deste capítulo.

Evento ready

O evento ready é executado quando a página já terminou de ser carregada, ou seja, quando todos os elementos já estiverem sido carregados e a página está pronta para ser trabalhada pelo jQuery.

Vamos adicionar o código do evento ready na página anterior.

```
<html>
<head>
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js"></script>
  <script>

    $(document).ready(f_ready);

    function f_ready(){
      $("#p1").text("Canal Fessor Bruno");
    }

  </script>
</head>
<body>

  <p id="p1"></p>

</body>
</html>
```

Vamos ao passo a passo do código jQuery.

A primeira linha selecionemos o elemento “document”, note que para o document pode usar ou não as aspas.

```
$(document) ou $("document")
```

Para o elemento selecionado (document) informamos que vamos configurar o evento “ready”.

```
$(document).ready();
```

O ultimo passo da definição da primeira linha é informar qual função será chamada pelo evento “ready”, em nosso caso a função “f_ready”.

```
$(document).ready(f_ready);
```

Na sequência temos a declaração da função “f_ready”, lembre-se que o nome da função pode ser definido pelo programador.

```
function f_ready(){
```

Finalmente programamos o que deverá ser executado pela função “f_ready”, que será o texto do elemento do id “p1”.

```
$("#p1").text("Canal Fessor Bruno");
```

Para resumir o procedimento acima, criamos um função chamada “f_ready” com um comando que insere o texto “Canal Fessor Bruno” no elemento que tem o id “p1”. Esta função foi adicionada ao evento “ready” do documento, ou

seja, quando o documento estiver totalmente carregado o evento “ready” será disparado, executando assim a função “f_ready”.

Podemos simplificar um pouco mais todo este procedimento, não precisamos declarar um função separada do evento “ready”, podemos inserir a rotina diretamente na declaração do evento, veja.

```
<html>
<head>
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js"></script>
  <script>

    $("document").ready(function() {
      $("#p1").text("Canal Fessor Bruno");
    });

  </script>
</head>
<body>

  <p id="p1"></p>

</body>
</html>
```

Esta prática diminui um pouco nosso código e exige um pouco menos de “burocracia”.

Assim, praticamente todas as vezes que formos usar algum evento iremos usar esta convenção resumida.

```
$( "elemento" ).evento( function() {
    //COMANDOS
});
```

Mais eventos

Basicamente um evento representa o exato momento que uma determinada ação do usuário ocorre, cliques com o mouse, teclas pressionadas, página redimensionada ou carregada são algumas ações que são interceptadas por eventos em jQuery e o mais interessante é que podemos interceptar estes eventos e executar comandos quando são ocorridos.

Uma observação importante é que em jQuery os eventos são na verdade métodos que internamente tratam os eventos.

Anteriormente vimos como interceptar o evento “ready” que é disparado quando a página terminou de ser carregada, neste capítulo vamos ver mais alguns eventos importantes para nosso desenvolvimento.

click

Evento disparado quando se aplica um clique com o botão do mouse sobre o elemento, no exemplo ao clicar na div com o botão esquerdo do mouse será mostrado uma caixa de texto do tipo alert.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
</head>
<body>

  <div id="alvo">
    Clique aqui
  </div>

  <script>
    $( "#alvo" ).click(function() {
      alert("Usuário clicou no texto" );
    });
  </script>
```

```
    });  
  </script>  
  
</body>  
</html>
```

dblclick

Evento disparado quando se aplica um clique duplo com o botão do mouse sobre o elemento, no exemplo ao aplicar um clique duplo na div com o botão esquerdo do mouse será mostrado uma caixa de texto do tipo alert.

```
<!doctype html>  
<html lang="pt-br">  
<head>  
  <meta charset="utf-8">  
  <title>CFB</title>  
  <script src="jquery-3.0.0.min.js "></script>  
</head>  
<body>  
  
  <div id="alvo">  
    Clique duplo aqui  
  </div>  
  
  <script>  
    $( "#alvo" ).dblclick(function() {  
      alert("Usuário aplicou um clique duplo no texto" );  
    });  
  </script>  
  
</body>  
</html>
```

mousedown e mouseup

O evento mousedown é disparado quando o botão do mouse está pressionado e mouseup é disparado quando o botão do mouse é liberado do pressionamento, ou seja, mousedown é quando o botão vai para baixo e mouseup quando o botão vai para cima.

No código de exemplo ao pressionar o botão sobre o texto vemos a mensagem “Botão do mouse para BAIXO” e ao soltar o botão vemos a mensagem “Botão do mouse para CIMA”.

```
<!doctype html>  
<html lang="pt-br">  
<head>  
  <meta charset="utf-8">  
  <title> CFB </title>  
  <script src="jquery-3.0.0.min.js "></script>  
</head>  
<body>  
  
  <p id="txt1">Pressione e solte o botão do mouse sobre este texto</p>  
  <p id="txt2"></p>  
  
  <script>  
    $( "#txt1" ).mousedown(function() {  
      $( "#txt2" ).text("Botão do mouse para BAIXO");  
    })  
    .mouseup(function() {  
      $( "#txt2" ).text("Botão do mouse para CIMA");  
    });  
  </script>  
  
</body>  
</html>
```

mouseenter e mouseleave

O evento mouseenter é disparado quando se posiciona o cursor do mouse sobre o elemento e o evento mouseleave é disparado quando o mouse sai de cima do elemento.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <style>
    #bloco{      width:200px;
                  height:200px;
                  background-color:#f00;
    }
  </style>
</head>
<body>

  <div id="bloco"></div>
  <p></p>

  <script>
    $("#bloco").mouseenter(function() {
      $("p").text("Mouse foi posicionado sobre a div");
    })
    .mouseleave(function() {
      $("p").text("Mouse saiu de cima da div");
    });
  </script>

</body>
</html>
```

mousemove

Evento disparado quando o mouse se move sobre o elemento.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <style>
    #bloco{      width:200px;
                  height:200px;
                  background-color:#f00;
    }
  </style>
</head>
<body>

  <div id="bloco"></div>
  <p></p>

  <script>
    $("#bloco").mousemove(function() {
      $("p").text("Mouse se movendo sobre a div");
    })
    .mouseleave(function() {
      $("p").text("");
    });
  </script>

</body>
</html>
```

hover

Evento disparado quando o mouse está sobre o elemento parecido com mousemove.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <style>
    #bloco{      width:200px;
```

```

        height:200px;
        background-color:#f00;
    }
</style>
</head>
<body>

    <div id="bloco"></div>
    <p></p>

    <script>
        $("#bloco").hover(function(){
            $("p").text("Mouse está sobre a div");
        })
        .mouseleave(function(){
            $("p").text("");
        });
    </script>

</body>
</html>

```

keypress

Evento disparado quando pressionamos uma tecla do teclado.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
</head>
<body>

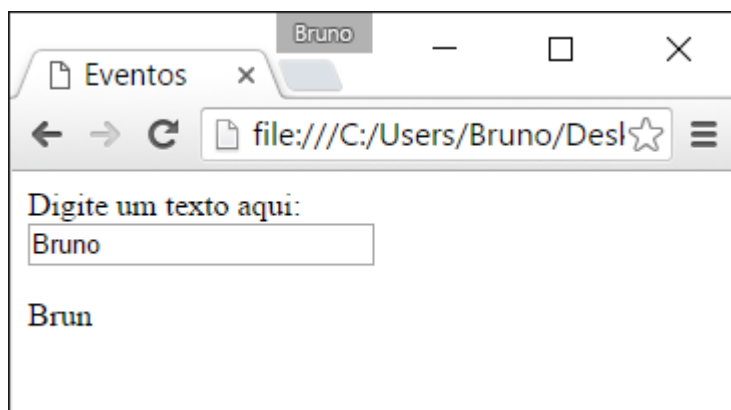
    <form>
        <label>Digite um texto aqui:</label><br/>
        <input type="text" id="txt">
    </form>
    <p></p>

    <script>
        $("#txt").keypress(function(){
            $("p").text($("#txt").val());
        });
    </script>

</body>
</html>

```

Uma observação importante, note na ilustração que no elemento input foi digitado o texto “Bruno”, porém, no <p> estamos com atraso de uma letra, isso acontece devido ao momento de execução do evento, ao pressionar a tecla o evento é capturado, então a ação é executada e só depois a caixa input recebe a letra correspondente à tecla pressionada.



keydown e keyup

O evento `keydown` é disparado quando o tecla do teclado está pressionada (para baixo) e `keyup` quando a tecla pressionada é liberada (para cima).

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
</head>
<body>

  <form>
    <label>Digite um texto aqui:</label><br/>
    <input type="text" id="txt">
  </form>
  <p id="p1"></p>
  <p id="p2"></p>

  <script>
    $("#txt").keyup(function() {
      $("#p1").text($("#txt").val());
      $("#p2").text("");
    })
    .keydown(function() {
      $("#p2").text("tecla pressionada");
    });
  </script>

</body>
</html>
```

submit

Evento disparado quando o formulário é submetido.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>

    $(document).ready(function() {
      $("#form").submit(function() {
        alert("Formulário enviado");
      });
    });
  </script>
</head>
<body>

  <form>
    <label>Nome:</label><br>
    <input type="text" id="txt_nome"><br><br>
    <label>Telefone:</label><br>
    <input type="text" id="txt_fone"><br><br>
    <input type="submit" value="enviar">
  </form>

</body>
</html>
```

change

Evento disparado quando um elemento de formulário tem seu valor alterado.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
```

```
<title> CFB </title>
<script src="jquery-3.0.0.min.js "></script>
<script>

    $(document).ready(function() {
        $("input:text").change(function() {
            $("p").text("O texto original foi alterado");
        });
    });
</script>
</head>
<body>

    <input type="text" id="txt_nome" value="Canal Fessor Bruno">
    <p></p>

</body>
</html>
```

focus e blur

O evento focus é disparado quando um elemento de formulário está selecionado (em foco) e o evento blur disparado no momento que o elemento do formulário deixa de estar selecionado.

No código de exemplo definimos os eventos focus e blur para o input do tipo text, quando o cursor está dentro da caixa de texto (focus) e quando o cursor deixa a caixa de texto (blur).

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>

        $(document).ready(function() {
            $("input:text").focus(function() {
                $("p").text("Cursor está dentro da caixa de texto");
            })
            .blur(function() {
                $("p").text("Cursor deixou a caixa de texto");
            });
        });
    </script>
</head>
<body>

    <label>Digite seu nome:</label><br>
    <input type="text" id="txt_nome">
    <p></p>

</body>
</html>
```

focusin e focusout

O evento focusin é disparado no momento que o elemento do formulário recebe o foco e o evento focusout é disparado no momento que o elemento do formulário perde o foco.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>

        $(document).ready(function() {
            $("input:text").focusin(function() {
                $("p").text("Cursor está dentro da caixa de texto");
            })
            .focusout(function() {
                $("p").text("Cursor deixou a caixa de texto");
            });
        });
    </script>
</head>
<body>

    <label>Digite seu nome:</label><br>
    <input type="text" id="txt_nome">
    <p></p>

</body>
</html>
```

```

    });
  }
</script>
</head>
<body>

  <label>Digite seu nome:</label><br>
  <input type="text" id="txt_nome">
  <p></p>

</body>
</html>

```

select

Evento disparado quando um input do tipo text por exemplo tem seu conteúdo de texto selecionando.

