

Curso de JavaScript jQuery

Apostila: <https://goo.gl/32aGMK>

Exercícios: <https://goo.gl/F2exuM>

JSFiddle: <http://jsfiddle.net/user/diegokeller/fiddles/>

Módulo 1 - O Básico e Necessário

HTML

[O que é](#)

[Não são funções do HTML](#)

[São funções do HTML](#)

[Sintaxe e Nomeclatura](#)

[HTML 5](#)

[DOCTYPE](#)

[Debate: Web Semântica](#)

[Exercício: Layout HTML](#)

CSS

[O que é](#)

[Não são funções do CSS](#)

[São funções do CSS](#)

[Aonde colocar CSS](#)

[Sintaxe e Nomeclatura](#)

[Seletores](#)

[CSS 3](#)

[Um passo a frente: Frameworks CSS](#)

[Um passo a frente: Programando CSS com LESS ou SASS](#)

[Debate: HTML 5 e CSS 3 vs Flash](#)

[Exercício: Layout com CSS](#)

JavaScript Nativo

[O que JavaScript pode fazer](#)

[Onde posso colocar JavaScript](#)

[Dentro da Página](#)

[Em arquivos externos](#)

[Mostrando informações](#)

[Depurando código JavaScript](#)

[Funções](#)

[Palavras Reservadas](#)

[Variáveis](#)

[Operadores](#)

[Operadores Aritiméticos](#)

[Operadores de Atribuição](#)

[Operadores de Comparação](#)

[Operadores Lógicos](#)

[Operadores de Bit \(Bitwise\)](#)

[Precedência de Operadores](#)

[Tipos de Dados](#)

[Strings](#)

[Números](#)[Datas](#)[Objetos](#)[Arrays](#)[Objeto Math](#)[Eventos](#)[Interação com o Navegador \(BOM\)](#)[Window](#)[Window.Location](#)[Window.History](#)[Window.Navigator](#)[Manipulação da DOM HTML](#)[Encontrando elementos HTML](#)[Alterando Elementos HTML](#)[Adicionando eventos aos elementos](#)[Navegando pela árvore de elementos \(nodos\)](#)[Adicionando e removendo elementos](#)[Lista de Exercícios](#)[Exercício: Triângulo](#)[Exercício: Fizz Buzz](#)[Exercício: Báskara](#)[Exercício: Fibonacci](#)[Exercício: Retângulo](#)[Exercício: Fatorial](#)[Exercício: Ordenação](#)[Exercício: Idade](#)[Módulo 2 - Introdução ao jQuery](#)[O que é jQuery](#)[Funcionalidades](#)[Por que jQuery](#)[jQuery roda em todos os navegadores?](#)[Incluindo jQuery em sua página](#)[Baixando jQuery](#)[jQuery CDN](#)[Vantagens de usar por CDN](#)[Sintaxe](#)[O evento de Document Ready](#)[Seletores](#)[Seletor por nome do elemento HTML](#)[Seletor por #id](#)[Seletor por nome da classe CSS](#)[Exemplos de Seletores](#)[Eventos](#)[Método genérico para eventos](#)[Módulo 3 - jQuery HTML](#)[Obtendo e alterando valores e atributos](#)[Manipulando Texto, HTML e Valor](#)[Manipulando atributos](#)

[Adicionando elementos e conteúdo](#)

[Removendo elementos](#)

[Manipulando classes CSS](#)

[Alterando propriedades CSS](#)

[Manipulando propriedades de dimensão](#)

[Armazenando dados em elementos HTML](#)

[Para atribuir um valor](#)

[Para obter um valor](#)

[Para remover um valor](#)

[Módulo 4 - jQuery Traversing](#)

[Navegando para cima na hierarquia da DOM](#)

[Navegando para baixo na hierarquia da DOM](#)

[Navegando para os lados na hierarquia DOM](#)

[Pesquisando elementos](#)

[Seletores: Mais exemplos e formas de usar](#)

[Módulo 5 - jQuery Effects](#)

[Esconder e Mostrar](#)

[Hide e Show](#)

[Toggle](#)

[Desvanecer](#)

[Fade In](#)

[Fade Out](#)

[Fade Toggle](#)

[Fade To](#)

[Escolher](#)

[Slide Down](#)

[Slide Up](#)

[Slide Toggle](#)

[Animações](#)

[Introdução](#)

[Animando múltiplas propriedades](#)

[Animações relativas](#)

[Usando valores pré-definidos](#)

[Fila de Animações](#)

[Parando animações](#)

[Funções de Callback](#)

[Encadeamento](#)

[Módulo 6 - jQuery AJAX](#)

[Carregando um conteúdo para um elemento](#)

[Fazendo requisições GET](#)

[Fazendo requisições POST](#)

[Requisitando JSON](#)

[Eventos Globais](#)

[Encerramento](#)

[Mais de jQuery](#)

[Outros frameworks](#)

[Documentação](#)

[jQuery](#)

[Mozilla Developer Network.](#)

[HTML5](#)

[Demos](#)

Módulo 1 - O Básico e Necessário

“You must unlearn what you have learned.”

Yoda - The Empire Striked Back

Como um dos principais objetivo do jQuery é manipular HTML e CSS, é importante relembrar alguns conceitos antes de iniciarmos.

HTML

O que é

O HTML (HyperText Markup Language) é usado para definir o **conteúdo** de uma página, ou seja sua estrutura, tópicos, assuntos, textos e imagens. É através dele que expressamos as ideias e assuntos que queremos exibir em nosso site.

Não são funções do HTML

- Posicionar elementos na tela
- Formatar estilos de elementos
- Formatar o layout e aparência do site

São funções do HTML

- Compor o conteúdo da página.
- Através do uso de tags semânticas, permitir uma leitura mais inteligente do conteúdo.
- Estruturar o conteúdo da página através de seções.

Sintaxe e Nomeclatura

- Tag: é como chamamos um comandos HTML, como o “div” ou o “img”.
- Atributo: é como chamados uma informação complementar contida dentro de uma tag, como o “style” ou o “src”.



Estrutura básica de uma tag HTML

HTML 5

Em 2014 foi lançado o HTML 5¹, que proporcionou novas tags, que nos fornecem mais opções semânticas, permitindo a criação de páginas mais organizadas, legíveis e portanto, facilitando a indexação e localização de informações pelos mecanismos de busca como Google, Bing, etc.

DOCTYPE

- A tag DOCTYPE não é uma tag HTML.
- O DOCTYPE irá instruir o navegador sobre qual versão do HTML a página foi escrita, assim o navegador saberá como ler e interpretar esse conteúdo².
- O DOCTYPE deve ser a primeira linha dentro de um arquivo HTML.
- É altamente recomendado que você inclua a declaração de DOCTYPE em todas as suas páginas HTML.

¹ http://www.w3schools.com/html/html5_intro.asp

² http://www.w3schools.com/tags/tag_doctype.asp

- Você pode usar ferramentas de validação³ para verificar se o HTML que você escreveu está de acordo com os padrões estabelecidos.
- Veja abaixo a declaração básica de uma página em HTML 5

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Título do Documento</title>
</head>

<body>
  Conteúdo do documento
</body>
</html>
```

Estrutura básica de um documento HTML 5.



Debate: Web Semântica

- Você sabe como os buscadores como Google e Bing funcionam?
- Por que houve tanta preocupação em criar o HTML 5 de forma mais semântica?

Exercício: Layout HTML



- Monte um layout simples em HTML 5 com um cabeçalho, rodapé, menu de navegação e conteúdo.

CSS

O que é

O CSS (Cascading Style Sheets) é uma linguagem de folhas de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.⁴

Não são funções do CSS

- Definir o conteúdo de uma página.
- Definir o comportamento de uma página.
- Executar a lógica ou comandos/instruções de um programa.
- Executar efeitos de animação sobre elementos da página.

São funções do CSS

- Formatar e estilizar o conteúdo que foi escrito em HTML ou XML.
- Posicionar elementos na tela e definir o layout de uma página.

Aonde colocar CSS

- Dentro de elemento HTML, também chamado de “inline”:

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

○

- Dentro da tag “head” do seu documento HTML

³ http://www.w3schools.com/website/web_validate.asp

⁴ http://pt.wikipedia.org/wiki/Cascading_Style_Sheets

```

<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>

```

- Em arquivos externos que serão vinculados a sua página através da tag “link” dentro do “head” do HTML.

```

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

```

Sintaxe e Nomeclatura

- Seletor: é como chamamos a regra que irá selecionar os elementos que terão seu estilo alterado.
- Declaração: é como chamamos uma regra CSS.
- Uma declaração é uma lista de pares “propriedade” e “valor”.

seletor	declaração	declaração
h1	{ color:red; font-size:14px;}	
	<div style="display: inline-block; width: 40px; text-align: center;">↑</div> <div style="display: inline-block; width: 40px; text-align: center;">↑</div>	<div style="display: inline-block; width: 40px; text-align: center;">↑</div> <div style="display: inline-block; width: 40px; text-align: center;">↑</div>
	propriedade valor	propriedade valor

Sintaxe básica de um comando em CSS

Seletores

Existem diversas forma de selecionar os elementos para aplicar uma regra de CSS.

- Seleção pela tag

```

p {
    text-align: center;
    color: red;
}

```

- Seleciona todos elementos HTML da tag “p”.

- Seleção pelo ID

```

#para1 {
    text-align: center;
    color: red;
}

```

- Seleciona o elemento com o ID “para1”

- Seleção pela classe CSS

```
.center {  
  text-align: center;  
  color: red;  
}
```

-
- Seleciona todos os elementos que tem o valor "center" no atributo "class".

```
p.center {  
  text-align: center;  
  color: red;  
}
```

-
- Seleciona todos os elementos HTML "p" que possuem a classe CSS "center".

CSS 3

- CSS 3 ainda está em desenvolvimento, mas navegadores modernos já oferecem um grande suporte a essa nova versão da especificação.
- Consulte esse [link](#)⁵ para verificar quais recursos do CSS 3 estão disponíveis em cada navegador.
- Consulte esse [link](#)⁶ para testar quais recursos do CSS 3 o seu navegador suporta.
- Consulte esse [link](#)⁷ para verificar quais navegadores suportam algum determinado recurso do CSS 3, HTML 5 e JavaScript.



Um passo a frente: Frameworks CSS

- Atualmente é muito comum utilizar frameworks CSS como o [Bootstrap](#)⁸ e o [Foundation](#)⁹ para facilitar o desenvolvimento Web.
- Veja uma comparação de alguns frameworks CSS através desse [link](#)¹⁰.
- Leia [esse artigo](#)¹¹ para mais informações.



Um passo a frente: Programando CSS com LESS ou SASS

- CSS as vezes pode ser muito repetitivo.
- Não haveriam maneiras de melhorar isso?
- Sim! Através de LESS¹² e SASS¹³.
- Leia [esse artigo](#)¹⁴ para mais informações.



Debate: HTML 5 e CSS 3 vs Flash

- Você conhece Flash?
- Como Flash e HTML 5 e CSS 3 se comparam?
- Qual será o futuro?

⁵ http://www.w3schools.com/cssref/css3_browsersupport.asp

⁶ <http://css3test.com/>

⁷ <http://caniuse.com/>

⁸ <http://getbootstrap.com/>

⁹ <http://foundation.zurb.com/>

¹⁰ <http://usablica.github.io/front-end-frameworks/compare.html>

¹¹ <http://www.infoq.com/br/news/2009/07/dry-css-less-yass>

¹² <http://lesscss.loopinfinito.com.br/>

¹³ <http://sass-lang.com/>

¹⁴ <http://www.infoq.com/br/news/2009/07/dry-css-less-yass>



Exercício: Layout com CSS

- Aplique CSS no layout HTML 5 que você montou.
- Posicione o menu a esquerda e o conteúdo no centro.

JavaScript Nativo

Antes de iniciarmos com o estudo do framework jQuery, vamos revisar o JavaScript nativo e fixar alguns conceitos básicos.

O HTML nos dá o conteúdo, o CSS nos ajuda a formatar e estilizar esse conteúdo para apresentação, e o JavaScript irá nos auxiliar a definir o comportamento desse conteúdo quando o usuário interage com a página.

O que JavaScript pode fazer

- Acessar elementos HTML da página e alterar o seu conteúdo:
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_inner_html
- Alterar atributos de elementos HTML:
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst
- Alterar o estilos CSS de elementos HTML
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_style
- Validar dados de formulários:
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_validate

Onde posso colocar JavaScript

Você pode inserir código JavaScript em diversos locais de sua página, vejamos alguns exemplos.

Dentro da Página

Você pode adicionar código javascript nas áreas de “head” e “body” de sua página através da tag “script”.

```
<script>
document.getElementById("demo").innerHTML = "Meu primeiro JavaScript";
</script>
```

Exemplo de código javascript.

Note que não é mais necessário incluir a tag “<script type=“text/javascript”>”. JavaScript é a linguagem padrão de script para HTML.

Em arquivos externos

- Scripts podem ser colocados em arquivos externos a sua página. Eles devem ser salvos coma extensão “.js”.
- Arquivos externos são úteis quando você usa o mesmo código em diversas páginas.
- Você pode vincular arquivos externos em ambos os elementos “head” e “body”.
- O navegador irá carregá-los exatamente no mesmo lugar em que a tag “script” foi colocada.
- Para definir um script externo a página use o atributo “src” da tag “script”.
- Vantagens de utilizar arquivos externos:
 - Separa o código HTML do JS.
 - Facilita a leitura e manutenção do código.
 - Arquivos JS podem ser salvos em cache pelo navegador e diminuem o tempo de carregamento da página.

