

Desenvolvimento de Software para WEB

JavaScript

Prof. Regis Pires Magalhães

regismagalhaes@ufc.br - <http://bit.ly/ufcregis>

Elemento <script> em <head>

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>Aula de JS</title>
```

```
    <script>
```

```
      alert("Olá, Mundo!");
```

```
    </script>
```

```
  </head>
```

```
  <body>
```

```
    <h1>JavaScript</h1>
```

```
    <h2>Linguagem de programação</h2>
```

```
  </body>
```

```
</html>
```

Elemento <script> no final

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>Aula de JS</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>JavaScript</h1>
```

```
    <h2>Linguagem de Programação</h2>
```

```
    <script>
```

```
      alert("Olá, Mundo!");
```

```
    </script>
```

```
  </body>
```

```
</html>
```

Externalizando...

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>Aula de JS</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>JavaScript</h1>
```

```
    <h2>Linguagem de Programação</h2>
```

```
    <script src="js/hello.js"></script>
```

```
  </body>
```

```
</html>
```

js/hello.js

```
alert("Olá, Mundo!");
```



Console

- Chrome
 - Control + Shift + C
- Firefox
 - Control + Shift + K

```
console.log("hello, world")
```

Comentários

```
// this is a comment
```

```
/* this is a multi-line  
   or block comment */
```

Tipos

- Number
- String
- Boolean
- Object
 - Function
 - Array
 - Date
 - RegExp
- Null
- Undefined

Números

- Não há inteiro.
 - $0.1 + 0.2 = 0.300000000000000000000004$
- Operações:
 - $3 + 5.3$
 - $28 \% 6$
- Objeto Math
 - `Math.sin(3.5);`
 - `d = Math.PI * r * r;`

parseInt (e parseFloat)

- Conversão de string para número (defina a base):

> parseInt("123", 10)

123

> parseInt("010", 10)

10

> parseInt("11", 2)

3

NaN (Not a Number) e Infinity

```
> parseInt("hello", 10)
```

NaN

```
> NaN + 5
```

NaN

```
> isNaN(NaN)
```

true

```
> 1 / 0
```

Infinity

```
> -1 / 0
```

-Infinity

Strings

```
> "hello".length
```

```
5
```

- Strings são objetos:

```
> "hello".charAt(0)
```

```
h
```

```
> "hello, world".replace("hello", "goodbye")  
goodbye, world
```

```
> "hello".toUpperCase()
```

```
HELLO
```

Strings

String	<code>length</code>	Returns the number of characters in a string
	<code>concat()</code>	Joins two or more strings
	<code>indexOf()</code>	Returns the position of the first occurrence of a specified string value in a string
	<code>lastIndexOf()</code>	Returns the position of the last occurrence of a specified string value, searching backward from the specified position in a string
	<code>match()</code>	Searches for a specified string value in a string
	<code>replace()</code>	Replaces some characters with others in a string
	<code>slice()</code>	Extracts a part of a string and returns the extracted part in a new string
	<code>split()</code>	Splits a string into an array of strings
	<code>substring()</code>	Extracts the characters in a string between two specified indexes
	<code>toLowerCase()</code>	Displays a string in lowercase letters
	<code>toUpperCase()</code>	Displays a string in uppercase letters

Null e undefined

- null = deliberadamente sem valor
- undefined = ainda sem valor atribuído
 - Variáveis declaradas, mas não inicializadas
 - Membros Objeto/Array que não existem

Boolean

- true ou false
- False: 0, "", NaN, null, undefined
- Tudo mais é true
- Operações: &&, || e !
- Conversão para equivalente booleano:

> !!""

false

> !!234

true

Variáveis

- Declaradas com a palavra chave **var**:
`var a;`
`var name = "simon";`
- Declarar sem atribuir valor: undefined
- Se esquecer a palavra **var**, a variável é global. Não faça isso, por favor.