Confira a seguir o código de exemplo com alguns eventos de formulário.

```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
</head>
<body>

  <form>
    <label>Digite seu nome:</label><br>
    <input type="text" id="f_nome"><br><br>
    <label>Digite sua idade:</label><br>
    <input type="text" id="f_idade"><br><br>
    <label>Módulo favorito:</label><br>
    <select>
      <option>HTML</option>
      <option>CSS</option>
      <option>Javascript</option>
      <option>jQuery</option>
    </select>
    <p id="mod_sel">Módulo selecionado: HTML</p>
    <input type="submit" value="Enviar">
  </form>

  <script>
    $("form").submit(function(){
      alert("Formulário enviado");
    });
    $("input:text").focus(function(){
      $(this).css("background-color", "#ddd");
    })
    .blur(function(){
      $(this).css("background-color", "#fff");
    });
    $("select").change(function(){
      var txt="Módulo selecionado: ";
      $("select option:selected").each(function(){
        txt+=$(this).text()+" ";
      });
      $("#mod_sel").text(txt);
    });
  </script>

</body>
</html>

```

ready

Evento disparado após o carregamento total da página.

```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>

```



```
<script src="jquery-3.0.0.min.js "></script>
<script>
    $(document).ready(function() {
        alert("Página totalmente carregada");
    });
</script>
</head>
<body>

</body>
</html>
```

resize

Evento disparado quando a janela (window) é redimensionada.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        var num=0;
        $(document).ready(function() {
            $(window).resize(function() {
                num+=1;
                $("p").text("Janela com tamanho alterado: "+num);
            });
        });
    </script>
</head>
<body>

    <p></p>

</body>
</html>
```

scroll

Evento disparado quando o elemento tem seu conteúdo rolado.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>

        var num=0;
        $(document).ready(function() {
            $("div").scroll(function() {
                $("p").text(num+=1);
            });
        });
    </script>
</head>
<body>

    <div style="border:1px solid #000;width:300px;height:100px;overflow:scroll">
        Apostila de jQuery, material exclusivo do Canal Fessor Bruno.<br>
        Assine nosso canal no youtube youtube.com/canalfessorbruno e curta nossos vídeos.<br>
        Acesse também nosso site e baixe nosso material exclusivo.
    </div>
    <p>0</p>

</body>
</html>
```

on

O método on permite adicionar um ou mais eventos ao mesmo elemento, veja sua sintaxe a seguir.

`$(selector).on(evento,seletor_filho,dados_adicionais,função,mapa_de_funções)`

evento → Um ou mais eventos que serão definidos para o elemento, em caso de múltiplos eventos basta separá-los por espaço.

dados_adicionais → Especifica algum tipo de dado (variável) para ser usada ao longo da função.

função → Função que será chamada pelos eventos, caso haja múltiplos eventos que usem a mesma função.

mapa_de_funções → Definição de múltiplas funções para múltiplos eventos, caso haja múltiplos eventos que usem funções diferentes.

Vamos a alguns exemplos.

O primeiro exemplo é bem simples, anexamos um evento de clique com uma função simples.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#caixa").on("click",function() {
        $("p").text("Clicou na caixa vermelha.");
      });
    });
  </script>
</head>
<body>

  <div id="caixa" style="background:#f00; height:100px; width:100px"></div>
  <p></p>

</body>
</html>
```

No segundo exemplo iremos adicionar dois eventos ao mesmo elemento, ambos usando a mesma função, note que para múltiplos eventos os separamos por espaço.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#caixa").on("click dblclick",function() {
        $("p").text("Clicou na caixa vermelha.");
      });
    });
  </script>
</head>
<body>

  <div id="caixa" style="background:#f00; height:100px; width:100px"></div>
  <p></p>

</body>
</html>
```

No terceiro exemplo iremos adicionar três eventos ao elemento com id "#caixa", click, dblclick e mouseout, cada evento terá sua própria função, então iremos usar o parâmetro de mapa_de_funções.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
```

```
<script src="jquery-3.0.0.min.js "></script>
<script>
    $(document).ready(function() {
        $("#caixa").on({
            click:function(){$("#p").text("Clicou na caixa vermelha.");},
            dblclick:function(){$("#p").text("Clique duplo na caixa vermelha.");},
            mouseout:function(){$("#p").text("Mouse saiu da caixa vermelha.");}
        });
    });
</script>
</head>
<body>

    <div id="caixa" style="background:#f00; height:100px; width:100px"></div>
    <p></p>

</body>
</html>
```

O quarto exemplo anexamos um evento de clique que chama a função “mensagem”, note que definimos o valor do dado msg que é usado dentro da função.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>

        function mensagem(e) {
            alert(e.data.msg);
        }
        $(document).ready(function() {
            $("#caixa").on("click",{msg:"Clicou na caixa vermelha"}, mensagem);
        });
    </script>
</head>
<body>

    <div id="caixa" style="background:#f00; height:100px; width:100px"></div>
    <p></p>

</body>
</html>
```

off

O método off é exatamente o contrário do método on, ele simplesmente remove o evento adicionado ao elemento.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#p").on("click", function() {
                $(this).css("background-color", "#F88");
            });
            $("#button").click(function() {
                $("#p").off("click");
            });
        });
    </script>
</head>
<body>

    <p>Clique neste texto para mudar sua cor de fundo</p>
    <button>Clique neste botão para remover o evento de clique do texto</button>

</body>
</html>
```

trigger

O método `trigger` dispara um evento especificado para um elemento indicado, em nosso código de exemplo temos um elemento "button" com id "btSubmit" que dispara o evento "submit" do formulário.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#btSubmit").click(function() {
        $("form").trigger("submit");
      });
    });
  </script>
</head>
<body>

  <form>
    <label>Nome:</label><br>
    <input type="text"><br><br>
    <button id="#btSubmit">Enviar</button>
  </form>

</body>
</html>
```

event.pageX e event.pageY

Os métodos `pageX` e `pageY` do evento "event" retornam a posição X e Y do mouse de forma bem simples.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
</head>
<body>
  <div id="coord"></div>

  <script>
    $(document).mousemove(function(event) {
      $("#coord").text("X: " + event.pageX + ", Y: " + event.pageY);
    });
  </script>

</body>
</html>
```

event.target

Retorna o elemento que disparou o evento.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <style>
    span, strong, p {
      padding: 25px;
      display: block;
      border: 2px solid #000;
    }
  </style>
</head>
<body>
  <div id="log"></div>
</div>
```

```

        <p>
            <strong>
                <span>Clique</span>
            </strong>
        </p>
    </div>

    <script>
        $("body").click(function(event) {
            $("#log").html("Elemento clicado: "+event.target.nodeName);
        });
    </script>

</body>
</html>

```

event.which

Este método é usado junto de eventos de mouse ou teclado e indica qual tecla ou botão foi pressionado.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
</head>
<body>

    <input id="tecla" value="Digite algo">
    <div id="txt"></div>

    <script>

        $("#tecla").keydown(function(event) {
            $("#txt").html(event.type+": "+event.which);
        })
        .click(function() {
            $(this).val("");
        });
    </script>

</body>
</html>

```

event.stopPropagation

O método stopPropagation para o bubbling (borbulhamento / propagação) de um evento para os elementos pais ao elemento que disparou o evento.

Veja o código a seguir sem o uso do método stopPropagation.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("span").click(function(event) {
                alert("Span Clicado");
            });
            $("p").click(function(event) {
                alert("P Clicado");
            });
            $("div").click(function(event) {
                alert("Div Clicado");
            });
        });
    </script>
</head>
<body>

    <div>

```

```
<p>
    Canal Fessor Bruno - <br>
    <span>Curso de jQuery</span>
</p>
</div>

</body>
</html>
```

Note que temos um `` dentro de um `<p>` que está dentro de um `<div>`, então, `<div>` é pai de `<p>` que por sua vez é pai de ``.

Note também que temos eventos de clique adicionados aos três elementos, desta forma, se clicarmos no `` serão disparados os eventos de clique de `<p>` e de `<div>`.

Faça o teste.

Vamos alterar esta situação impedindo a propagação do evento para os elementos pais de ``, adicione o método `stopPropagation` conforme o código a seguir, salve e faça um novo teste.

```
<script>
    $(document).ready(function() {
        $("span").click(function(event) {
            alert("Span Clicado");
            event.stopPropagation();
        });
        $("p").click(function(event) {
            alert("P Clicado");
        });
        $("div").click(function(event) {
            alert("Div Clicado");
        });
    });
</script>
```

event.type

O método `type` retorna o tipo do evento que foi disparado, caso um elemento tenha mais de um evento associado.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("div").on("click dblclick", function(event) {
                alert("Evento disparado: " + event.type);
            });
        });
    </script>
</head>
<body>

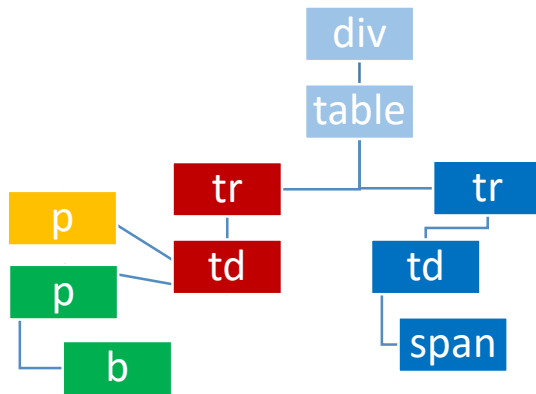
    <div>
        <p>Canal Fessor Bruno</p>
    </div>

</body>
</html>
```

Referenciando elementos, como o jQuery busca e encontra os elementos

Basicamente jQuery "se move" pela árvore de elementos da página para "encontrar" o elementos HTML selecionado para trabalho, com base na "relação/parentesco" com os outros elementos até chegar ao elemento que desejamos trabalhar, o que foi indicado pelo seletor.

Veja a árvore a seguir. Veja como podemos mover nesta árvore a partir do elemento selecionado.



Iniciando com primeiro elemento <div> que no caso é o nosso elemento base.

- <div> é **parent** (pai) de <table>, e **ancestor** (antecessor) de todos elementos dentro dele.
- <table> é **parent** (pai) dos dois elementos <tr> e **child** (filho) do elemento <div> anterior.
- <tr> vermelho é **parent** (pai) do elemento <td> vermelho, **child** (filho) do <tr> vermelho e **descendant** (descendente) do elemento <div>.
- Os elementos <p> amarelo e verde são **child** (filhos) do elemento <td> e **descendant** (descendente) do <tr> vermelho.
- Os dois elementos <tr> são **siblings** (irmãos) pois compartilham o mesmo **parent** (pai) que é o <table>.
- <tr> azul é **parent** (pai) do elemento <td> azul, **child** (filho) do elemento <table> e **descendant** (descendente) do elemento <div>.
- é child (filho) do elemento <p> verde e **descendant** (descendente) dos elementos <td> <tr> <table> e <div>.

Referências a elementos pai (parent)

Vamos aprender sobre os principais métodos para referência a elementos pai.

parent()

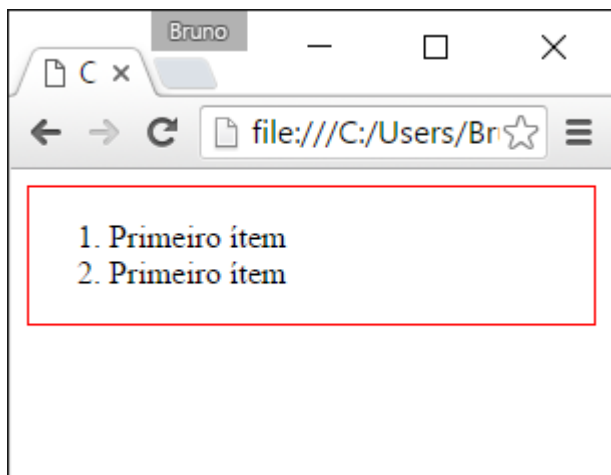
O método parent retorna o elemento “pai” do elemento selecionado, em nosso código de exemplo vamos aplicar uma formatação CSS para aplicar uma borda vermelha ao elemento pai, no caso o elemento pai do elemento é o elemento <div>, então a formatação CSS será aplicada ao elemento <div>.