```
<!DOCTYPE html>
<html>
<body>
<script src="meuScript.js"></script>
</body>
</html>
```

Exemplo de arquivo JS externo a página.

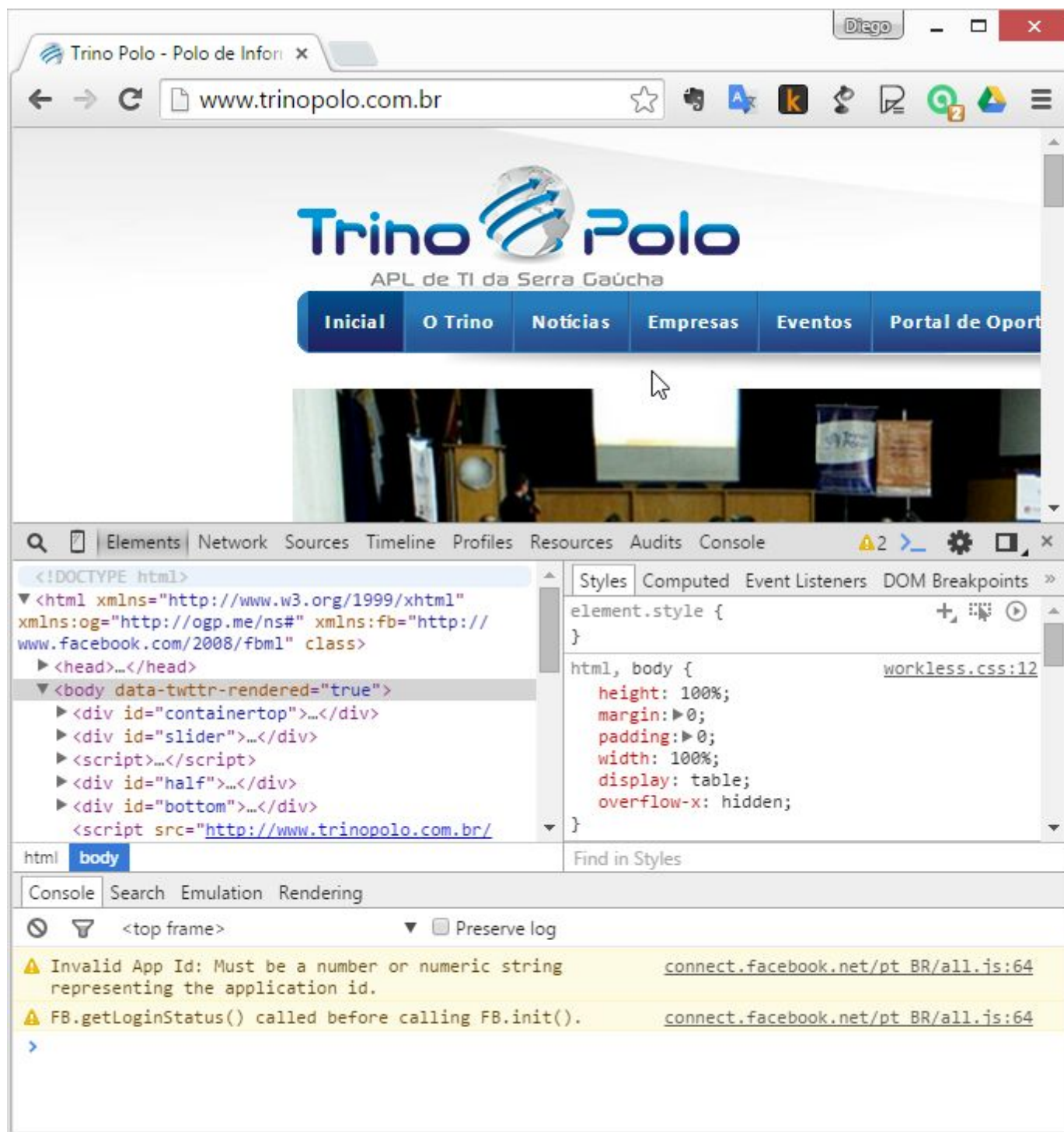
Mostrando informações

Javascript pode mostrar informações de 4 maneiras diferentes:

Comando	Descrição
window.alert()	Mostra uma caixa de alerta no navegador. http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert
document.write()	Escreve no documento (página) atual. Se o comando estiver no meio do carregamento da página, o conteúdo será escrito no local em que o comando for colocado. http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write Se o comando for executado depois que a página terminar de carregar, todo o conteúdo da página será substituído pelo comando. http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write_over
document.getElementById("demo").innerHTML	Substitui o conteúdo HTML de um elemento com o id "demo" http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom
console.log()	Gera uma mensagem de log na console do seu navegador. Ative a console através da tecla de atalho "F12" para ver a mensagem. http://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console

Depurando código JavaScript

A maioria dos navegadores atuais já inclui um depurador de código JavaScript embutido, você precisa apenas pressionar a tecla de atalho F12 para exibir o painel com as ferramentas de desenvolvimento. Use o Firefox ou o Chrome e pressione F12 para ver o painel de ferramenta. O painel de ferramentas permite que você defina "break points" no seu código para acompanhar a execução do código JavaScript.



Painel de ferramentas de desenvolvimento do Google Chrome.

Funções

- Você pode agrupar comandos JS em blocos de código através de chaves “{” e “}”.
- Esses blocos de código podem ter um nome e parâmetros de entrada.
- Também podem retornar um valor ao final de sua execução.
- Esses blocos de código são chamados de “funções” e podem ser executados a qualquer momento depois de sua declaração através da “chamada” do seu nome.
- Veja um exemplo de função:
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return
- Mais informações sobre funções:
 - http://www.w3schools.com/js/js_functions.asp

```
var x = multiplicar(4, 3);

function multiplicar(a, b) {
    return a * b;
}
```

O resultado em "x" será:

12

Exemplo de função multiplicando dois números.

Palavras Reservadas

Palavras reservadas são exclusivas de linguagem e não podem ser usadas como nomes de variáveis ou de funções. Clique sobre o nome da palavra reservada para ver mais informações. Para uma lista completa das palavras reservadas consulte [esse link](#)¹⁵.

Palavra	Descrição	Exemplo
break	Termina a execução de um switch ou um loop.	<pre>for (i = 0; i < 10; i++) { if (i === 3) { break; } text += "The number is " + i + "
"; }</pre>
continue	Pula a interação atual de um loop e retornar para o início do loop.	<pre>for (i = 0; i < 10; i++) { if (i === 3) { continue; } text += "The number is " + i + "
"; }</pre>
do ... while	Executa um bloco de comandos até que a condição seja verdadeira.	<p>Exemplo 1 - While ... do</p> <pre>while (i < 10) { text += "The number is " + i; i++; }</pre> <p>Exemplo 2 - Do ... while</p> <pre>do { text += "The number is " + i; i++; } while (i < 10);</pre>
for	Executa um bloco de comandos enquanto a expressão for verdadeira.	<p>Exemplo 1 - Array</p> <pre>for (i = 0; i < cars.length; i++) { text += cars[i] + "
"; }</pre>

¹⁵ http://www.w3schools.com/js/js_reserved.asp

		<p>Exemplo 2 - Objeto</p> <pre>var person = {fname:"John", lname:"Doe", age:25}; var text = ""; var x; for (x in person) { text += person[x]; }</pre>
function	Declara uma função.	<pre>function toCelsius(fahrenheit) { return (5/9) * (fahrenheit-32); } document.getElementById("demo").innerHTML = toCelsius(32);</pre>
if ... else	Executa um bloco de comandos ou não dependendo da condição.	<pre>if (time < 10) { greeting = "Good morning"; } else if (time < 20) { greeting = "Good day"; } else { greeting = "Good evening"; }</pre>
return	Sai de uma função.	Veja o item "function" dessa tabela.
switch	Executa um bloco de comandos de acordo com a condição.	<pre>switch (new Date().getDay()) { case 6: text = "Today is Saturday"; break; case 0: text = "Today is Sunday"; break; default: text = "Looking forward to the Weekend"; }</pre>
try ... catch	Implementa tratamento de erros para um bloco de código.	<pre><!DOCTYPE html> <html> <body> <p id="demo"></p> <script> try { adddlert("Welcome guest!"); } catch(err) { document.getElementById("demo").innerHTML = err.message; } </script> </body> </html></pre> <p>Veja o exemplo: http://www.w3schools.com/js/tryit.asp?filename=tryjs_try_catch</p>

var	Declara uma variável.	<pre>var pi = 3.14; var person = "John Doe"; var answer = 'Yes I am!';</pre>
---------------------	-----------------------	--

Variáveis

- Todas as variáveis em Javascript devem ser identificadas com nomes únicos. Esses nomes únicos são chamados de identificadores.
- Você pode declarar variáveis em qualquer parte do seu código Javascript, mas é aconselhado declarar todas as variáveis necessárias no início do script ou função, agrupando todas em um mesmo local.
- O JavaScript move todas as declarações de variáveis para o topo do bloco de código aonde ela é usada. Isso é chamado hoisting.¹⁶
- Regras para nomenclatura de identificadores:
 - Podem conter letras, número, *underline* e sinal de dolar (\$).
 - Devem começar com uma letra.
 - Também podem começar com "\$" e "_".
 - Identificadores são case sensitiv, ou seja, "x" e "X" são duas variáveis diferentes.
 - Palavras reservadas não podem ser usadas como nomes de variáveis.

Operadores^{17 18}

Operadores Aritiméticos

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo da divisão (resto).
++	Incremento
--	Decremento

Operadores de Atribuição

Operador	Exemplo	Equivalente a
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y

¹⁶ http://www.w3schools.com/js/js_hoisting.asp

¹⁷ http://www.w3schools.com/js/js_operators.asp

¹⁸ http://www.w3schools.com/js/js_arithmetic.asp

Operadores de Comparação

Operador	Descrição
==	Igualdade
===	Igualdade de conteúdo e tipo
!=	Diferente
!==	Diferente em conteúdo ou tipo
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que

Operadores Lógicos

Supondo que x = 6, y = 3.

Operador	Descrição	Exemplo
&&	and	(x < 10 && y > 1) = true
	or	(x == 5 y == 5) = false
!	not	!(x == y) = true

Operadores de Bit (Bitwise)

- Funcionamento em número de 32 bits.
- Os números são convertidos para binário e depois que a instrução é executada, eles são novamente convertidos para decimal.

Operador	Descrição	Exemplo	Equivalente a	Resultado	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
	OR	x = 5 1	0101 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

Precedência de Operadores

Operador	Precedência
()	Agrupamento de expressões
++ --	Incremento e decremento
* / %	Multiplicação, divisão e módulo da divisão
+ -	Adição e subtração.

Tipos de Dados



Jogo: Adivinhe o resultado

Tente adivinhar o resultado de cada um desses comandos abaixo.

1.

```
var x = 16 + "Volvo";
```

2.

```
var x = 16 + 4 + "Volvo";
```

3.

```
var x = "Volvo" + 16 + 4;
```

4.

```
typeof undefined
typeof null
null === undefined
null == undefined
```

- Variáveis em Javascript podem ser de diferentes tipos de dados: Número, String, Objeto, Array, Booleano.
- Em uma expressão, se o segundo parâmetro é uma string, o primeiro será tratado como string também, por isso no jogo acima o exemplo 1 resultará em "16Volvo".
- Expressões são executadas da esquerda para a direita, por isso no jogo acima o exemplo 2 resultará em "20Volvo" e o exemplo 3 resultará em "Volvo164".
- Javascript tem tipos dinâmicos, isso significa que você pode trocar o tipo de uma variável em tempo de execução.

```
var x;           // Agora x é "undefined"
var x = 5;       // Agora x é um "Number"
var x = "John";  // Agora x é uma "String"
```

- Strings podem usar aspas duplas ou simples.
- Valores booleanos são representados por "true" e "false".
- Arrays são definidos usando colchetes e separando os valores por vírgula.

```
var cars = ["Saab", "Volvo", "BMW"];
```

- Objetos são definidos usando chaves e são escritos em um formato de pares de chave e valor.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

- Você pode usar o comando "typeof" para descobrir o tipo de dados de uma variável.

```
typeof "John"           // Returns string
typeof 3.14              // Returns number
typeof false            // Returns boolean
typeof [1,2,3,4]         // Returns object
typeof {name:'John', age:34} // Returns object
```

- Quando uma variável ainda não foi inicializada, ela terá um tipo chamado "undefined".
- Undefined e Null são diferentes


```

typeof undefined // undefined
typeof null      // object
null === undefined // false
null == undefined // true

```

- Não instancie String com o operador “new”, isso torna a execução mais lenta e pode levar a erros como no exemplo abaixo:

```

var x = new String("John");
var y = new String("John");

// (x == y) será FALSO
// Objetos não podem ser comparados em Javascript

```

Strings

Objetos do tipo String possuem algumas funções pré-definidas. Clique sobre o nome da função para ver um exemplo ou [clique aqui](#)¹⁹ para a lista completa das funções de string.

Propriedade / Método	Descrição
length	Retorna o comprimento da string (quantos caracteres ela tem). Note que é uma propriedade, não um método (não tem ()).
charAt()	Retorna o caracter em uma posição.
charCodeAt()	Retorna o Unicode de um caracter em uma posição.
concat()	Junta duas ou mais strings e retorna.
fromCharCode()	Converte valores Unicode para caracteres.
indexOf()	Retorna a posição da primeira ocorrência de um termo em uma string.
lastIndexOf()	Retorna a posição da última ocorrência de um termo em uma string.
localeCompare()	Compara duas string com a localidade corrente.
match()	Faz uma pesquisa na string usando uma expressão regular e retorna as ocorrências.
replace()	Substitui uma parte de uma string por outra.
search()	Retorna a posição de uma expressão dentro de uma string.
slice()	Extrai uma parte de uma string.
split()	Divide a string em um array de strings.
substr()	Extrai uma parte de uma string baseado no índice de início e número de caracteres subsequentes.
substring()	Extrai uma parte de uma string baseado no índice inicial e final.
toLocaleLowerCase()	Transforma a string para minúsculo de acordo com a localidade do host.
toLocaleUpperCase()	Transforma a string para maiúsculo de acordo com a localidade do host.
toLowerCase()	Converte uma string para minúsculo.
toString()	Retorna o valor de um objeto String.

¹⁹ http://www.w3schools.com/jsref/jsref_obj_string.asp

toUpperCase()	Converte uma string para maiúsculo.
trim()	Remove os espaço em branco do início e fim de uma string.
valueOf()	Retorna o valor primitivo de um objeto string.

Números

- Javascript tem apenas um tipo de dados para número.
- Podem ser escritos com ou sem casas decimais.
- Use "." para definir número com casas decimais.