```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
    });
  </script>
</head>
<body>

  <div>
    <ol>
      <li>Primeiro item</li>
      <li>Primeiro item</li>
    </ol>
  </div>

</body>
</html>
  
```



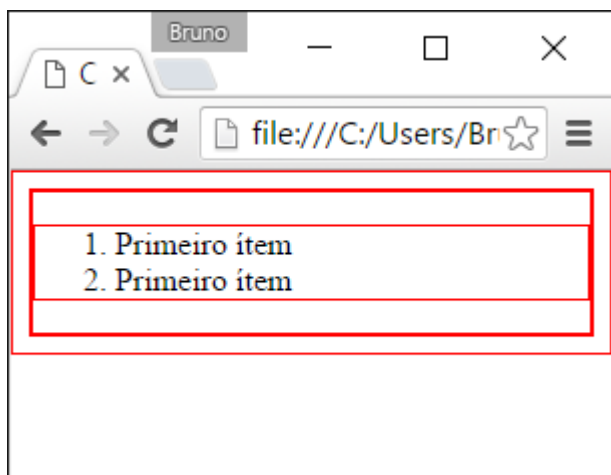
parents()

Semelhante ao método parent, porém com uma diferença, o método parent retorna somente o elemento pai direto e o método parents retorna todos os elementos que sejam anteriores ao elemento indicado.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("li").parents().css({"border":"1px solid #f00"})
    });
  </script>
</head>
<body>

  <div>
    <ol>
      <li>Primeiro item</li>
      <li>Primeiro item</li>
    </ol>
  </div>

</body>
</html>
```



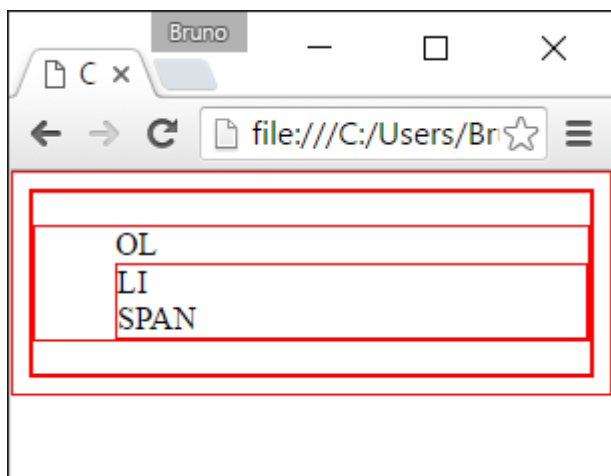
parentsUntil()

O método parentUntil retorna os elementos pai até o elemento especificado. Em nosso código de exemplo vamos primeiro aplicar o método parents e depois comparar com parentsUntil.


```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("span").parents().css({"border":"1px solid #f00"})
    });
  </script>
</head>
<body>

  <div>
    <ol>OL
      <li style="display:block">LI
        <span style="display:block">SPAN</span>
      </li>
    </ol>
  </div>

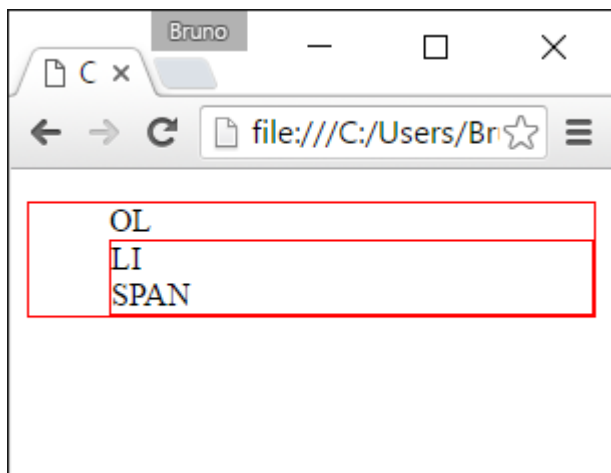
</body>
</html>
```



Veja que todos os elementos receberam a formatação CSS.

Agora vamos mudar para parentsUntil.

```
<script>
  $(document).ready(function() {
    $("span").parentsUntil("div").css({"border":"1px solid #f00"})
  });
</script>
```



Referências a descendentes

Para nos referir a elementos descendentes podemos usar dois métodos importantes children e find.

children()

Este método retorna os elementos filhos do elemento indicado.

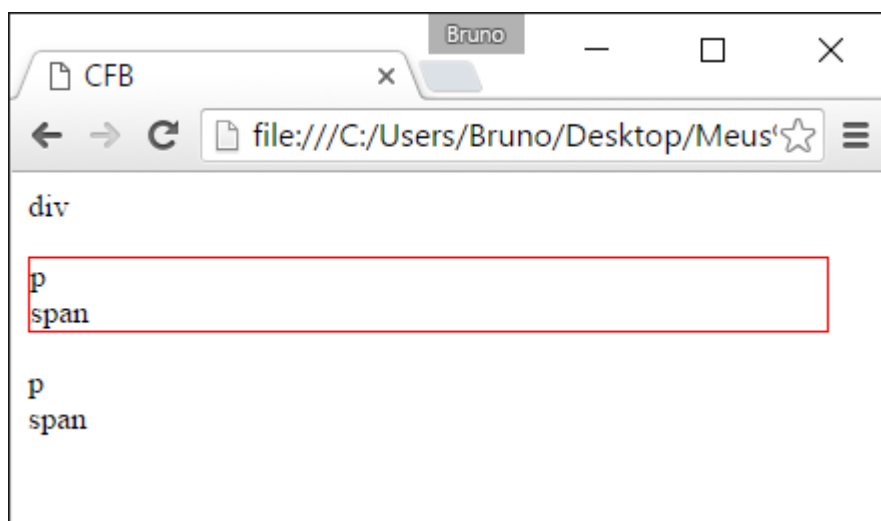
```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").children().css({"border":"1px solid #f00"})
    });
  </script>
  <style>
    span{display:block}
  </style>
</head>
<body>

  <div style="width:400px;">div
    <p>p
      <span>span</span>
    </p>
    <p>p
      <span>span</span>
    </p>
  </div>

</body>
</html>
```

Podemos indicar um elemento específico, no código de exemplo selecionamos o primeiro elemento <p> filho do elemento <div>

```
<script>
  $(document).ready(function() {
    $("div").children("p:first").css({"border":"1px solid #f00"})
  });
</script>
```



find()

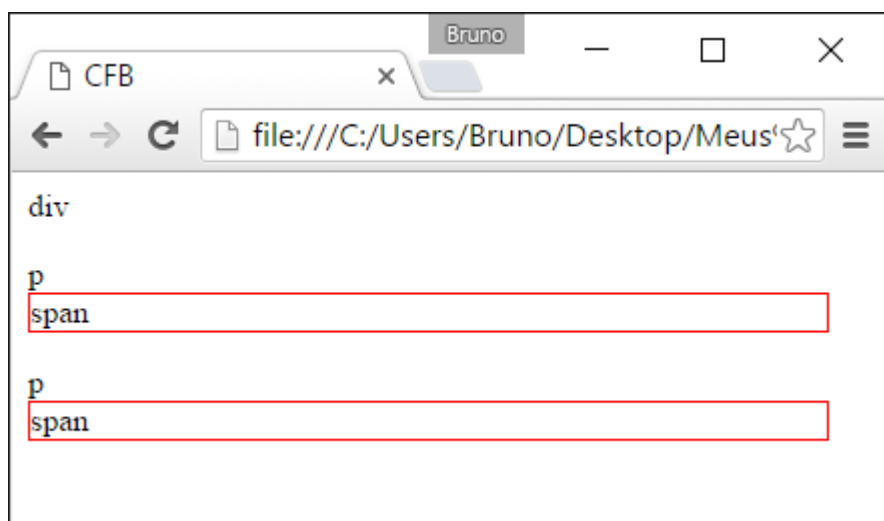
O método find procura por elementos filhos específicos do elemento indicado.

```
<!doctype html>
<html lang="pt-br">
```

```
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").find("span").css({"border":"1px solid #f00"})
    });
  </script>
  <style>
    span{display:block}
  </style>
</head>
<body>

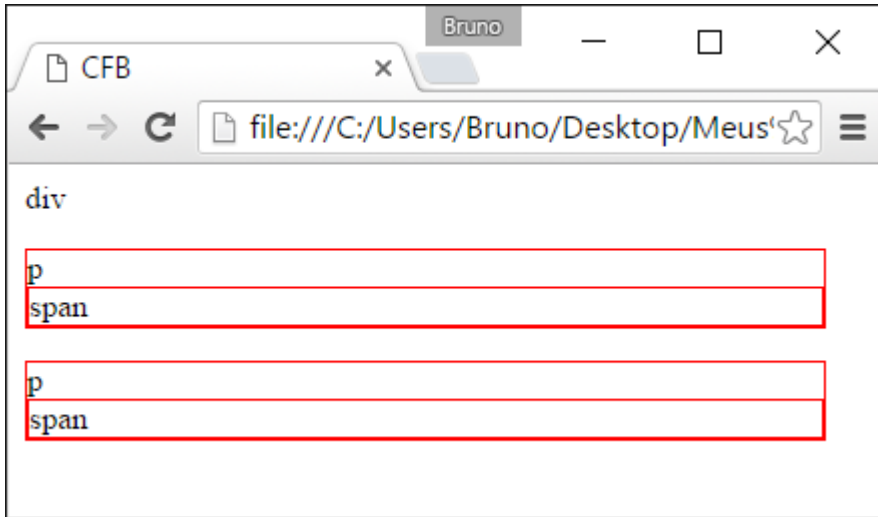
  <div style="width:400px;">div
    <p>p
      <span>span</span>
    </p>
    <p>p
      <span>span</span>
    </p>
  </div>

</body>
</html>
```



A alteração a seguir seleciona todos os filhos do elemento indicado.

```
<script>
  $(document).ready(function() {
    $("div").find("*").css({"border":"1px solid #f00"})
  });
</script>
```



Referências aos elementos irmãos (Siblings)

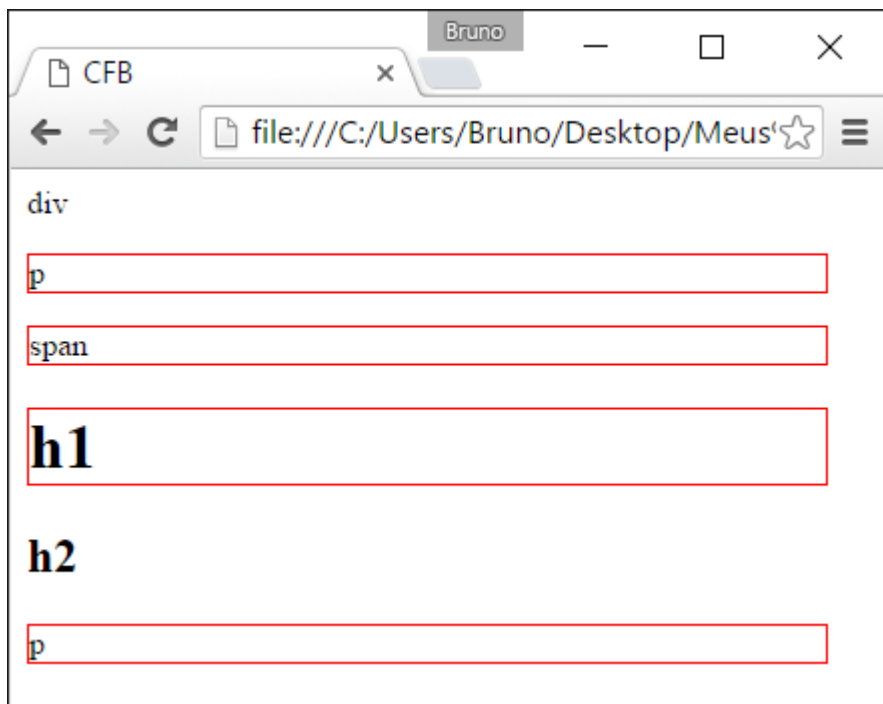
Existem uma série de métodos para referenciar elementos irmãos, elementos que estão um ao lado do outro na árvore DOM, vamos ver.