```
var x1 = 34.00;
var x2 = 34;
```

- Valores muito grandes ou muito pequenos podem ser expressos usando notação científica (exponencial).

```
var y = 123e5;      // 12300000
var z = 123e-5;    // 0.00123
```

- Números em JavaScript são sempre armazenados como double, seguindo o padrão internacional ([IEEE 754](#)²⁰)
- Os número são armazenados em 64 bits, de acordo com a definição abaixo:

Valor	Expoente	Sinal
52 bits (0 - 51)	11 bits (52 - 62)	1 bit (63)

- Precisão da parte inteira: 15 dígitos:

```
var x = 999999999999999; // x will be 999999999999999
var y = 999999999999999; // y will be 1000000000000000
```

- Precisão da parte decimal, até 17 casas, mas operações aritméticas com pontos flutuantes nem sempre são 100% precisas.

```
var x = 0.2 + 0.1;      // x will be 0.30000000000000004
```

- Quando um número ultrapassa os valores definidos por esses limites, o Javascript irá atribuir o valor "Infinite".

```
var myNumber = 2;
while (myNumber != Infinity) {      // Execute until Infinity
    myNumber = myNumber * myNumber;
}
```

- Divisão por zero não gera um erro, mas o resultado será "Infinite".

```
var x = 2 / 0;      // x will be Infinity
var y = -2 / 0;     // y will be -Infinity
```

²⁰ http://pt.wikipedia.org/wiki/IEEE_754

- Quando uma operação matemática não pode ser realizada, Javascript irá retornar NaN (Not a Number / Não é um número) como resultado.

```
var x = 100 / "Apple"; // x will be NaN (Not a Number)
```

- Você pode usar a função isNaN() para testar se um valor é um número ou não.

```
var x = 100 / "Apple";  
isNaN(x); // returns true because x is Not a Number
```

- Não instancie número usando o operador "new". Isso torna a execução mais lenta e pode levar a erros como o exemplo abaixo.

```
var x = new Number(500);  
var y = new Number(500);  
  
// (x == y) será FALSO  
// objetos não podem ser comparados em Javascript
```

Objetos de números tem algumas funções pré-definidas. Clique sobre o nome da função para ver um exemplo ou [clique aqui](#)²¹ e [aqui](#)²² para a lista completa das funções de string.

Propriedade / Método	Descrição
MAX_VALUE	Retorna o maior valor possível em Javascript.
MIN_VALUE	Retorna o menor valor possível em Javascript.
NEGATIVE_INFINITY	Representa o negativo infinito (retornado em caso de overflow).
NaN	Representa um valor que não é um número.
POSITIVE_INFINITY	Representa o infinito (retornado em caso de overflow).
toExponential(x)	Converte o número em notação exponencial.
toFixed(x)	Formata o número com "x" dígitos após o separador decimal.
toPrecision(x)	Formata um número para "x" dígitos de precisão.
toString()	Converte um número para String.
valueOf()	Retorna o valor primitivo do número.
Funções globais do Javascript	
parseFloat()	Interpreta o argumento e o transforma em um número de ponto flutuante.
parseInt()	Interpreta o argumento e o transforma em um número inteiro.

Datas

- Javascript permite criar e manipular datas.
- Internamente, Javascript armazena datas como um número inteiro que representa o número de milissegundos desde 01/01/1970 00:00:00.
- Você pode criar datas de 4 maneiras diferentes

²¹http://www.w3schools.com/js/js_number_methods.asp

²² http://www.w3schools.com/jsref/jsref_obj_number.asp

```
// Cria a data atual
new Date()

// Cria uma data com o número de milissegundos desde 01/01/1970 00:00:00
new Date(milissegundos)

// Cria uma data a partir de uma String que representa uma data
new Date(dataEmString)

// Cria uma data com os valores especificados
new Date(ano, mes, dia, hora, minuto, segundo, milissegundo)
```

- O mês é um intervalo entre 0 e 11, onde 0 é Janeiro e 11 Dezembro.
- Para ver exemplos de criação de objetos de data consulte [esse link](#)²³ e [esse link](#)²⁴.
- Objetos de data já possuem alguns funções pré-definidas que você pode usar, consulte na tabela abaixo. Clique no nome da função para ver um exemplo. Clique [nesse link](#)²⁵ para ver a lista completa de funções disponíveis.

Método	Descrição
getDate()	Retorna o dia do mês (entre 1-31)
getDay()	Retorna o dia da semana (entre 0-6)
getFullYear()	Retorna o ano (4 dígitos)
getHours()	Retorna a hora (entre 0-23)
getMilliseconds()	Retorna os milissegundos (entre 0-999)
getMinutes()	Retorna os minutos (entre 0-59)
getMonth()	Retorna o mês (entre 0-11)
getSeconds()	Retorna os segundos (entre 0-59)
getTime()	Retorna o número de milissegundos transcorridos desde 01/01/1970 até a data representada pelo objeto.

Objetos

Você pode criar objeto em Javascript, definir suas propriedades e atribuir-lhe comportamento através de funções (métodos). Veja um exemplo.

²³ http://www.w3schools.com/js/js_dates.asp

²⁴ http://www.w3schools.com/js/js_date_formats.asp

²⁵ http://www.w3schools.com/jsref/jsref_obj_date.asp

<pre>1 </pre>	HTML	1
<pre>1 // Cria um objeto de pessoa 2 var pessoa = { 3 4 // Atributos da pessoa 5 primeiroNome:"João", 6 ultimoNome:"da Silva", 7 nascimento: new Date("1980-10-25"), 8 9 // Nome completo 10 nomeCompleto: function() { 11 return this.primeiroNome + ' ' + this.ultimoNome; 12 }, 13 14 // Calcula a idade 15 idade: function() { 16 var dataAtual = new Date(); 17 return dataAtual.getFullYear() - this.nascimento.getFullYear(); 18 } 19 }; 20 21 // Mostra informações da pessoa 22 var info = document.getElementById('informacoes'); 23 info.innerHTML = pessoa.nomeCompleto() + 24 " tem " + pessoa.idade() + " anos.";</pre>		João da Silva tem 35 anos.

<http://jsfiddle.net/diegokeller/j957s2Lr/>

Arrays

- Arrays são usados para armazenar múltiplos valores em uma única variável.
- Criando arrays

Sintaxe

```
var array-name = [item1, item2, ...];
```

Exemplo

```
var cars = ["Saab", "Volvo", "BMW"];
```

- Você também pode criar usando a palavra chave "new", mas isso é desaconselhado pois diminui a performance do programa.

```
var cars = new Array("Saab", "Volvo", "BMW");
```

- Você pode criar um array com um determinado número de elementos. No exemplo abaixo o array terá 40 posições.


```
var points = new Array(40);
```

- Os elementos de um array não precisam ser do mesmo tipo de dados. Dentro de um mesmo array você pode armazenar, números, strings, datas, etc.
- Typeof de um array será "object".
- O que diferencia um array de um objeto é que no array acessamos as propriedades (elementos) através de um índice numérico, e nos objetos acessamos através do nome da propriedade.
- Arrays em Javascript não podem ser indexados por Strings.
- Para saber quantos elementos um array tem use a propriedade "length".
- Arrays são indexados usando número, iniciando em 0. O primeiro elemento do array está na posição 0 (zero), o segundo elemento do array está na posição 1 (um), e assim por diante.

```
var person = [];  
person[0] = "John";  
person[1] = "Doe";  
person[2] = 46;  
var x = person.length;           // person.length retorna 3  
var y = person[0];               // person[0] retorna "John"
```

- Para percorrer os elementos de um array use o comando "for".

```
var index;  
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
for (index = 0; index < fruits.length; index++) {  
    text += fruits[index];  
}
```

Arrays já possuem algumas funções pré-definidas que você pode usar. Consulta a tabela abaixo e clique no nome da função para ver um exemplo. Para ver um lista completa de exemplos clique [nesse link](http://www.w3schools.com/js/js_array_methods.asp)²⁶.


Propriedade	Descrição
length	Seta ou obtém o número de elementos de um array
Método	Descrição
concat()	Junta dois ou mais arrays
indexOf()	Retorna a posição de um elemento dentro do array
join()	Junta todos os elementos de um array em uma string
lastIndexOf()	Retorna a última posição de um elemento dentro do array
pop()	Remove o último elemento de um array e retorna esse elemento
push()	Adiciona um novo elemento ao final do array e retorna o novo comprimento do array
reverse()	Reverte a ordem dos elementos de um array
shift()	Remove o primeiro elemento de um array e retorna esse elemento.
slice()	Extraí uma parte de um array e retorna um novo array com essa parte.
sort()	Ordena os elementos de um array

²⁶ http://www.w3schools.com/js/js_array_methods.asp

splice()	Adiciona o remove elementos de um array
toString()	Converte o array uma string e retorna essa string.
unshift()	Adiciona elementos ao início de um array e retorna o seu novo comprimento
valueOf()	Retorna o valor primitivo de um array (equivalente a toString())

Objeto Math

- Javascript tem um objeto chamado “Math” que lhe permite executar diversas operações matemáticas.
- Com esse objeto você pode gerar número aleatório, efetuar arredondamentos, calcular raiz quadrada, exponencial, etc.
- Veja abaixo um exemplo de como gerar um número aleatório:

○ 

Consulte a lista completa de propriedades e funções disponíveis o objeto Math. Clique no nome da propriedade ou função para ver um exemplo, ou consulte a referência completa [clikando aqui](#)²⁷.

Propriedade	Descrição
E	Retorna o número de Euler (aprox. 2.718)
LN2	Retorna o logaritmo natural de 2 (aprox. 0.693)
LN10	Retorna o logaritmo natural de 10 (aprox. 2.302)
LOG2E	Retorna o logaritmo de E na base 2 (aprox. 1.442)
LOG10E	Retorna o logaritmo de E na base 10 (aprox. 0.434)
PI	Retorna o valor de PI (aprox. 3.14)
SQRT1_2	Retorna a raiz quadrada de 1/2 (aprox. 0.707)
SQRT2	Retorna a raiz quadrada de 2 (aprox. 1.414)
Método	Descrição
abs(x)	Retorna o valor absoluto de x
acos(x)	Retorna o arco co-seno de x em radianos
asin(x)	Retorna o arco seno de x em radianos
atan(x)	Retorna a arco tangente de x como um valor numérico entre -PI/2 e PI/2 radianos
atan2(y,x)	Retorna o angulo em radianos no sentido anti horário entre o eixo X positivo e o ponto definido por y e x
ceil(x)	Retorna o número x arredondado para cima
cos(x)	Retorna o co seno de x (x em radianos)
exp(x)	Retorna o valor de E elevado na potência x
floor(x)	Retorna o número x arredondado para baixo
log(x)	Retorna o logaritmo natural de E na base x.
max(x,y,z,...,n)	Retorna o maior valor dentre os argumentos passados para a função

²⁷ http://www.w3schools.com/jsref/jsref_obj_math.asp

min(x,y,z,...,n)	Retorna o menor valor dentre os argumentos passado para a função
pow(x,y)	Retorna o valor de x elevado na potência y
random()	Gera um número aleatório entre 0 e 1
round(x)	Retorna o número x arredondado para o inteiro mais próximo (para cima se a parte decimal for maior ou igual a 0.5 e para baixo se for menor que 0.5)
sin(x)	Retorna o valor do seno de x (x em radianos)
sqrt(x)	Retorna a raiz quadrada de x
tan(x)	Retorna a tangente um angulo

Eventos

- Eventos são “coisas” que acontecem aos elementos HTML. Quando usamos Javascript podemos reagir a esses eventos.
- Eventos podem acontecer em função de algo que o navegador fez, ou que o usuário fez.
- Exemplos de eventos seriam:
 - Quando a página termina de carregar
 - Quando o valor de um campo é alterado
 - Quando um botão é clicado
- Quando alguns desses eventos acontecem você pode querer executar alguma ação.
- Eventos são definidos em atributos dos elementos HTML que “disparam” esses eventos.
- Você pode programar um evento diretamente dentro do elemento HTML (não recomendado):

```
<button onclick="this.innerHTML=Date()">Que hora é?</button>
```

-
- Ou você pode chamar funções para serem executadas (aconselhado)

```
<button onclick="mostrarHora()">Que hora é?</button>
```

-
- Alguns eventos mais comuns:

Evento	Descrição
onchange	Quando um elemento HTML é alterado.
onclick	Quando o usuário clica em um elemento.
onmouseover	Quando o usuário passa o mouse sobre um elemento.
onmouseout	Quando o usuário tira o mouse de um elemento.
onkeydown	Quando o usuário pressiona uma tecla.
onload	Quando a página termina de carregar.

Para uma lista completa dos eventos disponíveis consulte:

http://www.w3schools.com/jsref/dom_obj_event.asp

Interação com o Navegador (BOM)

Window

- Propriedades

- Obter o tamanho da janela (sem as barras de rolagem e de ferramentas). Válido para IE 9 ou maior.
 - **window.innerWidth**: largura da janela.
 - **window.innerHeight**: altura da janela.
- Para IE menor ou igual 8
 - **document.body.clientHeight**: altura da janela.
 - **document.body.clientWidth**: largura da janela.
- Métodos
 - **window.open**: abre uma nova janela.
 - **window.close**: fecha a janela atual.
 - **window.moveTo**: move a janela para uma posição x e y.
 - **window.resizeTo**: redimensiona a janela para um tamanho x e y.
 - **window.alert**: mostra uma mensagem.
 - **window.confirm**: solicita a confirmação do usuário.
 - **window.prompt**: solicita uma informação ao usuário.
 - **window.setTimeout**: agenda a execução periódica de uma função.
 - **window.setInterval**: agenda a execução única de uma função.
 - **window.cookie**: permite a criação, alteração e leitura de cookies.

Você só pode redimensionar e mover janela que você mesmo abriu.

[Clique aqui](#)²⁸ para ver exemplos de criação de *timers*.

[Clique aqui](#)²⁹ para ver exemplos de criação de cookies.

Clique [nesse link](#)³⁰ para ver mais propriedades e métodos.

Window.Location

Usado para acessar a barra de endereços do navegador.

- Propriedades:
 - **window.location.href**: Retorna o endereço atual da página.
- Métodos:
 - **window.location.assing**: Carrega uma nova página passada por parâmetro.

Clique [nesse link](#)³¹ para ver mais propriedades e métodos.

Window.History

Usado para acessar o histórico de páginas visitadas na janela.

- Métodos:
 - **window.history.back**: Voltar para a página anterior.
 - **window.history.forward**: Avança para a página seguinte no histórico.

Clique [nesse link](#)³² para ver mais propriedades e métodos.

²⁸ http://www.w3schools.com/js/js_timing.asp

²⁹ http://www.w3schools.com/js/js_cookies.asp

³⁰ http://www.w3schools.com/jsref/obj_window.asp

³¹ http://www.w3schools.com/jsref/obj_location.asp

³² http://www.w3schools.com/jsref/obj_history.asp

Window.Navigator

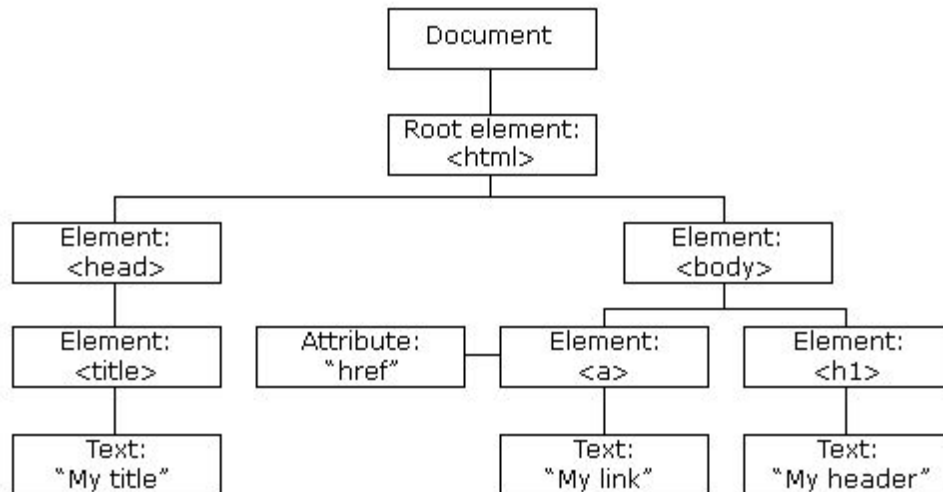
Usado para acessar informações sobre o navegador sendo usado para visualizar a página.

- Propriedades:
 - **window.navigator.userAgent**: Retorna informações sobre o navegador.

Clique [nesse link](#) para ver mais propriedades e métodos.

Manipulação da DOM HTML

Document Object Model (DOM) é a árvore de objetos que constituem uma página HTML:



Com JavaScript você pode acessar e manipular todos os elementos da DOM, incluindo, alterando e excluindo elementos ou atributos de elementos. Vamos ver algumas funções úteis:

Encontrando elementos HTML

Método	Descrição
<code>document.getElementById()</code>	Busca um elemento HTML pelo seu ID.
<pre>var x = document.getElementById("intro");</pre>	
<code>document.getElementsByTagName()</code>	Busca elementos HTML pelo nome da tag HTML.
<pre>var x = document.getElementsByTagName("p");</pre>	
<pre>var x = document.getElementById("main"); var y = x.getElementsByTagName("p");</pre>	
<code>document.getElementsByClassName()</code>	Busca elementos HTML pelo nome de uma classe CSS.
<pre>var x = document.getElementsByClassName("intro");</pre>	

Alterando Elementos HTML

Método/Propriedade	Descrição
element.innerHTML=	Altera o conteúdo HTML que está dentro da tag.
<pre><html> <body> <p id="p1">Hello World!</p> <script> document.getElementById("p1").innerHTML = "New text!"; </script> </body> </html></pre>	
element.attribute=	Altera um atributo de um elemento HTML.
<pre><!DOCTYPE html> <html> <body> <script> document.getElementById("myImage").src = "landscape.jpg"; </script> </body> </html></pre>	
element.style.property=	Altera o estilo CSS de um elemento.
<pre><html> <body> <p id="p2">Hello World!</p> <script> document.getElementById("p2").style.color = "blue"; </script> <p>The paragraph above was changed by a script.</p> </body> </html></pre>	

[Clique aqui](#)³³ para acessar uma lista mais completa das propriedades CSS que podem ser alteradas.

Adicionando eventos aos elementos

Você pode adicionar eventos aos seus elementos HTML de três formas:

- 1) Adicionando código JS no atributo HTML do próprio elemento:

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML='Oops!'">Click on this text!</h1>

</body>
</html>
```

a)

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
    id.innerHTML = "Oops!";
}
</script>

</body>
</html>
```

b)

- 2) Adicionando eventos via JS, depois de obter o elemento HTML, através da propriedade correspondente ao evento.

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

a)

- 3) Adicionando eventos via JS, depois de obter o elemento HTML, através da função `addEventListener`. Nessa opção, note que o prefixo "on" não é adicionado ao nome do evento, ficando "click" ao invés de "onclick". Esta forma não é muito utilizada, pois só o Internet Explorer só aceita essa opção a partir da versão 9.

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

a)

```
element.addEventListener("click", myFunction);

function myFunction() {
    alert ("Hello World!");
}
```

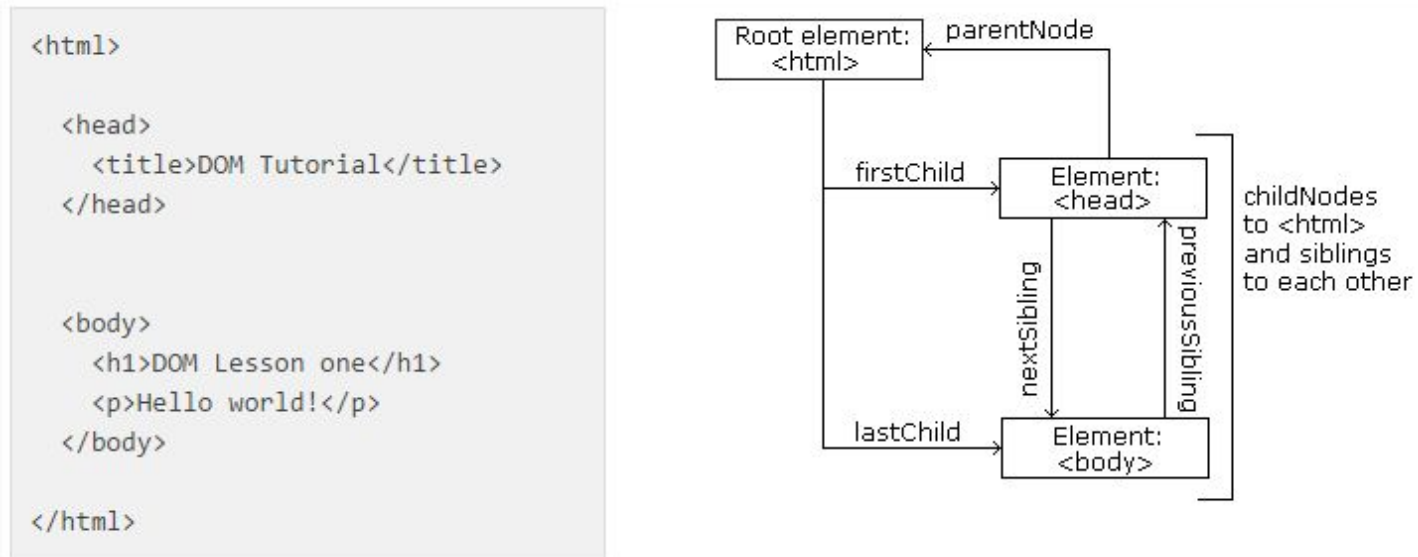
b)

³³ http://www.w3schools.com/jsref/dom_obj_style.asp

[Clique aqui](#)³⁴ para consultar uma lista dos eventos disponíveis no DOM HTML.

Navegando pela árvore de elementos (nodos)

JavaScript oferece métodos para navegar entre os elementos da árvore DOM. Vejamos esse exemplo abaixo:



Você pode usar as seguintes propriedades de um elemento HTML para navegar pela árvore DOM:

Método	Descrição
elemento.parentNode	Acesso o pai do elemento em questão.
elemento.childNodes[i]	childNodes Retorna os filhos do elemento em questão. childNodes[i] Retorna o filho na posição "i". Cuidado para não acessar uma posição inválida (inexistente).
elemento.firstChild	Retorna o primeiro filho do elemento.
elemento.lastChild	Retorna o último filho do elemento.
elemento.nextSibling	Retorna o próximo elemento "irmão" (de mesmo nível) no elemento atual
elemento.previousSibling	Retorna o elemento anterior de mesmo nível do elemento atual (irmão).

A propriedade "nodeName" retorna o nome do nodo.

- nodeName é somente leitura
- nodeName de um elemento HTML é o mesmo nome da tag HTML.
- nodeName de um nodo do tipo atributo é o nome do atributo.
- nodeName de um nodo de texto é sempre #text
- nodeName do nodo representado pelo "document" será sempre #document

A propriedade "nodeValue" de um nodo define o valor do nodo.

- nodeValue para nodos do tipo Element é undefined.
- nodeValue para nodos do tipo Text é o texto em si.
- nodeValue para nodos de atributo é o valor do atributo.

A propriedade "nodeType" retorna um inteiro com o tipo do nodo. Os tipos de nodo mais importantes são:

³⁴ http://www.w3schools.com/jsref/dom_obj_event.asp

Tipo de Nodo	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Adicionando e removendo elementos

Método	Descrição
document.createElement()	Cria um elemento HTML
elemento.appendChild()	Adiciona um elemento HTML a outro, tornando o elemento inserido o último filho do elemento ao qual ele foi adicionado.

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

pai.insertBefore(element, irmao)	Adiciona o "element" dentro do "pai" e antes do "irmao", permitindo escolher a posição em que o elemento será inserido.
---	---

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(para, child);
</script>
```

element.removeChild()	Remove um elemento HTML
------------------------------	-------------------------


```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

pai.replaceChild(novo, antigo)

Substitui o elemento "antigo" pelo "novo" dentro do "pai".

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para, child);
</script>
```

document.write(text)

Escreve no documento HTML.

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write(Date());
</script>

</body>
</html>
```

Lista de Exercícios



Exercício: Triângulo

- Monte uma página HTML com três inputs texto e um botão.
- Ao clicar no botão o programa deve mostrar um alerta informando se o triângulo formado pelos três valores é equilátero (todos iguais) isósceles (dois lados iguais) ou escaleno (os três são diferentes).

Exercício: Fizz Buzz

- Faça uma programa que liste os número de 1 a 100
- Para os múltiplos de 3, ao invés do número, escreva "Fizz".
- Para os múltiplos de 5, ao invés do número escreva "Buzz".
- Para os que são múltiplos de 3 e 5, ao invés do número escreva "FizzBuzz".

Exercício: Báskara

- Monte uma página que tenha três campos correspondentes aos valores de a, b e c de uma fórmula de báskara.
- Calcule as duas raízes possíveis e mostre o resultado em elemento HTML.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

•

Exercício: Fibonacci

- Monte uma página HTML com um campo texto.
- O programa deve gerar e mostrar em um alerta o enésimo termo da série de fibonacci (n será digitado no campo texto).

Exercício: Retângulo

- Crie um objeto chamado retângulo que tenha dois atributos referentes a altura e largura.
- Crie um método que calcule a área e mostre em um alert.

Exercício: Fatorial

- Monte uma página HTML com um campo texto.
- Calcule e mostre o fatorial do número informado.

Exercício: Ordenação

- Monte uma página HTML com um campo texto e um botão chamado "Ordenar".
- O usuário irá digitar dentro desse campo texto n números separados por ",".
- Ao clicar em "Ordenar", ordene os valores do campo texto e atualize o campo mostrando os valores ordenados novamente.
- Você mesmo deve implementar o algoritmo de ordenação. Não usar a função sort().

Exercício: Idade

- Monte uma página HTML com um campo texto para que o usuário digite a data de nascimento.
- Calcule a idade do usuário e mostra em um alerta a se é maior de idade ou menor, e a sua idade.

Módulo 2 - Introdução ao jQuery

“ I'm trying to free your mind, Neo. But I can only show you the door. You're the one that has to walk through it.”

Morpheus - Matrix

O que é jQuery

jQuery é uma biblioteca JavaScript *cross-browser* desenvolvida para simplificar os *scripts client side* que interagem com o HTML. Ela foi lançada em dezembro de 2006 no BarCamp de Nova York por John Resig. Usada por cerca de 77% dos 10 mil sites mais visitados do mundo, jQuery é a mais popular das bibliotecas JavaScript.

jQuery é uma biblioteca de código aberto e possui licença dual, fazendo uso da Licença MIT ou da GNU General Public License versão 2. A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos e desenvolver aplicações AJAX. A biblioteca também oferece a possibilidade de criação de plugins sobre ela. Fazendo uso de tais facilidades, os desenvolvedores podem criar camadas de abstração para interações de mais baixo nível, simplificando o desenvolvimento de aplicações web dinâmicas de grande complexidade.³⁵

Funcionalidades

Principais funcionalidades do jQuery:

- Resolução da incompatibilidade entre os navegadores.
- Redução de código.
- Reutilização do código através de plugins.
- Utilização de uma vasta quantidade de plugins criados por outros desenvolvedores.
- Trabalha com AJAX e DOM.
- Implementação segura de recursos do CSS1, CSS2 e CSS3.

Por que jQuery

Há vários outros frameworks JavaScript, mas jQuery parece ser o mais popular, e também o mais extensível. Muitas das maiores empresas na Web usam jQuery, tais como:

- Google
- Microsoft
- IBM
- Netflix

jQuery roda em todos os navegadores?

O time da jQuery sabe tudo sobre questões cross-browser, e eles têm escrito esse conhecimento para a biblioteca jQuery. jQuery irá executar exatamente o mesmo em todos os principais navegadores, incluindo o Internet Explorer 6!

Incluindo jQuery em sua página

Há duas maneiras de incluir o jQuery em sua página:

- Baixando a biblioteca jQuery do site jQuery.com
- Incluindo jQuery de uma CDN, como o Google

³⁵ <http://pt.wikipedia.org/wiki/jQuery>

Baixando jQuery

Há duas versões do jQuery disponíveis para download:

- Versão de produção - essa é para o seu site de produção, pois esta foi minimizada e comprimida.
- Versão em desenvolvimento - essa é para testes e desenvolvimento (código descompactado e legível)

Ambas as versões podem ser baixadas do [jQuery.com](http://jquery.com).

A biblioteca jQuery é um arquivo JavaScript único, e você faz referência a ele com a tag HTML `<script>` (veja que o tag `<script>` deve estar dentro da seção `<head>`):