O código de exemplo a seguir seleciona todos os elementos que são irmãos do elemento <h2>.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("h2").siblings().css({"border":"1px solid #f00"})
    });
  </script>
  <style>
    span{display:block}
  </style>
</head>
<body>

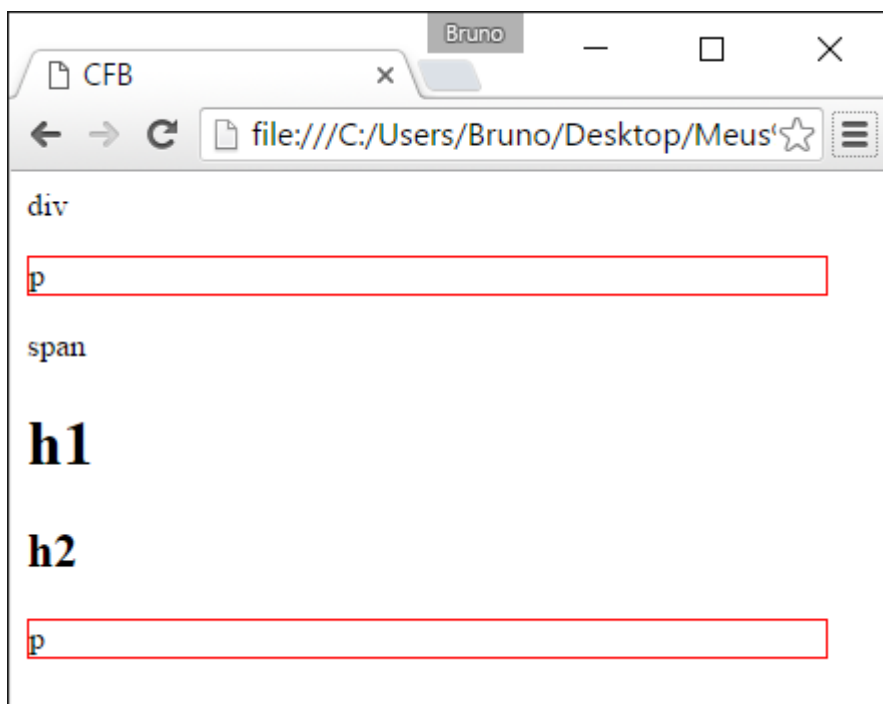
  <div style="width:400px;">div
    <p>p</p>
    <span>span</span>
    <h1>h1</h1>
    <h2>h2</h2>
    <p>p</p>
  </div>

</body>
</html>
```



A alteração a seguir retorna somente os elementos <p> que são irmãos do elemento <h2>.

```
<script>
  $(document).ready(function() {
    $("h2").siblings("p").css({"border": "1px solid #f00"});
  });
</script>
```



next()

O método next retorna o elemento irmão diretamente à direita.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
```

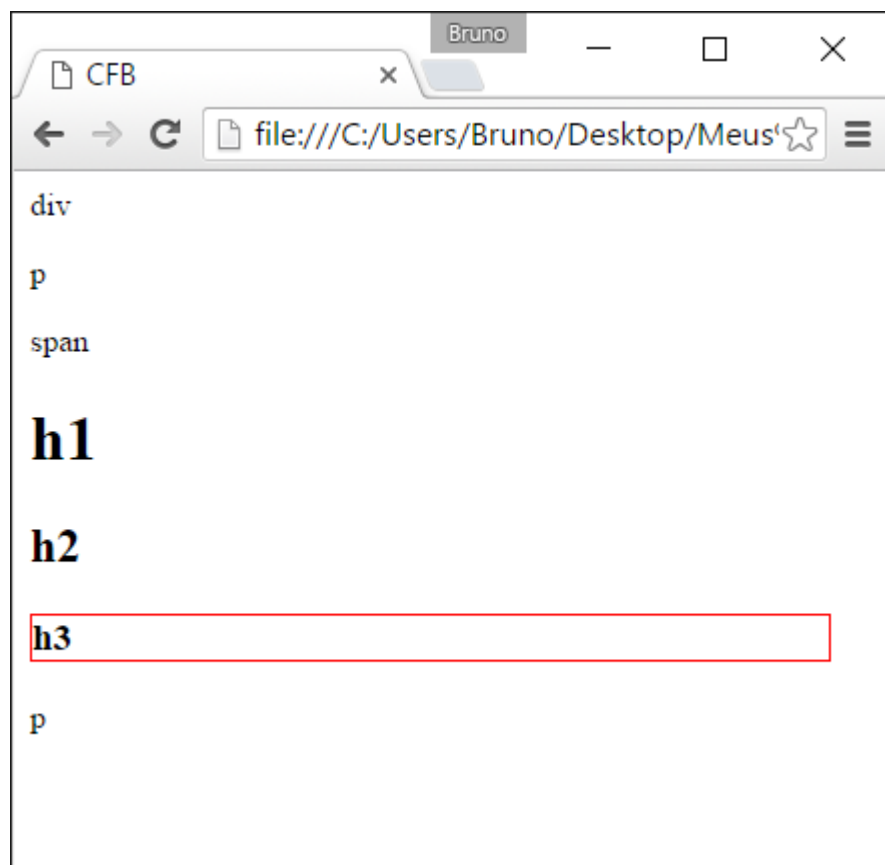
```

<script src="jquery-3.0.0.min.js "></script>
<script>
    $(document).ready(function() {
        $("h2").next().css({"border":"1px solid #f00"})
    });
</script>
<style>
    span{display:block}
</style>
</head>
<body>

    <div style="width:400px;">div
        <p>p</p>
        <span>span</span>
        <h1>h1</h1>
        <h2>h2</h2>
        <h3>h3</h3>
        <p>p</p>
    </div>

</body>
</html>

```



nextAll()

Selelmente ao método next, porém retorna todos os elementos que estão à direita.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("h1").nextAll().css({"border":"1px solid #f00"})
        });
    </script>
    <style>
        span{display:block}
    </style>

```

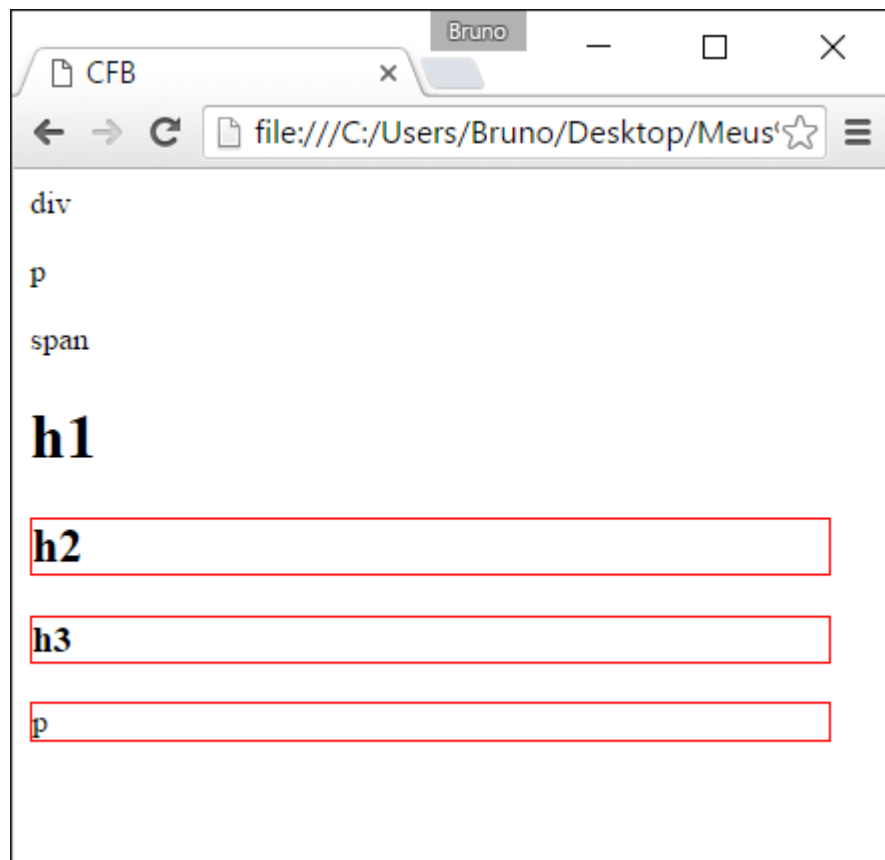
```

</style>
</head>
<body>

  <div style="width:400px;">div
    <p>p</p>
    <span>span</span>
    <h1>h1</h1>
    <h2>h2</h2>
    <h3>h3</h3>
    <p>p</p>
  </div>

</body>
</html>

```



nextUntil()

Retorna todos os irmãos à direita até o elemento especificado.

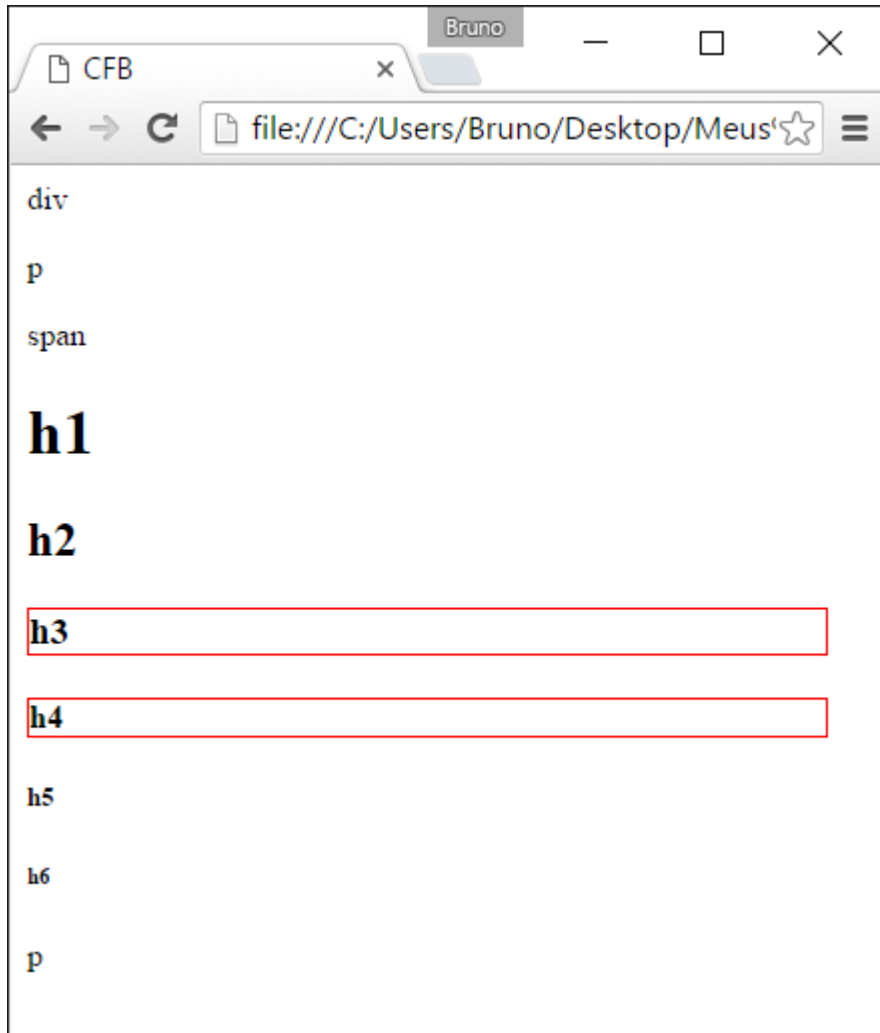
```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("h2").nextUntil("h5").css({"border":"1px solid #f00"})
    });
  </script>
  <style>
    span{display:block}
  </style>
</head>
<body>

  <div style="width:400px;">div
    <p>p</p>
    <span>span</span>
    <h1>h1</h1>

```