```
<head>
<script src="jquery-2.1.4.min.js"></script>
</head>
```

Nesse exemplo o arquivo do jQuery está localizado dentro da mesma pasta dos arquivos irão utilizá-lo. Se você desejar colocá-lo em outro sub diretório de sua aplicação, por exemplo "js", você deve alterar o atributo "src" e incluir esse caminho (src="js/jquery-2.1.4.min.js").

jQuery CDN

Se você não deseja fazer o download e hospedar o jQuery, você pode incluí-lo a partir de um CDN (Content Delivery Network). Tanto o Google e Microsoft hospedam o jQuery. Para usar jQuery do Google³⁶ ou Microsoft³⁷, use um dos seguintes procedimentos:

Google CDN:

```
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
</head>
```

Microsoft CDN:

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.4.min.js"></script>
</head>
```

Vantagens de usar por CDN

Muitos usuários já baixaram jQuery do Google ou Microsoft ao visitar outro site. Como resultado, ele será carregado a partir do cache quando eles visitam seu site, o que leva a um tempo de carregamento mais rápido. Além disso, a maioria dos servidores de CDN irá certificar-se que uma vez que um usuário solicita um arquivo a partir dele, este será servido a partir do servidor mais próximo do usuário, o que também leva a um tempo de carregamento mais rápido.

Sintaxe

Com jQuery você basicamente seleciona (query) elementos para então executar ações sobre ele. A sintaxe básica de um comando em jQuery é: **\$(selector).action()**

³⁶ <https://developers.google.com/speed/libraries/>

³⁷ http://www.asp.net/ajax/cdn#jQuery_Releases_on_the_CDN_0

- Um sinal de “\$” para definir o acesso á biblioteca do jQuery.
- Um seletor “(**selector**)” para buscar um ou vários elementos HTML.
- Uma ação jQuery “.**action()**” para ser executada sobre o(s) elemento(s).

Exemplos:

\$(this).hide() - Esconde o elemento atual.

\$("p").hide() - Esconde todos os elementos “p”.

\$(".test").hide() - Esconde todos os elementos com a classe CSS “test”.

\$("#test").hide() - Esconde o elemento com o id “test”

jQuery usa a sintaxe do CSS para selecionar os elementos. Vamos ver mais opções de seleção de elementos em seguida.

O evento de Document Ready

Você deve ter notado que todos os métodos do jQuery nossos exemplos, estão dentro de um evento de “document ready”:

```
$(document).ready(function(){  
  
    // Comandos do jQuery vão aqui...  
  
});
```

Isso é para evitar qualquer código jQuery seja executado antes que o documento seja carregado por completo (esteja pronto). É uma boa prática esperar que o documento seja totalmente carregado e esteja pronto antes de trabalhar com ele. Isso também permite que você coloque o seu código de JavaScript antes do “body” do documento, na seção “head”.

Alguns exemplos de ações que podem falhar se os métodos do jQuery forem executados antes que o documento seja totalmente carregado:

- Tentar esconder um elemento que ainda não foi criado no HTML.
- Tentar obter o tamanho de uma imagem que ainda não está carregada.

O time da jQuery também criou um método ainda mais curto para o evento pronto documento:

```
$(function(){  
  
    // Comandos do jQuery vão aqui...  
  
});
```

Use a sintaxe que você preferir, mas é recomendado usar a primeira opção, pois ela deixa a leitura do código mais clara.

Seletores

Seletores são uma das partes mais importantes da biblioteca jQuery. Seletores permitem selecionar e manipular elemento(s) HTML. Seletores são usados para “encontrar” (ou selecionar) elementos HTML com base no ID, classes, tipos, atributos, valores de atributos e muito mais. Os seletores do jQuery são baseados

nos seletores do CSS³⁸, e, além disso, tem algumas seletores próprios personalizados. Todos os seletores começam com o sinal de dólar e parênteses: `$()`. Vamos ver agora todas as opções de seleção de elementos através do jQuery.

Seletor por nome do elemento HTML

O seletor de elemento seleciona elementos com base no nome da tag HTML. Você pode selecionar todos os elementos `<p>` em uma página, por exemplo:

```
$("p")
```

Seletor por #id

O seletor `#id` usa o atributo `id` de uma tag HTML para encontrar um elemento específico. Um `id` deve ser único dentro de uma página, então você deve usar o seletor `#id` quando você quer encontrar um único elemento. Para encontrar um elemento com um ID específico, escreva um caractere de sustenido (jogo da velha), seguido do `id` do elemento HTML que deseja selecionar:

```
$("#test")
```

Seletor por nome da classe CSS

O seletor de classe encontra elementos com uma classe CSS específica. Para encontrar elementos com uma classe específica, escrever um caractere de ponto, seguido pelo nome da classe:

```
$(".test")
```

Exemplos de Seletores

Veja abaixo alguns exemplos de seletores jQuery. Clique na última coluna para ver um exemplo.

Sintaxe	Descrição	Exemplo
<code>\$("*")</code>	Seleciona todos os elementos de uma página	Exemplo
<code>\$(this)</code>	Selecione o elemento atual (corrente), usado quando se escreve um comando dentro de um evento de um elemento HTML por exemplo.	Exemplo
<code>\$("p.intro")</code>	Seleciona todos os elementos <code><p></code> com a classe CSS <code>class="intro"</code>	Exemplo
<code>\$("p:first")</code>	Seleciona o primeiro elemento <code><p></code>	Exemplo
<code>\$("ul li:first")</code>	Seleciona o primeiro elemento <code></code> do primeiro elemento <code></code>	Exemplo
<code>\$("ul li:first-child")</code>	Seleciona o primeiro elemento <code></code> de cada elemento <code></code>	Exemplo
<code>\$("[href]")</code>	Seleciona todos os elementos que tem um atributo <code>href</code> .	Exemplo
<code>\$("a[target='_blank']")</code>	Seleciona todos os elementos <code><a></code> com um atributo de <code>target</code> com valor <code>"_blank"</code>	Exemplo
<code>\$("a[target!='_blank']")</code>	Seleciona todos os elementos <code><a></code> com um atributo <code>target</code> diferente de <code>"_blank"</code>	Exemplo
<code>\$(":button")</code>	Seleciona todos os elementos do tipo <code><button></code> e os elementos do tipo <code><input></code> que tem um atributo <code>type="button"</code>	Exemplo

³⁸ http://www.w3schools.com/cssref/css_selectors.asp

<code>\$("tr:even")</code>	Seleciona todos os elementos <tr> pares (0,2,4,6....)	Exemplo
<code>\$("tr:odd")</code>	Seleciona todos os elemento <tr> ímpares (1,3,5,7...)	Exemplo

Eventos

Todas as ações de um usuário que uma página pode responder são chamadas de eventos. Um evento representa o momento em que uma determinada ação ocorre. Por Exemplo:

- Mover o mouse sobre um elemento
- Selecionar um radio button
- Clicar em um elemento

O termo “disparado” (o termo em inglês é “fires”) é usado com frequência para descrever quando um evento ocorre. Por exemplo: o evento “keypress” é disparado no momento em que o usuário pressiona uma tecla. Estes são alguns eventos comum do DOM.

Evento	Descrição	Exemplo
Eventos de Mouse		
click	Quando o usuário clica sobre um elemento	<pre><code>\$("p").click(function() { \$(this).hide(); });</code></pre>
dblclick	Quando o usuário dá um clique duplo sobre um elemento	<pre><code>\$("p").dblclick(function() { \$(this).hide(); });</code></pre>
mouseenter	Quando o mouse entra na área definida por um elemento	<pre><code>\$("#p1").mouseenter(function() { alert("You entered p1!"); });</code></pre>
mouseleave	Quando o mouse sai da área definida por um elemento	<pre><code>\$("#p1").mouseleave(function() { alert("Bye! You now leave p1!"); });</code></pre>
mousedown	Quando o usuário pressiona o botão esquerdo do mouse enquanto o mouse está sobre um elemento.	<pre><code>\$("#p1").mousedown(function() { alert("Mouse down over p1!"); });</code></pre>
mouseup	Quando o usuário solta o botão esquerdo do mouse, depois de ter clicado em um elemento.	<pre><code>\$("#p1").mouseup(function() { alert("Mouse up over p1!"); });</code></pre>
hover	Quando o usuário passa o mouse sobre um elemento. Possui dois parâmetros, a função que é executada quando o mouse entra na área definida pelo elemento e a outra função executada quando o mouse sai da área definida por um elemento.	<pre><code>\$("#p1").hover(function() { alert("You entered p1!"); }, function() { alert("Bye! You now leave p1!"); });</code></pre>
Eventos de Teclado		
keypress	Quando o usuário pressionou uma tecla.	<pre><code>\$("input").keypress(function() { \$("span").text(i += 1); });</code></pre>
keydown	Quando o usuário está pressionando uma tecla.	<pre><code>\$("input").keydown(function() { \$("input").css("background-color", "yellow"); });</code></pre>

keyup	Quando o usuário está soltando uma tecla depois de pressionar.	<code>\$("#input").keyup(function(){ \$("#input").css("background-color", "pink"); });</code>
Eventos de Formulário		
submit	Quando o formulário é submetido.	<code>\$("#form").submit(function(){ alert("Submitted"); });</code>
change	Quando o conteúdo de algum elemento de um formulário é alterado.	<code>\$("#input").change(function(){ alert("The text has been changed."); });</code>
focus	Quando um elemento recebe o foco.	<code>\$("#input").focus(function(){ \$(this).css("background-color", "#cccccc"); });</code>
blur	Quando um elemento perde o foco.	<code>\$("#input").blur(function(){ \$(this).css("background-color", "#ffffff"); });</code>
Eventos de documento e janela		
load	Quando a página é carregada.	<code>\$("#img").load(function(){ alert("Image loaded."); });</code>
resize	Quando a janela é redimensionada.	<code>\$(window).resize(function(){ \$('span').text(x += 1); });</code>
scroll	Quando é feita rolagem do conteúdo da página.	<code>\$("#div").scroll(function(){ \$("span").text(x += 1); });</code>

Para uma lista completa dos eventos que o jQuery fornece consulte [esse link](#)³⁹.

Método genérico para eventos

O método "on" do jQuery adiciona um ou mais eventos aos elementos selecionados. Você pode informar o nome do evento que deseja e a função que deverá ser executada:

```
$("#p").on("click", function(){  
    $(this).hide();  
});
```

[Clique aqui para testar.](#)

Um exemplo adicionado mais de um evento ao mesmo tempo:

³⁹ http://www.w3schools.com/jquery/jquery_ref_events.asp

```
$("#p").on({  
  mouseenter: function(){  
    $(this).css("background-color", "lightgray");  
  },  
  mouseleave: function(){  
    $(this).css("background-color", "lightblue");  
  },  
  click: function(){  
    $(this).css("background-color", "yellow");  
  }  
});
```

[Clique aqui para testar](#)

Módulo 3 - jQuery HTML

“Sometimes the questions are complicated and the answers are simple”

L - Death Note

jQuery fornece uma forma simples, rápida e poderosa de manipular o HTML da página. Ele trás diversos métodos simples e fáceis de usar que lhe permitem alterar qualquer elemento de sua página.

Obtendo e alterando valores e atributos

Manipulando Texto, HTML e Valor

Caso você queira alterar o texto, html (innerHTML) ou valor de um elemento (input) você pode usar esses métodos:

- **text()**: Usado para retornar ou para definir o texto de um elemento.
- **html()**: Usado para retornar ou para definir o HTML de um elemento (incluindo código HTML, é equivalente ao innerHTML do JS nativo).
- **val()**: Usado para retornar ou para definir o valor de um elemento HTML que é um campo de formulário.

Exemplos:

Obtendo o texto e o HTML de elementos:

```
$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});
```

Definindo o texto, html e valor de elementos:

```
$("#btn1").click(function(){
    $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
});
```

Manipulando atributos

Caso você precise obter ou definir um determinado atributo de um elemento HTML, você pode usar o método **attr()**. Ele funciona para qualquer atributo HTML.

Exemplos:

Obtendo o valor do atributo href de um elemento:


```
$("#button").click(function(){  
    alert($("#w3s").attr("href"));  
});
```

Definindo o valor do atributo href de um elemento:

```
$("#button").click(function(){  
    $("#w3s").attr("href", "http://www.w3schools.com/jquery");  
});
```

Você também pode definir mais de um atributo por vez:

```
$("#button").click(function(){  
    $("#w3s").attr({  
        "href" : "http://www.w3schools.com/jquery",  
        "title" : "W3Schools jQuery Tutorial"  
    });  
});
```

Adicionando elementos e conteúdo

Você pode usar a função **append()** para adicionar conteúdo ao final do elemento selecionado.

```
$("#p").append("Some appended text.");
```

Caso deseje adicionar conteúdo ao início do elemento selecionado, você pode usar a função **prepend()**.

```
$("#p").prepend("Some prepended text.");
```

Ambos os métodos podem ser usados para adicionar mais de um elemento por vez. Para fazer isso, basta passar por parâmetro todos os elementos que deseja adicionar, um após o outro.

```
function appendText() {  
    var txt1 = "<p>Text.</p>";           // Create element with HTML  
    var txt2 = $("<p></p>").text("Text."); // Create with jQuery  
    var txt3 = document.createElement("p"); // Create with DOM  
    txt3.innerHTML = "Text.";            // Append the new elements  
    $("#p").append(txt1, txt2, txt3);  
}
```

Os métodos **append()** e **prepend()** adicionam o novo conteúdo respectivamente ao final e início do elemento selecionado, ou seja, adicionam conteúdo **dentro** de um elemento. Mas caso você precise adicionar conteúdo **antes** ou **depois** de um elemento, você pode usar as funções **before()** e **after()** respectivamente.

```
$("#img").after("Some text after");  
  
$("#img").before("Some text before");
```

Essas funções irão adicionar o novo conteúdo antes ou depois do novo elemento (mas fora dele). Elas também podem ser usadas para adicionar mais de um elemento simultaneamente, funcionando da mesma forma:

```
function afterText() {  
    var txt1 = "<b>I </b>";           // Create element with HTML  
    var txt2 = $("<i></i>").text("love "); // Create with jQuery  
    var txt3 = document.createElement("b"); // Create with DOM  
    txt3.innerHTML = "jQuery!";  
    $("img").after(txt1, txt2, txt3); // Insert new elements after <img>  
}
```

Removendo elementos

Caso você queira remover um elemento da DOM HTML use o método **remove()**.

```
$("#div1").remove();
```

Caso você queira “limpar” um elemento HTML, removendo todos os seus filhos, use o método **empty()**.

```
$("#div1").empty();
```

Ao remover elemento com o remove, você também pode passar por parâmetro um seletor jQuery para especificar quais elementos você deseja remover.

Exemplo: Remover todos os elemento com classe CSS “test”:

```
$("p").remove(".test");
```

Exemplo: Remover todos os elementos com classe CSS “test” ou “demo”:

```
$("p").remove(".test, .demo");
```

Manipulando classes CSS

Você também pode manipular as classes CSS que um elemento tem atribuídas, alterando seu estilo sem recarregar a página para isso.

Caso você queira adicionar ou remover classes CSS, use os métodos **addClass()** e **removeClass()** respectivamente.

Adicionando classes CSS:

```
$("#button").click(function(){  
    $("h1, h2, p").addClass("blue");  
    $("div").addClass("important");  
});
```

Adicionando mais de uma classe CSS:

```
$("#button").click(function(){  
    $("#div1").addClass("important blue");  
});
```

Removendo classes CSS:

```
$("#button").click(function(){  
    $("h1, h2, p").removeClass("blue");  
});
```

Você também pode alternar entre adicionar e remover uma classe CSS usando o método **toggleClass()**. Caso o elemento já tenha a classe CSS informada, ela será removida, e caso não tenha, ela será adicionada.

```
$("#button").click(function(){  
    $("h1, h2, p").toggleClass("blue");  
});
```

Alterando propriedades CSS

Caso você deseje manipular propriedades CSS separadamente, você pode usar o método **css()**. Ele funciona como os outros métodos `val()`, `text()` e `html()`, permitindo obter ou definir uma propriedades CSS.

Exemplos:

Obtendo o valor da propriedade CSS "background-color":

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

Definindo o valor da propriedade CSS "background-color":

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

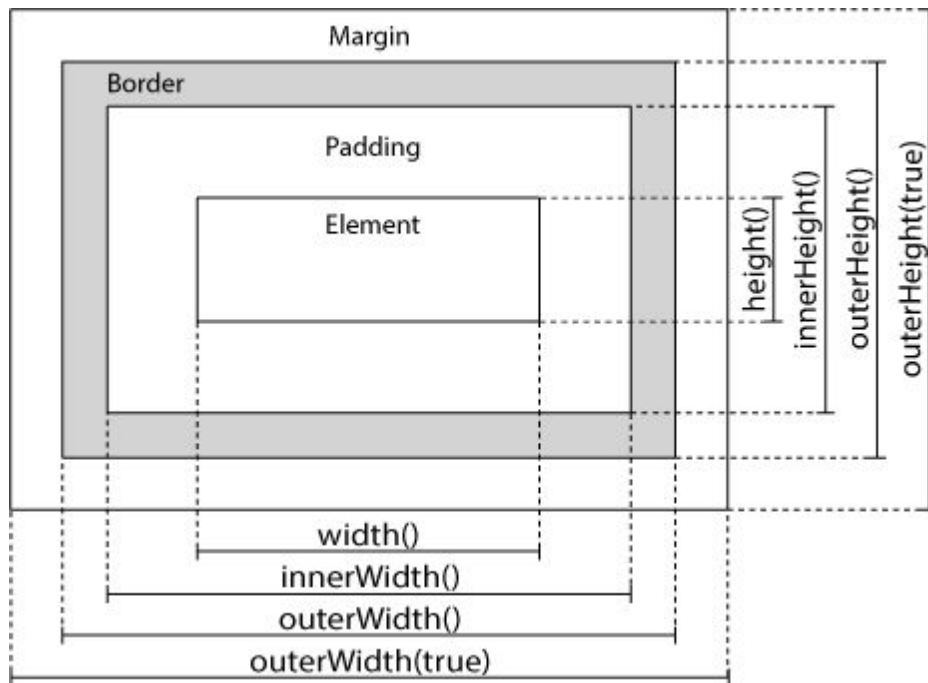
Você também pode definir múltiplas propriedades CSS de uma única vez:

```
$("#p").css({"background-color": "yellow", "font-size": "200%"});
```

Manipulando propriedades de dimensão

jQuery também oferece um conjunto de métodos que permitem consultar e definir as dimensões de um elemento. Eles funcionam como o `val()`, `text()` e `html()`.

Para compreender como usar esses métodos, antes você precisa conhecer o conceito de Box Model.



Para consultar ou alterar as propriedades de dimensão de um elemento, use o seguintes métodos:

- **width() e height():** para consultar ou alterar a largura e altura de um elemento (sem padding, border, ou margin).
 - Obtendo largura e altura:

```
$("#button").click(function(){
    var txt = "";
    txt += "Width: " + $("#div1").width() + "<br>";
    txt += "Height: " + $("#div1").height();
    $("#div1").html(txt);
});
```

- Definindo largura e altura:

```
$("#button").click(function(){
    $("#div1").width(500).height(500);
});
```

- **innerWidth() e innerHeight():** para consultar a largura e altura de um elemento, incluindo o espaço ocupado pelo padding.

```
$("#button").click(function(){
    var txt = "";
    txt += "Inner width: " + $("#div1").innerWidth() + "<br>";
    txt += "Inner height: " + $("#div1").innerHeight();
    $("#div1").html(txt);
});
```

-
- **outerWidth() e outerHeight():** para consultar a largura e altura de um elemento, incluindo o espaço ocupado pelo padding e border.

```
$("#button").click(function(){
    var txt = "";
    txt += "Outer width: " + $("#div1").outerWidth() + "<br>";
    txt += "Outer height: " + $("#div1").outerHeight();
    $("#div1").html(txt);
});
```

- **outerWidth(true)** e **outerHeight(true)**: para consultar a largura e altura de um elemento, incluindo o espaço ocupado pelo padding, border e margin.

```
$("#button").click(function(){
    var txt = "";
    txt += "Outer width (+margin): " + $("#div1").outerWidth(true) + "<br>";
    txt += "Outer height (+margin): " + $("#div1").outerHeight(true);
    $("#div1").html(txt);
});
```

Você também pode usar **width()** e **height()** para consultar a largura e altura do documento HTML e da janela do navegador.

```
$("#button").click(function(){
    var txt = "";
    txt += "Document width/height: " + $(document).width();
    txt += "x" + $(document).height() + "\n";
    txt += "Window width/height: " + $(window).width();
    txt += "x" + $(window).height();
    alert(txt);
});
```

Par ver mais opções de manipulação de HTML e CSS através do jQuery consulte [esse link](#)⁴⁰.

Armazenando dados em elementos HTML

Com o HTML 5 você pode usar os atributos “data-*” para armazenar informações complementares sobre um elemento HTML e com isso fornecer uma experiência de usuário mais rica, sem que seja necessário efetuar mais requisições ao servidores para buscar mais informações.

- Você pode criar quantos atributos “data-*” quiser.
- O nome do atributo deve ser “data-*”. Onde * deve ser pelo menos uma letra.
- O nome do atributo deve ser sempre em minúsculas.
- O valor do atributo deve ser uma string ou um objeto JSON.
- Todos os atributos “data-*” são ignorados pelo navegador.

Sintaxe:

```
<element data-*= "somevalue">
```

Com jQuery você pode acessar e alterar os valores desses atributos. Para isso use o método **.data()**.

⁴⁰ http://www.w3schools.com/jquery/jquery_ref_html.asp

Para atribuir um valor

```
$(selector).data(name, value)
```

O parâmetro “name” será a chave usada para armazenar o valor definido no parâmetro “value”. Exemplo:

```
$("#btn1").click(function(){  
    $("div").data("greeting", "Hello World");  
});
```

Nesse exemplo o valor “Hello World” está sendo armazenado na chave “greeting”.

Para obter um valor

```
$(selector).data(name)
```

O parâmetro “name” será a chave usada para armazenar o valor dentro do elemento. Exemplo:

```
$("#btn2").click(function(){  
    alert($("div").data("greeting"));  
});
```

Nesse exemplo estamos obtendo o valor do atributo com a chave “greeting”, retornando o valor “Hello World”

Para remover um valor

```
$(selector).removeData(name)
```

O parâmetro “name” é chave que foi usada para salvar o valor. Exemplo:

```
$("#btn2").click(function(){  
    $("div").removeData("greeting");  
    alert("Greeting is: " + $("div").data("greeting"));  
});
```

Nesse exemplo o alert irá mostrar “undefined” pois o valor foi removido.

Para mais informações sobre os atributos “data” do HTML 5, sugerimos a leitura [desse artigo](http://webdesign.tutsplus.com/tutorials/all-you-need-to-know-about-the-html5-data-attribute--webdesign-9642)⁴¹.

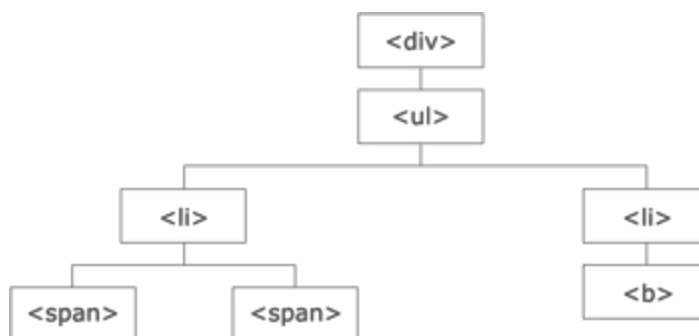
⁴¹ <http://webdesign.tutsplus.com/tutorials/all-you-need-to-know-about-the-html5-data-attribute--webdesign-9642>

Modulo 4 - jQuery Traversing

“The greatest weakness of most humans is their hesitancy to tell others how they love them while they're alive”

Optimus Prime - Transformers

jQuery também fornece métodos que lhe permitem navegar pela hierarquia de elementos da DOM HTML. Essas funções são conhecidas como jQuery Traversing pois lhe permitem “atravessar” pelos elementos da DOM. Antes de estudarmos os métodos que nos permitem fazer isso, vamos definir a nomenclatura. Tome essa imagem como exemplo:



Neste exemplo:

- o DIV é o pai do UL, e o antecessor de tudo que ele contém.
- UL é pai dos dois LI e filho de DIV.
- O LI esquerdo é pai dos dois SPAN, filho do UL e descendente do DIV.
- Os dois LI são irmãos, pois compartilham o mesmo pai.

Generalizando:

- Um antecessor é um pai, avo, bisavô e assim por diante.
- Um descendente é um filho, neto, bisneto e assim por diante.
- Irmão são aqueles que compartilham um mesmo pai.

Navegando para cima na hierarquia da DOM

O método **parent()** retorna o pai direto do elemento selecionado:

```
$(document).ready(function(){  
    $("span").parent();  
});
```

O método **parents()** retorna todos os antecessores do elemento selecionado até a tag <html>:

```
$(document).ready(function(){  
    $("span").parents();  
});
```

Você também pode passar por parâmetro uma string para filtrar os elementos que quer retornar:


```
$(document).ready(function(){  
    $("span").parents("ul");  
});
```

Caso você não queira todos os antecessores do elemento até a tag <html>, você pode usar o método **parentsUntil()** para especificar até que elemento você deseja navegar.

```
$(document).ready(function(){  
    $("span").parentsUntil("div");  
});
```

Navegando para baixo na hierarquia da DOM

O método **children()** retorna todos os filhos diretos do elemento selecionado.

```
$(document).ready(function(){  
    $("div").children();  
});
```

Opcionalmente você pode passar um parâmetro contendo um seletor jQuery para especificar quais filhos você quer obter. Nesse exemplo são retornados todos os elemento <p> que tem a classe CSS "1" e que são filhos diretos do <div> selecionado.

```
$(document).ready(function(){  
    $("div").children("p.1");  
});
```

O método **find()** permite pesquisar por descendentes de um elemento através de um seletor jQuery. Retornando todos os elementos que são descendentes do elemento <div> selecionado:

```
$(document).ready(function(){  
    $("div").find("span");  
});
```

Use "*" para retornar todos os descendentes do elemento selecionado:

```
$(document).ready(function(){  
    $("div").find("*");  
});
```

Navegando para os lados na hierarquia DOM

- **siblings()**: retorna todos os elemento irmãos do elemento selecionado.
 - Buscando todos os irmão do elemento selecionado.

```
$(document).ready(function(){  
    $("h2").siblings();  
});
```

- Você também pode passar por parâmetro um seletor jQuery para filtrar os irmão retornados. Nesse exemplo são retornados todos os elementos <p> que são irmãos do elemento selecionado:

```
$(document).ready(function(){
    $("h2").siblings("p");
});
```

- **next()**: retorna o próximo elemento irmão do elemento selecionado:

```
$(document).ready(function(){
    $("h2").next();
});
```

- **nextAll()**: retorna todos os próximos elementos irmãos do elemento selecionado.

```
$(document).ready(function(){
    $("h2").nextAll();
});
```

- **nextUntil()**: retorna todos os irmãos que estão entre o elemento selecionado e o segundo elemento passado parâmetro.

```
$(document).ready(function(){
    $("h2").nextUntil("h6");
});
```

- **prev()**: mesmo que next() mas retorna o irmão anterior ao elemento selecionado.
- **prevAll()**: mesmo que nextAll() mas retorna todos os elementos irmãos que estão antes do elemento selecionado.
- **prevUntil()**: mesmo que nextUntil() mas retorna os elementos que estão entre o elemento passado por parâmetro e o elemento selecionado.