```
<h2>h2</h2>
<h3>h3</h3>
<h4>h4</h4>
<h5>h5</h5>
<h6>h6</h6>
<p>p</p>
</div>
</body>
</html>
```



prev(), prevAll() e prevUntil()

Ao contrário dos métodos next também temos os métodos prev, que selecionam os irmão da esquerda ou anteriores, não vou mostrar exemplos porque a forma de uso é exatamente igual aos métodos next.

Filtrando elementos (filtering)

Os principais métodos para filtragem de elementos são first(), last() e eq (), com estes métodos podemos selecionar um elemento específico baseado em sua posição em um grupo de elementos.

Outros métodos para filtragem de elementos, como filter() e not() permitem que selecionemos elementos que correspondam ou não a determinados critérios.

first() e last()

O método first retorna o primeiro elemento de um grupo de elementos especificado e o método last retorna o último.

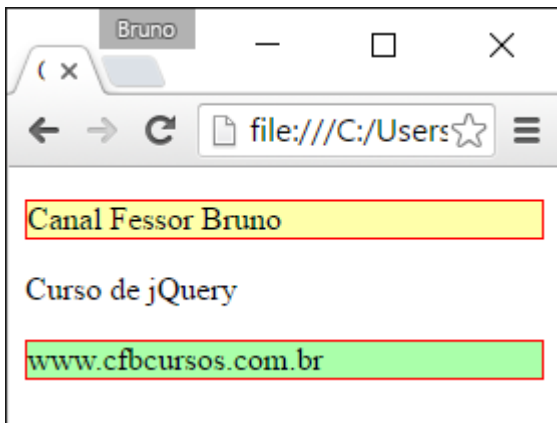
```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div p").first().css({"border":"1px solid #f00","background-color":"#ffa"});
      $("div p").last().css({"border":"1px solid #f00","background-color":"#afa"});
    });
  </script>
</head>
<body>

  <div>
    <p>Canal Fessor Bruno</p>
    <p>Curso de jQuery</p>
    <p>www.cfbcursos.com.br</p>
  </div>

</body>
</html>

```



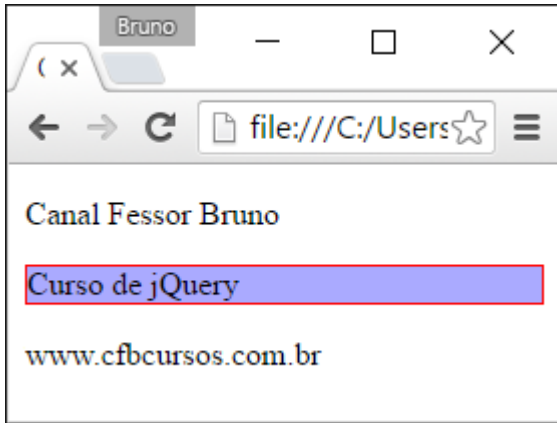
eq()

O método eq retorna o elemento equivalente à posição indicada como parâmetro, vamos alterar o código anterior para usar a função eq, não podemos esquecer que o primeiro elemento é o zero, o segundo elemento é o um e assim por diante.

```

<script>
  $(document).ready(function() {
    $("div p").eq(1).css({"border":"1px solid #f00","background-color":"#aaf"});
  });
</script>

```



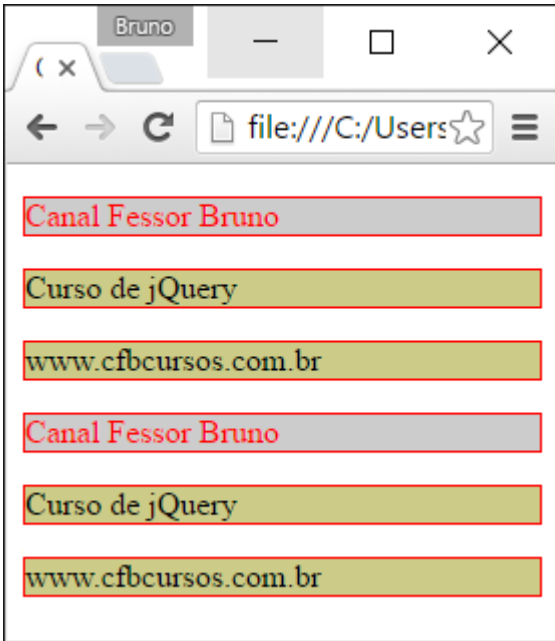
filter() e not()

O método `filter` retorna elementos bem específicos, por exemplo, elementos que usem uma determinada classe, já o método `not` é o oposto do método `filter`, retorna os elementos que não fazem equivalência com o filtro indicado.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("p").filter(".cfb").css({"border":"1px solid #f00","background-color":"#ccc"});
      $("p").not(".cfb").css({"border":"1px solid #f00","background-color":"#cc8"});
    });
  </script>
  <style>
    .cfb{color:#f00}
  </style>
</head>
<body>

  <p class="cfb">Canal Fessor Bruno</p>
  <p>Curso de jQuery</p>
  <p>www.cfbcursos.com.br</p>
  <p class="cfb">Canal Fessor Bruno</p>
  <p>Curso de jQuery</p>
  <p>www.cfbcursos.com.br</p>

</body>
</html>
```



Manipulando Elementos e atributos

É extremamente importante saber manipular elementos e atributos pelo jQuery, não é uma tarefa complicada, muito pelo contrário é bem simples.

Os três métodos mais importantes de jQuery para este fim são `text()`, `val()` e `html()`, vou descrevê-los brevemente.

`text()`

Define ou obtém o texto do elemento selecionado.

Obtendo texto do elemento.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").click(function() {
        alert("Texto do elemento: " + $(this).text());
      });
    });
  </script>
</head>
<body>

  <div id="dv">Canal Fessor Bruno</div>

</body>
</html>
```

Definindo o texto do elemento.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").click(function() {
```

```

        alert("Texto do elemento: " + $(this).text());
    });
});
</script>
</head>
<body>

    <div id="dv">Canal Fessor Bruno</div>

</body>
</html>

```

val()

O método `val()` define ou retorna o valor (value) de elementos de formulário.

Obtendo o valor do campo input.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                alert("Valor digitado no campo de texto: " + $("#txtNome").val());
            });
        });
    </script>
</head>
<body>

    <form>
        <label>Nome:</label><br>
        <input type="text" id="txtNome"><br><br>
    </form>

    <button id="bt1">Mostrar valor</button>

</body>
</html>

```

Definindo o valor da campo input.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#txtNome").val("Botão clicado");
            });
        });
    </script>
</head>
<body>

    <form>
        <label>Nome:</label><br>
        <input type="text" id="txtNome"><br><br>
    </form>

    <button id="bt1">Inserir valor</button>

</body>
</html>

```

html()

Define ou retorna o conteúdo do elemento selecionado (incluindo as tags HTML).

Obtendo o conteúdo com as tags html, o resultado da caixa alert será “Canal Fessor Bruno”.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt").click(function() {
        alert($("#p").html());
      });
    });
  </script>
</head>
<body>

  <p>Canal Fessor <b>Bruno</b></p>
  <button id="bt">Mostrar texto</button>

</body>
</html>
```

Definindo o texto de um elemento html, no caso uma tag <p>.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt").click(function() {
        $("#p").html("Texto definido");
      });
    });
  </script>
</head>
<body>

  <p>Canal Fessor <b>Bruno</b></p>
  <button id="bt">Mostrar texto</button>

</body>
</html>
```

attr()

Este método define ou retorna o valor do atributo indicado.

O exemplo a seguir obtém o valor do atributo width da div.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $(".div").click(function() {
        alert("Largura da div: " + $(this).attr("width"));
      });
    });
  </script>
</head>
<body>

  <div width="200px" height="100px">
    Clique aqui para mostrar a largura da div
  </div>

</body>
```

</html>

O código a seguir mostra o método `attr` definindo o valor do atributo "style" do elemento `div`.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").click(function() {
        $(this).attr("style", "background-color:#f88");
      });
    });
  </script>
</head>
<body>

  <div width="200px" height="100px">
    Clique aqui para definir a cor do fundo
  </div>

</body>
</html>
```

Efeitos

Agora que já sabemos trabalhar com eventos podemos dar continuidade para um assunto mais interessante que são os métodos de efeitos.

Neste capítulo vamos aprender a trabalhar com efeitos, o procedimento é bem simples e segue a sintaxe padrão do jQuery.

hide/show

Estes efeitos simplesmente ocultam (`hide`) caso os elementos estejam visíveis e mostram (`show`) caso estejam ocultos.

Em nosso código de exemplo temos dois botões que ao serem clicados um oculta e outro mostra a `div` abaixo deles.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("div").show();
      });
      $("#bt2").click(function() {
        $("div").hide();
      });
    });
  </script>
<style>
  div{ background-color:#eee;
    border:1px solid #000;
    border-radius:3px;
    padding:10px;
  }
</style>
</head>
<body>

  <button id="bt1">Mostrar</button>
  <button id="bt2">Ocultar</button><br><br>
  <div>Canal Fessor Bruno<br>Curso de jQuery</div>
```

```
</body>
</html>
```

Um detalhe interessante sobre os efeitos hide e show é que podemos informar o tempo de execução do efeito, por exemplo, ao ocultar um elemento não queremos que ele seja ocultado imediatamente, queremos que ele seja ocultado aos poucos que demore dois segundos, então basta informar o tempo do efeito como parâmetro dos métodos hide e ou show, veja.

```
$(document).ready(function(){
    $("#bt1").click(function(){
        $("#div").show(2000);
    });
    $("#bt2").click(function(){
        $("#div").hide(2000);
    });
});
```

O valor passado deve ser em milissegundos, então para dois segundos informamos o valor 2000.

Outra variação é que ainda podemos passar uma função “callback” que será chamada no fim dos efeitos, veja no exemplo que adicionamos a função “msg” como callback, assim, quando o efeito for concluído a função “msg” será chamada.

```
<script>
$(document).ready(function(){
    $("#bt1").click(function(){
        $("#div").show(2000,msg);
    });
    $("#bt2").click(function(){
        $("#div").hide(2000,msg);
    });
});

function msg(){
    alert("Efeito terminado");
}
</script>
```

toggle

O efeito toggle tem a função de ocultar e mostrar os elementos, se o elemento estiver oculto o efeito toggle o mostra, se estiver sendo mostrado o efeito toggle o oculta.

A sintaxe é a mesma dos efeitos hide e show.

`$(seletor).toggle(velocidade,callback);`

Assim como em hide e show os parâmetros de velocidade e callback podem ser ignorados.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function(){
            $("#bt").click(function(){
                $("#div").toggle();
            });
        });
    </script>
    <style>
        div{ background-color:#eee;
            border:1px solid #000;
            border-radius:3px;
            padding:10px;
        }
    </style>
</head>
```

```
<body>

    <button id="bt">Mostrar / Ocultar</button><br><br>
    <div>Canal Fessor Bruno<br>Curso de jQuery</div>

</body>
</html>
```

animate

O efeito animate usa as propriedades CSS para gerar as animações, desta forma temos um número bem grande de possibilidades.

A sintaxe padrão do método é a seguinte:

```
$(seletor).animate({estilos CSS}, velocidade, callback)
```

Lembrando que velocidade e callback podem ser ocultados.

Vamos a nosso primeiro exemplo, temos um quadrado que aumenta e diminui de tamanho conforme o clique no botão correspondente.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#caixa").animate({width:"400px"})
                .animate({height: "400px"});
            });
            $("#bt2").click(function() {
                $("#caixa").animate({width:"100px"})
                .animate({height:"100px"});
            });
        });
    </script>
    <style>
        #caixa{background:#f00;
            height:100px;
            width:100px;
        }
    </style>
</head>
<body>

    <button id="bt1">Aumentar</button>
    <button id="bt2">Diminuir</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>
```

Vamos alterar o código anterior para mover o quadrado ao invés de alterar seu tamanho.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#caixa").animate({left:"400px"})
                .animate({top:"400px"});
            });
            $("#bt2").click(function() {
                $("#caixa").animate({top:"50px"})
                .animate({left:"10px"});
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Mover para cima e direita</button>
    <button id="bt2">Mover para baixo e esquerda</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>
```



```

    });
  });
</script>
<style>
  #caixa{
    background:#f00;
    height:100px;
    width:100px;
    position:absolute;
    top:50px;
    left:10px;
  }
</style>
</head>
<body>

  <button id="bt1">Deslocar</button>
  <button id="bt2">Voltar</button>
  <br><br>
  <div id="caixa"></div>

</body>
</html>

```

Ainda podemos usar uma série de parâmetros adicionais, observe a sintaxe e confira a tabela com os parâmetros.

```
$(seletor).animate({estilos CSS},{parâmetros opcionais})
```

Os parâmetros opcionais que podemos usar são.