Pesquisando elementos

Esses métodos permitem selecionar elemento baseados na posição em que eles ocupam na DOM HTML.

- **first()**: retorna o primeiro elemento dentre os elementos selecionados.
 - Retorna o primeiro <p> dentro do primeiro <div>

```
$(document).ready(function(){
    $("div p").first();
});
```

- **last()**: retorna o último elemento dentre os elementos selecionado.
 - Retorna o último <p> dentro do último <div>

```
$(document).ready(function(){
    $("div p").last();
});
```

- **eq()**: retorna um elemento baseado na posição (ordem/índice) que ele ocupa. O índice começa em 0 (zero).
 - Retorna o segundo <p>

```
$(document).ready(function(){
    $("p").eq(1);
});
```

- **filter()**: permite filtrar por elementos a partir de uma seleção inicial utilizando um critério (seletor).
 - Retorna todos os elemento <p> que contém um classe CSS "intro".

```
$(document).ready(function(){
    $("p").filter(".intro");
});
```

- **not()**: retorna todos os elementos que não atendem ao critério de pesquisa especificado.
 - Retorna todos os elementos <p> que não tem uma classe CSS "intro".

```
$(document).ready(function(){
    $("p").not(".intro");
});
```

Para consultar mais métodos de navegação pela DOM consulte [esse link](#)⁴².

Seletores: Mais exemplos e formas de usar

Veja abaixo uma tabela de referência dos seletores disponíveis para jQuery.

Seletor	Exemplo	Descrição do que é selecionado
*	<code>\$("*")</code>	Todos os elementos
#id	<code>\$("#lastname")</code>	O elemento com o id="lastname"
.class	<code>\$(".intro")</code>	Todos os elementos com a classe CSS class="intro"
.class.class	<code>\$(".intro,.demo")</code>	Todos os elementos com a classe CSS "intro" ou "demo"
element	<code>\$("p")</code>	Todos os elementos HTML com tag <p>
el1.el2.el3	<code>\$("h1,div,p")</code>	Todos os <h1>, <div> <p>
:first	<code>\$("p:first")</code>	O primeiro elemento <p>
:last	<code>\$("p:last")</code>	O último elemento <p>
:even	<code>\$("tr:even")</code>	Todos os elementos <tr> que são pares
:odd	<code>\$("tr:odd")</code>	Todos os elementos <tr> que são ímpares
:first-child	<code>\$("p:first-child")</code>	Todos os elementos <p> que são o primeiro filho de seus pais
:first-of-type	<code>\$("p:first-of-type")</code>	Todos os elementos <p> que são o primeiro <p> do seu pai
:last-child	<code>\$("p:last-child")</code>	Todos os elementos <p> que são o último filho de seu pai

⁴² http://www.w3schools.com/jquery/jquery_ref_traversing.asp

:last-of-type	<code>\$("p:last-of-type")</code>	Todos os elemento <p> que são o último <p> do seu pai
:nth-child(n)	<code>\$("p:nth-child(2)")</code>	Todos os elementos <p> que são o segundo filho do seu pai
:nth-last-child(n)	<code>\$("p:nth-last-child(2)")</code>	Todos os elementos <p> que são o segundo filho do seu pai, contando a partir do último filho
:nth-of-type(n)	<code>\$("p:nth-of-type(2)")</code>	Todos os elementos <p> que são o segundo <p> do seu pai.
:nth-last-of-type(n)	<code>\$("p:nth-last-of-type(2)")</code>	Todos os elementos <p> que são o segundo <p> do seu pai, contando a partir o último filho
:only-child	<code>\$("p:only-child")</code>	Todos o <p> que são o único filho de seus pais
:only-of-type	<code>\$("p:only-of-type")</code>	Todos os elementos <p> que são o único <p> de seus pais.
parent > child	<code>\$("div > p")</code>	Todos os elemento <p> que são filhos diretos de um elemento <div>
parent descendant	<code>\$("div p")</code>	Todos os elemento <p> que são descendentes de um elemento <div>
element + next	<code>\$("div + p")</code>	O elemento <p> que está ao lado de um elemento <div>
element ~ siblings	<code>\$("div ~ p")</code>	Todos os elementos <p> que são irmãos de um elemento <div>
:eq(index)	<code>\$("ul li:eq(3)")</code>	O quarto elemento de uma lista (o índice começa em zero)
:gt(no)	<code>\$("ul li:gt(3)")</code>	Elementos de uma lista cujo índice é maior do que 3
:lt(no)	<code>\$("ul li:lt(3)")</code>	Elementos de uma lista cujo índice é menor do que 3
:not(selector)	<code>\$("input:not(:empty)")</code>	Todos os elemento <input> que não estão vazios
:header	<code>\$(":header")</code>	Todos os elementos de título (h1, h2, h3, etc)
:animated	<code>\$(":animated")</code>	Todos os elementos que tem uma animação
:focus	<code>\$(":focus")</code>	O elemento que atualmente tem o foco
:contains(text)	<code>\$(":contains('Hello')")</code>	Todos os elementos que contém o texto "Hello"
:has(selector)	<code>\$("div:has(p)")</code>	Todos os elementos <div> que contém um elemento <p>
:empty	<code>\$(":empty")</code>	Todos os elementos que estão vazios
:parent	<code>\$(":parent")</code>	Todos os elementos que são pais de algum elemento
:hidden	<code>\$("p:hidden")</code>	Todos os elementos <p> que estão ocultos
:visible	<code>\$("table:visible")</code>	Todas as tabelas que estão visíveis
:root	<code>\$(":root")</code>	Elemento raiz do documento
:lang(language)	<code>\$("p:lang(de)")</code>	Todos os elemento <p> com um atributo "lang" cujo valor começa com "de".

[attribute]	<code>\$("[href]")</code>	Todos os elementos com um atributo href.
[attribute=value]	<code>\$("[href='default.htm']")</code>	Todos os elementos com um atributo href cujo valor é "default.htm"
[attribute!=value]	<code>\$("[href!='default.htm']")</code>	Todos os elementos com um atributo href cujo valor não é "default.htm"
[attribute\$=value]	<code>\$("[href\$='.jpg']")</code>	Todos os elementos com um atributo href cujo valor termina em ".jpg"
[attribute =value]	<code>\$("[title='Tomorrow']")</code>	Todos os elemento com um atributo cujo valor é 'Tomorrow', ou começa com 'Tomorrow' seguido de um hífen.
[attribute^=value]	<code>\$("[title^='Tom']")</code>	Todos os elemento com um atributo title cujo valor começa com "Tom".
[attribute~=value]	<code>\$("[title~='hello']")</code>	Todos os elemento com um atributo title cujo valor é igual a "hello"
[attribute*=value]	<code>\$("[title*='hello']")</code>	Todos os elementos com um atributo title cujo valor contém a palavra "hello"
:input	<code>\$(":input")</code>	Todos os elementos de input.
:text	<code>\$(":text")</code>	Todos os elemento de input cujo type="text".
:password	<code>\$(":password")</code>	Todos os elemento de input cujo type="password"
:radio	<code>\$(":radio")</code>	Todos os elemento de input cujo type="radio"
:checkbox	<code>\$(":checkbox")</code>	Todos os elemento de input cujo type="checkbox"
:submit	<code>\$(":submit")</code>	Todos os elemento de input cujo type="submit"
:reset	<code>\$(":reset")</code>	Todos os elemento de input cujo type="reset"
:button	<code>\$(":button")</code>	Todos os elemento de input cujo type="button"
:image	<code>\$(":image")</code>	Todos os elemento de input cujo type="image"
:file	<code>\$(":file")</code>	Todos os elemento de input cujo type="file"
:enabled	<code>\$(":enabled")</code>	Todos os elementos de input que estão habilitados.
:disabled	<code>\$(":disabled")</code>	Todos os elementos de input que estão desabilitados.
:selected	<code>\$(":selected")</code>	Todos os elementos de input que estão selecionados.
:checked	<code>\$(":checked")</code>	Todos os elementos de input que estão checados.

Você também pode usar [esse link](#)⁴³ para testar alguns seletores do jQuery. Clique sobre o nome de algum dos seletores que são mostrados no link para que o elemento HTML correspondente seja destacado em amarelo.

⁴³ <http://www.w3schools.com/jquery/tryse.asp>

Módulo 5 - jQuery Effects

“If you can dream it you can do it”

Walt Disney

Através do jQuery você pode aplicar diversos efeitos em seus componentes HTML. Efeitos permitem que você torne sua página mais dinâmica, dando um melhor feedback as ações realizadas pelos usuários. Vamos ver agora alguns efeitos que o jQuery nos fornece.

Esconder e Mostrar

Hide e Show

Com jQuery você pode esconder elementos através do método “hide()” e mostrá-los através do método “show()”. Exemplo:

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_hide_show

A sintaxe geral desses comandos é a seguinte:

```
$(selector).hide(speed,callback);

$(selector).show(speed,callback);
```

- **selector:** é o elemento sobre o qual você deseja aplicar o efeito. Você pode usar os comandos de seletores que já foram vistos para selecionar os elementos que deseja animar.
- **speed:** é um parâmetro opcional que lhe permite informar a duração da animação em milissegundos, ou então você pode passar as palavras “slow” (lento) e “fast” (rápido).
- **callback:** é um parâmetro opcional que lhe permite passar uma função que deverá ser executada assim que o evento for finalizado.

A maioria dos comandos de efeito do jQuery possuem os mesmos parâmetros que o exemplo acima. Nos próximos exemplos, considere que os parâmetros tem as mesmas opções e comportamentos desse exemplo. Se houver alguma exceção ou particularidade ela será descrita explicitamente.

Toggle

Com jQuery você também pode usar o método “toggle()” para mostrar ou esconder elementos. Esse método irá inverter o estado atual de um elemento, ou seja, se ele estiver aparecendo, ele será escondido, e se estiver escondido, ele será mostrado novamente.

```
$("#button").click(function(){
    $("p").toggle();
});
```


http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_toggle

A sintaxe do comando é a seguinte:

```
$(selector).toggle(speed, callback);
```

Os parâmetros funcionam da mesma forma que nos métodos show() e hide().

Desvanecer

Fade In

Usado para mostrar elementos escondidos aplicando um efeito de desvanecimento.

Sintaxe:

```
$(selector).fadeIn(speed, callback);
```

Exemplo:

```
$("#button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadein

Fade Out

Usado para ocultar elementos que estão sendo exibidos, aplicando um efeito de desvanecimento.

Sintaxe:

```
$(selector).fadeOut(speed, callback);
```

Exemplo:

```
$("#button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadeout

Fade Toggle

Usado para alternar a situação atual de um elemento, ou seja, caso o elemento tenha sido escondido por fadeOut, ele será exibido novamente com efeito de fadeIn, e caso ele tenha sido exibido com fadeIn, ele será ocultado usando efeito de fadeOut.

Sintaxe:

```
$(selector).fadeToggle(speed, callback);
```

Exemplo:


```
$("#button").click(function(){  
    $("#div1").fadeToggle();  
    $("#div2").fadeToggle("slow");  
    $("#div3").fadeToggle(3000);  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadetoggle

Fade To

Esse método permite que você esmaieça um elemento até uma determinada opacidade, que você irá escolher.

Sintaxe:

```
$(selector).fadeTo(speed, opacity, callback);
```

Nesse caso, os parâmetros “speed” e “opacity” são obrigatórios. “Callback” continua sendo opcional. O parâmetro “opacity” deve ser um número entre 0 e 1, quanto mais próximo de 0 (zero), mais esmaecido o elemento irá ficar. Exemplos:

```
$("#button").click(function(){  
    $("#div1").fadeTo("slow", 0.15);  
    $("#div2").fadeTo("slow", 0.4);  
    $("#div3").fadeTo("slow", 0.7);  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadeto

Escorregar

Slide Down

Escorregar um elemento para baixo. Sintaxe:

```
$(selector).slideDown(speed, callback);
```

Exemplo:

```
$("#flip").click(function(){  
    $("#panel").slideDown();  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_slide_down

Slide Up

Escorregar um elemento para cima. Sintaxe:

```
$(selector).slideUp(speed, callback);
```

Exemplo:

```
$("#flip").click(function(){  
    $("#panel").slideUp();  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_slide_up

Slide Toggle

Escolher um elemento de forma inversa ao último efeito. Ou seja, se o elemento foi animado com slideUp, ele será animado com slideDown, e se foi animado com slideDown, ele será animado com slideUp.. Sintaxe:

```
$(selector).slideToggle(speed,callback);
```

Exemplo:

```
$("#flip").click(function(){  
    $("#panel").slideToggle();  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_slide_toggle

Animações

Introdução

Além dessas animações pré-definidas, jQuery permite que você crie animações personalizadas, através do método animate(). Sintaxe:

```
$(selector).animate({params}, speed, callback);
```

- O parâmetro "params" é obrigatório. Nele você pode passar uma lista de propriedades CSS que você deseja alterar nessa animação.
- Os demais parâmetros (speed e callback) são opcionais e funcionam da mesma forma já descrita nos outros efeitos.

Veja um exemplo simples que mostra um DIV sendo animado, posicionando-o 250px a esquerda da sua posição atual:

```
$("#button").click(function(){  
    $("#div").animate({left: '250px'});  
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation1

Importante: Por padrão todos os elementos HTML tem uma posição estática e não podem ser movidos. Para manipular a posição de um elemento HTML, antes altere a propriedade "position" para "relative", "fixed", ou "absolute".

```
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">  
</div>
```

Animando múltiplas propriedades

Você pode manipular múltiplas propriedades CSS ao mesmo tempo:

```
$("#button").click(function(){
    $("#div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
    });
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation1_multicss

Posso animar todas as propriedades CSS?

Sim, quase todas. Mas é importante observar que os nomes das propriedades devem ser escritos no padrão “camel case”, ou seja, você terá de escrever `paddingLeft`, ao invés de `padding-left`, e assim por diante. A animação de cores não está incluída na biblioteca jQuery por padrão, mas se você quiser animar cores, poderá baixar o plugin [jQuery Cores](#)⁴⁴

Animações relativas

Você criar animações passando valores relativos (ao valor corrente do elemento). Isso é feito adicionando os caracteres de “+” ou “-” antes do valor da propriedade.

Exemplo:

```
$("#button").click(function(){
    $("#div").animate({
        left: '250px',
        height: '+=150px',
        width: '+=150px'
    });
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation1_relative

Usando valores pré-definidos

Você especificar o valor de uma propriedade como “toggle”, “show” ou “hide”. Exemplo:

```
$("#button").click(function(){
    $("#div").animate({
        height: 'toggle'
    });
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation1_toggle

Fila de Animações

Internamente o jQuery usa uma fila para ordenar todas as animações. Se você aplicar mais de uma animação sobre um mesmo elemento, o jQuery irá colocá-las na fila e executar uma por uma. Veja esse exemplo:

⁴⁴ <http://plugins.jquery.com/color/>

```
$("#button").click(function(){
    var div = $("#div");
    div.animate({height: '300px', opacity: '0.4'}, "slow");
    div.animate({width: '300px', opacity: '0.8'}, "slow");
    div.animate({height: '100px', opacity: '0.4'}, "slow");
    div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation

Parando animações

Caso você queira parar uma animação que está em andamento, antes que ela termine, você pode usar o método `stop()`. Sintaxe:

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

- O parâmetro "stopAll" é opcional, e indica se você quer que todas as animações que estão na fila de animações desse elemento sejam canceladas também. O padrão é "false", então se você chamar `stop()` sem passar o primeiro parâmetro, apenas animação atualmente sendo executada será interrompida, as demais animações continuarão na fila e poderão ser executadas depois.
- O parâmetro "goToEnd" é opcional e indica se a animação atual deve ser completada imediatamente (ir para seu estado final) ou não. O padrão é "false", isso significa que por padrão a animação é interrompida e fica parada no estado em que estava quando `stop()` foi chamado.

Exemplo:

```
$("#stop").click(function(){
    $("#panel").stop();
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_stop_slide

Funções de Callback

Os comandos em JavaScript são executados linha por linha. No entanto, os efeitos são executados de forma assíncrona. Isso faz com que a linha de código seguinte possa ser executada antes que uma animação tenha terminado. Isso pode ocasionar erros. Para prevenir isso você pode usar funções de callback. Elas são funções que serão executadas depois que animação tiver terminado. Veja nesse exemplo, depois que o parágrafo for oculto, é emitido um alerta.

```
$("#button").click(function(){
    $("#p").hide("slow", function(){
        alert("The paragraph is now hidden");
    });
});
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_hide_callback

Encadeamento

Há uma técnica chamada de encadeamento que lhe permite executar vários comandos jQuery em uma única linha de código. Isso é útil pois evita que o navegador tenha que encontrar o mesmo elemento HTML mais de uma vez, além de facilitar o desenvolvimento. Para usar o encadeamento você precisa apenas adicionar um "." depois de um comando, e escrever o próximo comando. Veja um exemplo:

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_chaining

Você também pode formatar o código e colocar um comando por linha, para facilitar a leitura:

```
$("#p1").css("color", "red")  
  .slideUp(2000)  
  .slideDown(2000);
```

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_chaining2

Para ver uma lista dos efeitos disponíveis no jQuery consulte [esse link](#)⁴⁵.

⁴⁵ http://www.w3schools.com/jquery/jquery_ref_effects.asp

Módulo 6 - jQuery AJAX

“With great power comes great responsibility”

Uncle Bean - Spiderman

AJAX é uma técnica que permite carregar um conteúdo e atualizar sua página com esse conteúdo sem a necessidade de recarregar toda a página.

Carregando um conteúdo para um elemento

A função mais simples do jQuery para fazer AJAX é o método “load()”. Ele permite que você carregue uma página dinamicamente para dentro de um outro elemento de sua página. Sintaxe:

```
$(selector).load(URL, data, callback);
```

- O parâmetro “selector” é um seletor válido do jQuery, e será dentro desse elemento selecionado que o conteúdo carregado será exibido.
- “URL” é o endereço que você deseja carregar.
- “data” é um parâmetro opcional no qual você pode passar parâmetros para a requisição que será feita.
- “callback” também é opcional e você pode passar uma função para ser executada quando a requisição AJAX tiver terminado.

Exemplo:

Suponha que o seguinte código HTML esteja hospedado na URL “http://jsbin.com/cuzesu”.

```
<h2>Notícias Principais</h2>
<ul>
  <li>Grécia é anunciada para venda no MercadoLiv</li>
  <li>Zoo escolhe nome do seu novo filhote</li>
  <li>Papa Francisco faz viagem sagrada</li>
</ul>
<h2>Notícias Locais</h2>
<ul id="noticias-locais">
  <li>RS-122 tem mais buracos do que nunca</li>
  <li>Começa FestiQueijo em Carlos Barbosa</li>
  <li>Morre cantor Nico Fagundes</li>
</ul>
```

Vamos carregar dinamicamente essa lista de notícias localizada nesse endereço.

```
1 $('#carregar').click(function(){
2   $('#noticias').load('http://jsbin.com/cuzesu');
3 });
```

Nesse exemplo, ao clicar no link “Carregar”, uma lista de notícias é carregada dinamicamente para o DIV com ID “noticias”.

Você também pode especificar um seletor após a URL, para buscar apenas um elemento dentro da página HTML que você está carregando. Exemplo:

```
$('#carregar').click(function(){
  $('#manchete').load('http://jsbin.com/cuzesu #noticias-locais');
});
```

Note que dessa vez, ao final da URL especificamos que deve ser carregado apenas o conteúdo do DIV com ID “noticias-locais”.

A função de callback que você pode passar ao executar “load” irá lhe fornecer três parâmetros. Exemplo:

```
$("#button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

- responseTxt: irá lhe fornecer o conteúdo da requisição que foi feita.
- statusTxt: irá lhe retornar uma string com os valores “success” em caso de sucesso ou “error” caso a requisição tenha falhado por algum motivo.
- xhr: irá lhe retornar o objeto XMLHttpRequest usado na execução da requisição AJAX.

Fazendo requisições GET

Você também pode usar jQuery para fazer requisições HTTP GET para um endereço. Para isso use a função “.get()”. Sintaxe:

```
$.get(URL, callback);
```

- URL: é o endereço para o qual você deseja fazer a requisição.
- callback: é a função de callback que será executada quando a requisição terminar. Essa função de callback terá dois parâmetros:
 - data: o conteúdo resultante da requisição HTTP.
 - status: “success” em caso de sucesso e “error” em caso de algum erro com a requisição.

Exemplo:

```
$("#button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

Fazendo requisições POST

Você também pode fazer requisições POST e enviar dados para o servidor através jQuery AJAX. Para isso use a função “.post()”. Sintaxe:

```
$.post(URL, data, callback);
```

- URL: endereço para o qual você deseja fazer o POST.
- data: objeto JSON com os parâmetros a serem enviados por POST.
- callback: função de callback executada quando a requisição terminar. Da mesma forma que no “get()” a função de callback irá retornar dois parâmetros “data” e “status”.

Exemplo:


```
$("#button").click(function(){
    $.post("demo_test_post.asp",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

Requisitando JSON


Caso você queira fazer uma requisição AJAX que retorna dados em JSON, você pode usar o método "getJSON()". Através dele você poderá acessar os atributos JSON do objeto retorno. Sintaxe:

```
$(selector).getJSON(url, data, success(data, status, xhr))
```

- "URL" é o endereço que você deseja carregar.
- "data" é um parâmetro opcional no qual você pode passar parâmetros para a requisição que será feita.
- "callback" é a função que será executada quando a requisição AJAX tiver terminado. Essa função de callback terá três parâmetros:
 - data: os dados retornados e codificados como JSON
 - status: "sucess" ou "error".
 - xhr: o objeto XHR usado na requisição.

Exemplo:

Supondo que você tenha um arquivo JSON a seguir:

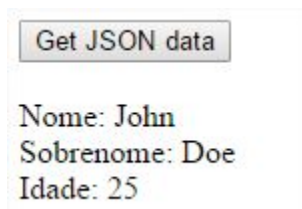


```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 25
}
```

Vamos carregar essas informações dinamicamente para a página:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
5 <script>
6 $(document).ready(function(){
7     $("button").click(function(){
8         $.getJSON("demo_ajax_json.js", function(result){
9             $("div").append('<br>Nome: ' + result.firstName);
10             $("div").append('<br>Sobrenome: ' + result.lastName);
11             $("div").append('<br>Idade: ' + result.age);
12         });
13     });
14 });
15 </script>
16 </head>
17 <body>
18
19 <button>Get JSON data</button>
20
21 <div></div>
22
23 </body>
24 </html>
```

Resultado:



Eventos Globais

jQuery fornece um conjunto de eventos relacionados as requisições AJAX para que você se registrar como observador e então executar algum comando quando um desses eventos ocorre. Os eventos fornecidos são:

- `ajaxComplete()`: Executado quando uma requisição AJAX termina.
- `ajaxError()`: Executado quando uma requisição termina com erro.
- `ajaxSend()`: Executada antes de uma requisição ser enviada.
- `ajaxStart()`: Executada quando a primeira requisição executar.
- `ajaxStop()`: Executada quando todas as requisições tiverem terminado.
- `ajaxSuccess()`: Executada sempre que uma requisição AJAX terminar.

Todas essas possuem apenas um parâmetro, que a função que será executada quando o evento ocorrer. Cada uma das funções passadas nesses eventos possuem parâmetros específicos, [consulte a documentação](#)

⁴⁶ de cada função para ver os parâmetros.

Todas essas funções devem ser chamadas através do elemento `$(document)`, como no exemplo abaixo em que sempre que houver algum erro com uma requisição vamos mostrar um alerta ao usuário.

⁴⁶ <http://api.jquery.com/category/ajax/global-ajax-event-handlers/>

```
1 $(function(){
2
3   // Todos os eventos AJAX devem ser
4   // configurados usando o elemento
5   // $(document).
6   $(document).ajaxError(function(){
7     alert('Ocorreu um erro ao executar a requisição.');
```

Encerramento

“If you are good at something, never do it for free”

Joker - The Dark Knight

Mais de jQuery

jQuery fornece outras ferramentas além da API em si. Entre essas ferramentas estão:

- **jQuery UI:** conjunto de componentes de interface como campos de data, abas, listas de sugestões, etc: <http://jqueryui.com/>
- **Theme Roller:** customize as cores dos componentes em jQuery apenas aplicando um novo CSS: <http://jqueryui.com/themeroller/>
- **jQuery Mobile:** biblioteca adaptada para o desenvolvimento de aplicações para dispositivos móveis e com tela sensível ao toque: <http://jquerymobile.com/>

Outros frameworks

jQuery é a API javascript padrão *de facto*, mas recentemente outros frameworks tem chamado a atenção, com um grande crescimento e participação de mercado. Dente eles:

- **AngularJS:** jQuery lhe fornece uma API com diversas funções, mas não lhe diz como organizar sua aplicação. Angular JS é desenvolvida pela Google, e ele não é apenas uma API, mas sim um framework, pois ele irá orientar como organizar sua aplicação. Além disso, Angular implementa um padrão *MVVM*⁴⁷. Atualmente Angular JS vem se tornando o padrão de mercado nesse novo contexto de frameworks JavaScript: <https://angularjs.org/>
- **Backbone:** <http://backbonejs.org/>
- **Ember:** <http://emberjs.com/>
- **React:** Desenvolvida pelo Facebook: <https://facebook.github.io/react/>

Documentação

Nessa apostila temos vários exemplos e link para o site W3Schools. Embora esse site seja muito fácil de navegar e tenha diversos exemplos, é importante ressaltar dois pontos:

1. O W3Schools não tem relação nenhuma com a W3C⁴⁸

⁴⁷

⁴⁸ <http://www.w3c.br/Home/WebHome>

2. Procure utilizar a documentação oficial de referência da API que você está usando, exemplo, jQuery. Em nosso curso usamos o W3Schools pela facilidade e quantidade de exemplos que podem ser visualizados on-line, mas agora que você tem um bom conhecimento, procure utilizar as documentações oficiais. Veja abaixo alguns exemplos.

jQuery

Documentação oficial da biblioteca jQuery, permite consultar cada um dos métodos disponíveis na API, os parâmetros necessários, e exemplos de utilização:

<http://api.jquery.com/>

Mozilla Developer Network.

Uma rede de desenvolvedores e vasta documentação sobre os padrões Web:

<https://developer.mozilla.org/pt-BR/>

HTML5

Documentação oficial elaborada pela W3C com a definição do padrão HTML5:

<http://www.w3.org/TR/html5/>

Demos

Veja algumas demonstrações do que HTML5, CSS3 e JS podem fazer:

<https://developer.mozilla.org/en-US/demos/>