Parâmetro	Descrição	Exemplo
duration	Tempo de duração da animação	duration:2000
easing	Easing da animação	easing:"linear"
complete	Especifica uma função callback para ser executada no final da animação	complete:function(){ //comandos}
step	Especifica uma função para ser executada a cada passo da animação	step:function(){ //comandos}
progress	Especifica uma função para ser executada após cada passo da animação	progress:function(){ //comandos}
queue	Parâmetro booleano que determina se as animações vão ser executadas em fila ou não	queue:false
start	Especifica uma função para ser executada quando a animação começar	start:function(){ //comandos}
done	Especifica uma função para ser executada quando a animação terminar	done:function(){ //comandos}
fail	Especifica uma função para ser executada se a animação falhar	fail:function(){ //comandos}
always	Especifica uma função para ser executada caso a animação pare sem ser completada	always:function(){ //comandos}

Fila de animação

Podemos aplicar mais de um efeito de animação ao elemento, a este procedimento podemos chamar de fila de animação, vou mostrar como podemos construir e controlar uma fila de animação.

O código a seguir possui uma fila com seis efeitos de animação aplicados ao elemento com id "#caixa".

```

<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {

```

```

        $("#bt1").click(function() {
            $("#caixa").animate({width:"400px"})
                        .animate({height:"400px"})
                        .animate({opacity:"0.3"})
                        .animate({opacity:"1"})
                        .animate({height:"100px"})
                        .animate({width:"100px"});

        });
    });
</script>
<style>
    #caixa{
        background:#f00;
        height:100px;
        width:100px;
    }
</style>
</head>
<body>

    <button id="bt1">Animar</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>

```

Ao executar este código iremos ver uma animação passo a passo, ou seja, primeiro a largura para 400px, depois a altura para 400px, depois a opacidade para 0.3, opacidade para 1, altura para 100px e finalmente largura para 100px.

As ilustrações a seguir mostram passo a passo das animações.



Podemos configurar a fila de animação indicando se animação será executada em fila ou juntamente com a animação seguinte, basta usar o parâmetro adicional “queue”.

Vamos configurar de forma que as três primeiras animações sejam executadas ao mesmo tempo, sem que sejam uma a uma.

```

<script>
    $(document).ready(function() {

```

```
$("#bt1").click(function(){
    $("#caixa").animate({width:"400px"},{queue:false})
    .animate({height:"400px"},{queue:false})
    .animate({opacity:"0.3"})
    .animate({opacity:"1"})
    .animate({height:"100px"})
    .animate({width:"100px"});
});
});
</script>
```

Outra forma de fazer com que mais de um efeito seja executado ao mesmo tempo, sem que seja um a um é inserir as modificações no mesmo método animate.

```
<script>
$(document).ready(function(){
    $("#bt1").click(function(){
        $("#caixa").animate({width:"400px",height:"400px",opacity:"0.3"})
        .animate({opacity:"1",height:"100px",width:"100px"});
    });
});
</script>
```

No procedimento anterior temos uma fila com duas animações, ambas com três modificações visuais, desta forma cada animação executa suas três modificações ao mesmo tempo.

clearQueue

O método clearQueue simplesmente limpa toda a fila de animação, em nosso código de exemplo iremos criar uma fila de animação com quatro animações, o botão da direita chama o método clearQueue e limpa a fila de animação.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function(){
            $("#bt1").click(function(){
                $("#caixa").css("top","50px")
                .css("left","10px")
                .animate({left:400},1500)
                .animate({top:400},1500)
                .animate({left:10},1500)
                .animate({top:50},1500);
            });
            $("#bt2").click(function(){
                $("#caixa").clearQueue();
            });
        });
    </script>
    <style>
        #caixa{
            background:#f00;
            height:100px;
            width:100px;
            position:absolute;
            top:50px;
            left:10px;
        }
    </style>
</head>
<body>

    <button id="bt1">Deslocar</button>
    <button id="bt2">Parar deslocamento</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>
```

toggle

O método `toggle` é bastante útil quando queremos criar uma segunda animação inversa à primeira, em nosso código de exemplo temos uma div configurada em vermelho com tamanho 400px por 400px, iremos criar uma animação com três modificações, `width`, `height` e `opacity`, todos com valor `toggle`, assim quando clicarmos no botão pela primeira vez o quadrado será ocultado usando os três parâmetros que usamos e ao clicar novamente ele será mostrado de forma reversa a que foi ocultado, veja o código de exemplo.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("#caixa").animate({width:"toggle",height:"toggle",opacity:"toggle"});
      });
    });
  </script>
  <style>
    #caixa{
      background:#f00;
      height:100px;
      width:100px;
    }
  </style>
</head>
<body>

  <button id="bt1">Animar</button>
  <br><br>
  <div id="caixa"></div>

</body>
</html>
```

delay

Este método cria um intervalo de atraso antes da execução de determinada animação, em nosso exemplo ao clicar no botão “Ocultar” o efeito de `fadeOut` não será executado imediatamente e sim somente após dois segundos.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("#caixa").delay(2000).hide(250);
      });
      $("#bt2").click(function() {
        $("#caixa").show(250);
      });
    });
  </script>
  <style>
    #caixa{
      background:#f00;
      height:100px;
      width:100px;
      position:absolute;
      top:50px;
      left:10px;
    }
  </style>
</head>
<body>

  <button id="bt1">Ocultar</button>
  <button id="bt2">Mostrar</button>
  <br><br>
```

```
<div id="caixa"></div>

</body>
</html>
```

fadeIn e fadeOut

Os efeitos de fadeIn e fadeOut ocultam (fadeOut) ou mostram (fadeIn) os elementos gradativamente.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("#caixa").delay(2000).fadeOut();
      });
      $("#bt2").click(function() {
        $("#caixa").fadeIn();
      });
    });
  </script>
  <style>
    #caixa{
      background:#f00;
      height:100px;
      width:100px;
      position:absolute;
      top:50px;
      left:10px;
    }
  </style>
</head>
<body>

  <button id="bt1">Ocultar</button>
  <button id="bt2">Mostrar</button>
  <br><br>
  <div id="caixa"></div>

</body>
</html>
```

fadeTo

Este efeito de fade aplica o fadeIn ou fadeOut porém usamos um parâmetro para indicar até que ponto iremos aplicar o fade.

A sintaxe é bem simples, basta informar a velocidade ou duração e até que ponto iremos com a transparência.

`$(seletor).fadeTo(velocidade, opacidade, easing, callback);`

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("#caixa").fadeTo(1000,0.3);
      });
      $("#bt2").click(function() {
        $("#caixa").fadeTo(1000,1);
      });
    });
  </script>
  <style>
    #caixa{
      background:#f00;
      height:100px;
      width:100px;
    }
  </style>
</head>
<body>

  <button id="bt1">Ocultar</button>
  <button id="bt2">Mostrar</button>
  <br><br>
  <div id="caixa"></div>

</body>
</html>
```

```

        position:absolute;
        top:50px;
        left:10px;
    }
</style>
</head>
<body>

    <button id="bt1">Ocultar</button>
    <button id="bt2">Mostrar</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>

```

fadeToggle

Este efeito aplica fadeIn e fadeOut. Se o elemento estiver oculto será usado fadeIn e se estiver sendo mostrado será usado fadeOut.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#cx1").fadeToggle();
                $("#cx2").fadeToggle();
                $("#cx3").fadeToggle();
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Mostrar / Ocultar</button>
    <br><br>
    <div id="cx1" style="background:#f00;height:100px;width:100px;margin-bottom:10px"></div>
    <div id="cx2" style="background:#f00;height:100px;width:100px;margin-bottom:10px"></div>
    <div id="cx3" style="background:#f00;height:100px;width:100px;margin-bottom:10px"></div>

</body>
</html>

```

finish

Este método finaliza o ciclo da animação corrente, não cancela animação, só conclui imediatamente o ciclo.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#caixa").animate({width:500},5000);
            });
            $("#bt2").click(function() {
                $("#caixa").finish();
            });
        });
    </script>
    <style>
        #caixa{
            background:#f00;
            height:100px;
            width:100px;
        }
    </style>
</head>

```

```
<body>

    <button id="bt1">Ocultar</button>
    <button id="bt2">Finalizar o ciclo</button>
    <br><br>
    <div id="caixa"></div>

</body>
</html>
```

queue

Este método retorna informação da fila de animações do elemento, como o tamanho da fila por exemplo usando "length".

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                var cx=$("#caixa");
                cx.animate({width:300},500)
                .animate({height:300},500)
                .animate({width:100},500)
                .animate({height:100},500);
                $("#txt").text(cx.queue().length);
            });
        });
    </script>
    <style>
        #caixa{
            background:#f00;
            height:100px;
            width:100px;
        }
    </style>
</head>
<body>

    <button id="bt1">Animar</button>
    <br>
    <p id="txt">Tamanho da fila: </p>
    <div id="caixa"></div>

</body>
</html>
```

slideUp e slideDown

Os efeitos slideUp e slideDown, mostram e ocultam respectivamente os elementos em um efeito deslizante, para cima e para baixo.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#txt").slideUp();
            });
            $("#bt2").click(function() {
                $("#txt").slideDown();
            });
        });
    </script>
    <style>
        #txt{ background:#f00;
            padding:20px;
            text-align:center;
        }
    </style>
</head>
<body>

    <button id="bt1">Ocultar</button>
    <button id="bt2">Mostrar</button>
    <p id="txt">Tamanho da fila: </p>
    <div id="caixa"></div>

</body>
</html>
```

```

border-radius:5px;
width:300px;
position:absolute;
top:30px;
left:10px;
    }
</style>
</head>
<body>

    <button id="bt1">Para cima</button>
    <button id="bt2">Para baixo</button>
    <p id="txt">Canal Fessor Bruno</p>

</body>
</html>

```

slideToggle

Assim como os outros efeitos “toggle” o efeito slideToggle não seria diferente, ele faz o slideUp e slideDown dependendo do estado do elemento.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#txt").slideToggle();
            });
        });
    </script>
    <style>
        #txt{ background:#f00;
            padding:20px;
            text-align:center;
            border-radius:5px;
            width:300px;
            position:absolute;
            top:30px;
            left:10px;
        }
    </style>
</head>
<body>

    <button id="bt1">Para cima / Para baixo</button>
    <p id="txt">Canal Fessor Bruno</p>

</body>
</html>

```

stop

O método stop para a animação corrente, pulando para a próxima animação na fila.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("#txt").animate({width:400},2000);
                $("#txt").animate({height:400},2000);
            });
            $("#bt2").click(function() {
                $("#txt").stop();
            });
        });
    </script>

```



```

</script>
<style>
    #txt{ background:#f00;
        padding:20px;
        text-align:center;
        border-radius:5px;
        width:150px;
        position:absolute;
        top:30px;
        left:10px;
    }
</style>
</head>
<body>

    <button id="bt1">Aumentar</button>
    <button id="bt2">Parar animação corrente</button>
    <p id="txt">Canal Fessor Bruno</p>

</body>
</html>

```

Podemos usar dois parâmetros booleanos opcionais para o método stop, que são “stopAll” e “goToEnd”, veja a sintaxe.

```
$(seletor).stop(stopAll, goToEnd);
```

stopAll = Se definido em true para todas as animações da fila, não só a corrente.

```

<script>
    $(document).ready(function(){
        $("#bt1").click(function(){
            $("#txt").animate({width:400},2000);
            $("#txt").animate({height:400},2000);
        });
        $("#bt2").click(function(){
            $("#txt").stop(true, false);
        });
    });
</script>

```

goToEnd = Completa de forma instantânea todas as animações da fila e pula para última, somente executa a última animação.

```

<script>
    $(document).ready(function(){
        $("#bt1").click(function(){
            $("#txt").animate({width:400},2000);
            $("#txt").animate({height:400},2000);
        });
        $("#bt2").click(function(){
            $("#txt").stop(false, true);
        });
    });
</script>

```

Métodos para adicionar novos conteúdos

Vamos ver quatro métodos importantes com a função de adicionar novos conteúdos/elementos.

append()

Adiciona o conteúdo no final do elemento selecionado.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function(){
            $("#button").click(function(){

```

```

        $("ol").append("<li>Ítem adicionado</li>");
    });
});
</script>
</head>
<body>

    <button>Adicionar ítems</button>
    <ol>
        <li>Primeiro ítem</li>
        <li>Segundo ítem</li>
    </ol>

</body>
</html>

```

appendTo()

Adiciona o conteúdo ao elemento indicado.

```

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("<span> - Canal Fessor Bruno</span>").appendTo("li");
            });
        });
    </script>
</head>
<body>

    <button>Adicionar</button>
    <ol>
        <li>Primeiro ítem</li>
        <li>Segundo ítem</li>
    </ol>

</body>
</html>

```

prepend()

Adiciona o conteúdo no início do elemento selecionado.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("ol").prepend("<li>Ítem adicionado</li>");
            });
        });
    </script>
</head>
<body>

    <button>Adicionar ítems</button>
    <ol>
        <li>Primeiro ítem</li>
        <li>Segundo ítem</li>
    </ol>

</body>
</html>

```

prependTo()

Semelhante ao método `appendTo`, porém adiciona antes do elemento.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("<span>Canal Fessor Bruno - </span>").prependTo("li");
      });
    });
  </script>
</head>
<body>

  <button>Adicionar</button>
  <ol>
    <li>Primeiro item</li>
    <li>Segundo item</li>
  </ol>

</body>
</html>
```

before() e after()

Os métodos `before` (antes) e `after` (depois) adicionam novos elementos antes ou depois do element especificado.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("span").before("antes ");
      });
      $("#bt2").click(function() {
        $("span").after(" depois");
      });
    });
  </script>
</head>
<body>

  <button id="bt1">Adicionar antes</button> <button id="bt2">Adicionar depois</button><br><br>
  <span> MEIO </span>

</body>
</html>
```

Métodos para remover elementos

Assim como existem métodos para adicionar novos elementos, também existem métodos para remover elementos.

remove()

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
```

```

        $(".div").remove();
    });
});
</script>
</head>
<body>

    <button id="bt1">Remover div</button><br><br>
    <div>Canal Fessor Bruno</div>

</body>
</html>

```

Podemos indicar um elemento específico a ser removido, podemos indicar um elemento específico por seu id ou pela classe CSS que este elemento está usando,

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $(".div").remove("#dv2");
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Remover div</button><br><br>
    <div id="dv1">Canal Fessor Bruno</div>
    <div id="dv2">DIV a ser removida</div>
    <div id="dv3">Canal Fessor Bruno</div>

</body>
</html>

```

Podemos indicar inclusive mais de um id ou classe.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $(".div").remove("#dv1, #dv3");
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Remover div</button><br><br>
    <div id="dv1">DIV a ser removida</div>
    <div id="dv2">Canal Fessor Bruno</div>
    <div id="dv3">DIV a ser removida</div>

</body>
</html>

```

empty()

O método empty também remove elementos HTML, porém não remove o elemento pai e sim seus elementos filhos.

```

<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>

```

```
<script src="jquery-3.0.0.min.js "></script>
<script>
    $(document).ready(function() {
        $("#bt1").click(function() {
            $("div").empty();
        });
    });
</script>
</head>
<body>

    <button id="bt1">Remover conteúdo da div</button><br><br>
    <div style="border:1px solid #000;width:200px;height:150px">
        <p>Canal Fessor Bruno</p>
        <p>Curso de jQuery</p>
        <p>cfbcursos.com.br</p>
    </div>

</body>
</html>
```

css()

O método `css` nos permite manipular estilos CSS dos elementos, podemos obter ou definir CSS para o elemento selecionado.

A sintaxe é bem simples.

```
$(seletor).css("propriedade","valor");
```

O código a seguir configura a propriedade "background-color" do elemento `<p>`.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("p").css("background-color", "#88f");
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Cor de fundo do texto</button><br>
    <p>Canal Fessor Bruno</p>

</body>
</html>
```

A seguir modificamos o código a fim de obter o valor da propriedade `background-color` do elemento `<p>`.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                alert($("#p").css("background-color"));
            });
        });
    </script>
</head>
<body>

    <button id="bt1">Cor de fundo do texto</button><br>
    <p>Canal Fessor Bruno</p>
```

```
</body>
</html>
```

Multiplas propriedades css

Podemos adicionar várias propriedades CSS ao mesmo tempo para o mesmo elemento, note as diferenças na sintaxe.

```
$(seletor).css({"propriedade1":"valor1" , "propriedade2":"valor2" , "propriedade3":"valor3"});
```

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("p").css({"background-color":"#88f","font-size":"50px"});
      });
    });
  </script>
</head>
<body>

  <button id="bt1">Cor de fundo do texto</button><br>
  <p>Canal Fessor Bruno</p>

</body>
</html>
```

Note que neste caso o valor é separado por dois pontos ":" e as propriedade são separadas por vírgula ",".

Adicionando ou removendo classes CSS

Existem três comandos interessantes onde podemos manipular CSS através de classes, adicionando ou removendo classes aos elementos.

addClass() e removeClass()

O método addClass tem a função de adicionar uma classe CSS ao elemento selecionado e o método removeClass tem a função de remover.

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt1").click(function() {
        $("p").addClass("destaque");
      });
      $("#bt2").click(function() {
        $("p").removeClass("destaque");
      });
    });
  </script>
<style>
  .destaque{font-size:20px;color:#f00;background-color:#ff8};
</style>
</head>
<body>

  <button id="bt1">Adicionar classe ao elemento P</button>
  <button id="bt2">Remover classe do elemento P</button><br>
  <p>Canal Fessor Bruno</p>

</body>
</html>
```

toggleClass

O método toggleClass gerencia a alternância entre adicionar e remover uma determinada classe.

<!doctype html>

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#bt").click(function() {
        $("p").toggleClass("destaque");
      });
    });
  </script>
  <style>
    .destaque{font-size:20px;color:#f00;background-color:#ff8};
  </style>
</head>
<body>

  <button id="bt">Adicionar/Remover classe ao elemento P</button><br>
  <p>Canal Fessor Bruno</p>

</body>
</html>
```

Mais métodos

Vamos ver mais alguns métodos interessantes para manipular os elementos HTML/CSS.

clone()

Este método clona o elemento especificado, em nosso código todos os elementos <p> são clonados e adicionados no <body>.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("p").clone().appendTo("body");
      });
    });
  </script>
</head>
<body>

  <p>Canal Fessor Bruno</p>
  <p>Curso de jQuery</p>
  <button>Clonar</button>

</body>
</html>
```

Veja a sintaxe do método clone.

```
$(seletor).clone(true|false)
```

Note que podemos passar um parâmetro booleano ao método, este parâmetro indica se os eventos e métodos associados ao elemento clonado será clonado junto (true) ou não (false). O padrão é FALSE, portanto de deseja clonar também os eventos devemos indicar TRUE.

```
$("p").clone(true).appendTo("body");
```

detach()

Este método remove todos os elementos do seletor especificado.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    var txt;
    $(document).ready(function() {
      $("#bt1").click(function() {
        txt=$("#p").detach();
      });
      $("#bt2").click(function() {
        $("#body").prepend(txt);
      });
    });
  </script>
</head>
<body>

  <p>Canal Fessor Bruno</p>
  <button id="bt1">Remover</button>
  <button id="bt2">Adicionar</button>

</body>
</html>
```

O método detach() tem uma diferença importante em relação ao método remove(), o método detach() preserva uma cópia fiel do elemento em caso de reinserção, com o método remove() os métodos associados ao elemento removido não são preservados já com o método detach() sim.

empty()

O método empty() remove todos os elementos filhos de um elemento indicado, em nosso código de exemplo somente os elementos que estiverem dentro da <div> serão removidos.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    var txt;
    $(document).ready(function() {
      $("button").click(function() {
        $("#div").empty();
      });
    });
  </script>
</head>
<body>

  <button>Remover</button>
  <p>Este P está fora da div</p>
  <div>
    <p>Canal Fessor Bruno</p>
    <p>Curso de jQuery</p>
    <p>cfbcursos.com.br</p>
  </div>

</body>
</html>
```

hasClass()

o método hasClass() retorna true se o seletor indicado possui associação com uma determinada classe e retorna false se não possuir a classe.

```
<html lang="pt-br">
```



```
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    var txt;
    $(document).ready(function() {
      $("button").click(function() {
        if ($("#p1").hasClass("cfb")) {
          alert("SIM");
        } else {
          alert("NÃO");
        }
      });
    });
  </script>
  <style>
    .cfb{color:#f00}
  </style>
</head>
<body>

  <button>Verificar</button>
  <p id="p1" class="cfb">Canal Fessor Bruno</p>
  <p id="p2" >Curso de jQuery</p>
  <p id="p3" >cfbcursos.com.br</p>

</body>
</html>
```

position()

Este método retorna a posição top e left do elemento indicado.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      var pos;
      $("button").click(function() {
        pos=$("#p").position();
        alert("X=" + pos.top + " - Y=" + pos.left);
      });
    });
  </script>
</head>
<body>

  <button>Posição</button>

  <p>Canal Fessor Bruno</p>

</body>
</html>
```

offset()

O método offset() define ou retorna as coordenadas (posição) do elemento dentro da página.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    var txt;
    $(document).ready(function() {
      var pos;
      $("button").click(function() {
        pos=$("#p").offset();
        alert("X=" + pos.top + " - Y=" + pos.left);
      });
    });
  </script>
</head>
<body>

  <button>Posição</button>

  <p>Canal Fessor Bruno</p>

</body>
</html>
```

```

</script>
</head>
<body>

    <button>Verificar</button>
    <p>Canal Fessor Bruno</p>

</body>
</html>

```

Para definir a posição basta passar as coordenadas como parâmetro.

```

<script>
    $(document).ready(function() {
        $("button").click(function() {
            $("p").offset({top:100, left:100});
        });
    });
</script>

```

offsetParent()

O método offsetParent() retorna o primeiro elemento pai que tenha sido posicionado, um elemento posicionado é aquele que usa a propriedade “position” com os valores “absolute”, “relative” ou “fixed”.

```

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("p").offsetParent().css({"background-color":"#f00"});
            });
        });
    </script>
</head>
<body>

    <button>offsetParent</button>

    <div style="border:1px solid #000;position:absolute;top:40;left:0;padding:20px">
        <div style="border:1px solid #000;padding:20px">
            <div style="border:1px solid #000;padding:20px">
                <p>Canal Fessor Bruno</p>
            </div>
        </div>
    </div>

</body>
</html>

```

removeAttr()

Este método remove um determinado atributo do elemento selecionado.

```

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("p").removeAttr("style");
            });
        });
    </script>
</head>
<body>

    <button>Posição</button>
    <p style="color:#f00">Canal Fessor Bruno</p>

```

```
</body>
</html>
```

removeClass()

Remove a classe aplicada ao elemento.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      var pos;
      $("button").click(function() {
        $(".cfb").removeClass();
      });
    });
  </script>
  <style>
    .cfb{color:#f00}
  </style>
</head>
<body>

  <button>Remover classe</button>
  <p class="cfb">Canal Fessor Bruno</p>

</body>
</html>
```

replaceAll()

O método replaceAll() substitui um elemento por outro.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("<h2>Canal Fessor Bruno</h2>").replaceAll("p");
      });
    });
  </script>
</head>
<body>

  <button>ReplaceAll</button>
  <p>Canal Fessor Bruno</p>
  <p>Canal Fessor Bruno</p>
  <p>Canal Fessor Bruno</p>

</body>
</html>
```

replaceWith()

O método replaceWith() substitui um elemento por outro de forma parecida com o método replaceAll.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $(".p:first").replaceWith("<h2>Canal Fessor Bruno</h2>");
      });
    });
  </script>
```

```
</head>
<body>

    <button>ReplaceWith</button>
    <p>Canal Fessor Bruno</p>
    <p>Canal Fessor Bruno</p>
    <p>Canal Fessor Bruno</p>

</body>
</html>
```

scrollLeft() e scrollTop()

Os métodos scrollLeft() e scrollTop() definem ou retornam a posição da barra de rolagem.

```
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("#bt1").click(function() {
                $("div").scrollLeft(75);
            });
            $("#bt2").click(function() {
                $("div").scrollTop(75);
            });
        });
    </script>
</head>
<body>

    <div style="border:1px solid #000;width:100px;height:150px;overflow:auto">
        Canal Fessor Bruno, curso de jQuery, não deixe de conferir nosso site e baixe todas apostilas
        gratuitas, configura também os posts com dicas e tutoriais, cfbcursos.com.br
    </div><br>
    <button id="bt1">Rolar left</button>
    <button id="bt2">Rolar top</button>

</body>
</html>
```

Para obter a posição das barras de rolagem vamos alterar conforme o código a seguir.

```
<script>
    $(document).ready(function() {
        $("button").click(function() {
            alert("Left: " + $("div").scrollLeft() + " - Top: " + $("div").scrollTop());
        });
    });
</script>
```

unwrap()

Remove o elemento parent do elemento selecionado.

```
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("p").unwrap();
            });
        });
    </script>
</head>
<body>

    <button>Remover parents</button>
    <div style="background-color:#ddd">
```

```
<p>Canal Fessor Bruno</p>
</div>
<div style="background-color:#ddd">
  <p>Canal Fessor Bruno</p>
</div>

</body>
</html>
```

wrap()

O método wrap faz exatamente o trabalho contrário do método unwrap, ele adiciona um elemento específico como “pai” ao elemento indicado.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("p").wrap("<div></div>");
      });
    });
  </script>
</head>
<body>

  <button>Remover parents</button>
  <p>Canal Fessor Bruno</p>
  <p>Canal Fessor Bruno</p>

</body>
</html>
```

wrapAll()

Tem a funcionalizada parecida com o método wrap, porém adiciona um único elemento como pai, ao invés de um elemento para cada <p>.

```
<script>
  $(document).ready(function() {
    $("button").click(function() {
      $("p").wrapAll("<div></div>");
    });
  });
</script>
```

wrapInner()

Tem a função semelhante ao método wrap, porém ao adicionar o novo elemento antes do especificado, o método wrapInner adiciona depois.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("p").wrapInner("<b></b>");
      });
    });
  </script>
</head>
<body>

  <button>Remover parents</button>
  <p>Canal Fessor Bruno</p>
  <p>Canal Fessor Bruno</p>

</body>
```

```
</body>
</html>
```

each()

O método `each()` permite rodar uma determinada função para cada um dos elementos especificados, em nosso código de exemplo será executada a função que chama o `alert` para cada um dos elementos `<p>`.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("p").each(function() {
          alert($(this).text());
        });
      });
    });
  </script>
</head>
<body>

  <button>each</button>
  <p>Canal Fessor Bruno</p>
  <p>CFB</p>
  <p>Curso de jQuery</p>
  <p>www.cfbcursos.com.br</p>

</body>
</html>
```

get()

O método `get()` retorna os elementos DOM indicados pelo seletor.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        alert("tag:" + $("p").get(2).nodeName + " - Texto:" + $("p").get(2).innerHTML);
      });
    });
  </script>
</head>
<body>

  <button>get</button>
  <p>Canal Fessor Bruno</p>
  <p>CFB</p>
  <p>Curso de jQuery</p>
  <p>www.cfbcursos.com.br</p>

</body>
</html>
```

index()

O método `index()` retorna a posição (index) do elemento indicado, não a posição X e Y, mas se é o primeiro, segundo, terceiro, etc, elemento do tipo indicado.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
```

```

        $(document).ready(function() {
            $("p").click(function() {
                alert($(this).index());
            });
        });
    </script>
</head>
<body>

    <p>Clique nos textos para ver a posição</p>
    <p>Canal Fessor Bruno</p>
    <p>CFB</p>
    <p>Curso de jQuery</p>
    <p>www.cfbcursos.com.br</p>

</body>
</html>

```

noConflict()

O método noConflict() permite mudar o nome da variável de controle \$ caso haja algum conflito com outra linguagem.

```

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        var cfb = $.noConflict();
        cfb(document).ready(function() {
            cfb("p").click(function() {
                alert(cfb(this).index());
            });
        });
    </script>
</head>
<body>

    <p>Clique nos textos para ver a posição</p>
    <p>Canal Fessor Bruno</p>
    <p>CFB</p>
    <p>Curso de jQuery</p>
    <p>www.cfbcursos.com.br</p>

</body>
</html>

```

length()

O método length() retorna a quantidade de elementos do seletor indicado.

```

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>CFB</title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                alert($(".p").length);
            });
        });
    </script>
</head>
<body>

    <button>Quantidade de P</button>
    <p>Canal Fessor Bruno</p>
    <p>CFB</p>
    <p>Curso de jQuery</p>
    <p>www.cfbcursos.com.br</p>

</body>
</html>

```

toArray()

O método `toArray()` retorna um array com todos os elementos do tipo especificado pelo seletor.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      var vetorP=$("#p").toArray();
      $("#button").click(function() {
        alert(vetorP[3].innerHTML);
      });
    });
  </script>
</head>
<body>

  <button>toArray</button>
  <p>Canal Fessor Bruno</p>
  <p>CFB</p>
  <p>Curso de jQuery</p>
  <p>www.cfbcursos.com.br</p>

</body>
</html>
```

jquery

Esta propriedade retorna a versão usada do jQuery.

```
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>CFB</title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("#button").click(function() {
        alert($("#").jquery);
      });
    });
  </script>
</head>
<body>

  <button>toArray</button>
  <p>Clique no botão para ver a versão do jQuery</p>

</body>
</html>
```

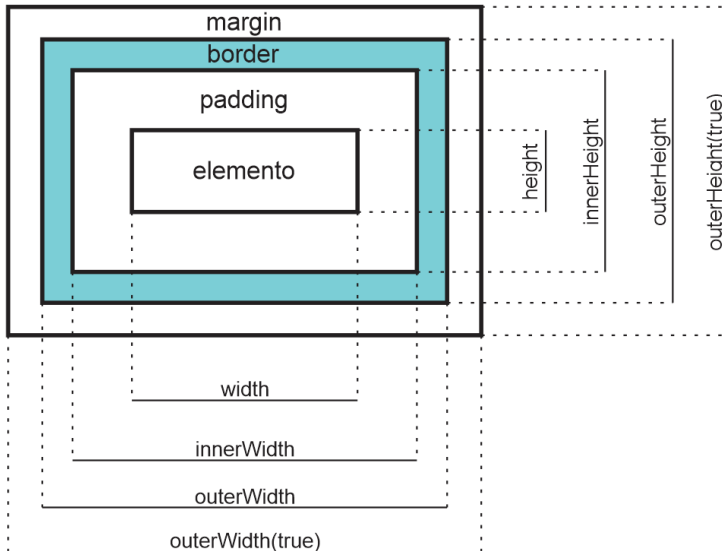
Métodos para definir ou obter alturas e larguras dos elementos

Note que o título está no plural (larguras e alturas), isso porque existem quatro formas de especificar a altura e a largura de um elemento.

Larguras → `width()`, `innerWidth()`, `outerWidth()` e `outerWidth(true)`

Alturas → `height()`, `innerHeight()`, `outerHeight()` e `outerHeight(true)`

Veja a ilustração a seguir para esclarecer as referências de larguras e alturas.



Vamos ao código de exemplo.

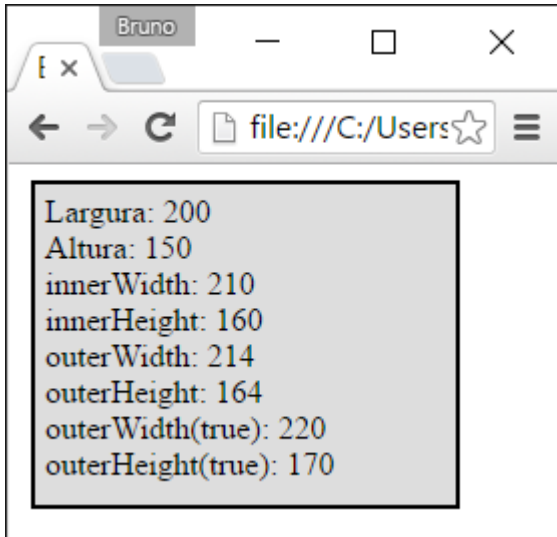
```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> CFB </title>
  <script src="jquery-3.0.0.min.js "></script>
  <script>
    $(document).ready(function() {
      $("div").click(function() {
        $(this).html("Largura: " + $(this).width() + "<br>Altura: " + $(this).height()
          + "<br>innerWidth: " + $(this).innerWidth() + "<br>innerHeight: " + $(this).innerHeight()
          + "<br>outerWidth: " + $(this).outerWidth() + "<br>outerHeight: " + $(this).outerHeight()
          + "<br>outerWidth(true): " + $(this).outerWidth(true) + "<br>outerHeight(true): " + $(this).outerHeight(true));
      });
    });
  </script>
  <style>
    div{background-color:#ddd;width:200px;height:150px;padding:5px;border:2px solid #000;margin:3px;}
  </style>
</head>
<body>

  <div>Clique aqui para ver as medidas desta div</div>

</body>
</html>
```

Compare a formatação CSS da div com o resultado depois de clicar.

```
div{ background-color:#ddd;
width:200px;
height:150px;
padding:5px;
border:2px solid #000;
margin:3px;
}
```



Largura = width

Altura = height

innerWidth = width + padding

innerHeight = height + padding

outerWidth = width + padding + border

outerHeight = height + padding + border

outerWidth(true) = width + padding + border + margin

outerHeight(true) = height + padding + border + margin

Podemos alterar o código para obter o tamanho da janela usando "document" ou "window".

```
<script>
    $(document).ready(function(){
        $("div").click(function(){
            $(this).html("Largura: " + $(document).width() + "<br>Altura: " + $(document).height());
        });
    });
</script>
```

Outra maneira de utilização é alterar o tamanho dos elementos.

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> CFB </title>
    <script src="jquery-3.0.0.min.js "></script>
    <script>
        $(document).ready(function(){
            $("div").click(function(){
                $(this).width(400)
                .height(300);
            });
        });
    </script>
    <style>
        div{background-color:#ddd;width:200px;height:150px;padding:5px;border:2px solid #000;margin:3px}
    </style>
</head>
<body>

    <div>Clique aqui para aumentar a div</div>

</body>
</html>
```

Considerações finais

Chegamos ao fim de mais uma super apostila do Canal Fessor Bruno, aprendemos como funciona e a usar a biblioteca jQuery, espero sinceramente que tenham gostado e que este conteúdo seja útil e de muita importância para você.

Agora é hora de você colocar em prática todo este conteúdo, usar sua imaginação para integrar as rotinas, mãos a obra e tenham um bom trabalho.

Um forte abraço Bruno P. Campos