```
var mensagem = "Olá mundo";  
console.log(mensagem);
```

Conversão de tipos

```
var textoInteiro = "10";  
var inteiro = parseInt(textoInteiro);  
  
var textoFloat = "10.22";  
var float = parseFloat(textoFloat);  
  
var milNumber = 1000;  
var milString = milNumber.toFixed(2); // recebe o retorno da  
função  
console.log(milString); // imprime a string "1000.00"
```


Ponto e vírgula

- É possível omitir o ponto e vírgula no final de cada declaração, mas é boa prática utilizá-lo, inclusive por ele permitir maior flexibilidade em técnicas de compressão, como veremos mais adiante.

Operadores

- Numéricos: +, -, *, / e %
- Atribuição: =, +=, -=, *=, /=, %=
- Incremento/decremento: a++, ++a, b--, --b
- Concatenação de string:
 > "hello" + " world"
 hello world
- Coerção de tipos:
 > "3" + 4 + 5
 345
 > 3 + 4 + "5"
 75
- Adicionar uma string vazia a alguma coisa, converte tudo para string.

Comparação

- Para números e strings: <, >, <= e >=
- Igualdade: == e !=
 - Faz conversão de tipos se necessário

```
> "dog" == "dog"
true
> 1 == true
true
```
- Identidade: === e !==
 - Não faz conversão de tipos
 - Se forem de tipos diferentes, o resultado será falso

```
> 1 === true
false
> true === true
true
```

typeof

number	'number'
string	'string'
boolean	'boolean'
function	'function'
object	'object'
array	'object'
null	'object'
undefined	'undefined'

Estruturas de Controle - if

```
var name = "kittens";
```

```
if (name == "puppies") {  
    name += "!";  
} else if (name == "kittens") {  
    name += "!!";  
} else {  
    name = "!" + name;  
}
```

```
name == "kittens!!"
```

Estruturas de Controle - while e do-while

```
while (true) {  
    // an infinite loop!  
}
```

```
do {  
    var input = get_input();  
} while (inputIsValid(input))
```

Estruturas de Controle - for

```
for (var i = 0; i < 5; i++) {  
    // Will execute 5 times  
}
```

Estruturas de Controle - switch

```
switch(action) {  
    case 'draw':  
        drawit();  
        break;  
    case 'eat':  
        eatit();  
        break;  
    default:  
        donothing();  
}
```


Estruturas de Controle - switch

Expressões são permitidas

Comparações usam ===

```
switch(1 + 3):  
    case 2 + 2:  
        yay();  
        break;  
    default:  
        neverhappens();  
}
```

Curto circuito lógico

- && e || só executam o segundo operando, dependendo do resultado do primeiro.
- Útil para checagem de objetos antes de acessar seus atributos:

```
var name = o && o.getName();
```

Atribuição de valor default:

```
var name = otherName || "default";
```

Operador ternário

```
var allowed = (age > 18) ? "yes" : "no";
```

Exceções

```
try {  
    // Statements in which  
    // exceptions might be thrown  
} catch(error) {  
    // Statements that execute  
    // in the event of an exception  
} finally {  
    // Statements that execute  
    // afterward either way  
}
```

```
throw new Error("An error!");  
throw "Another error!";
```

Objetos

- Simples pares nome-valor, como:
 - Dicionários em Python
 - Hashes em Perl e Ruby
 - Hash tables em C e C++
 - HashMaps em Java
 - Arrays associativos em PHP
- Muito comuns, estrutura de dados versátil.
- Nome é uma string; valor pode ser qualquer coisa.

JSON

- JavaScript Object Notation.
- Usado para serializar objetos em formato legível ao ser humano.

```
{ "Books":  
  [  
    { "ISBN":"ISBN-0-13-713526-2",  
      "Price":85,  
      "Edition":3,  
      "Title":"A First Course in Database Systems",  
      "Authors":[ {"First_Name":"Jeffrey", "Last_Name":"Ullman"},  
                   {"First_Name":"Jennifer", "Last_Name":"Widom"} ] }  
    ,  
    { "ISBN":"ISBN-0-13-815504-6",  
      "Price":100,  
      "Remark":"Buy this book bundled with 'A First Course' - a great deal!",  
      "Title":"Database Systems:The Complete Book",  
      "Authors":[ {"First_Name":"Hector", "Last_Name":"Garcia-Molina"},  
                   {"First_Name":"Jeffrey", "Last_Name":"Ullman"},  
                   {"First_Name":"Jennifer", "Last_Name":"Widom"} ] }  
  ]  
}
```

JSON

- Valores base
 - Número, string, booleano
 - Objetos { }
 - Conjuntos de pares chave-valor
 - Arrays []
 - Listas de valores

```
{ "alunos" : [  
    { "nome": "João", "notas": [ 8, 9, 7 ] },  
    { "nome": "Maria", "notas": [ 8, 10, 7 ] },  
    { "nome": "Pedro", "notas": [ 10, 10, 9 ] }  
]  
}
```

Criação de Objetos

```
var obj = new Object();
```

ou:

```
var obj = {};
```

Equivalentes. A segunda opção chama-se sintaxe literal de objeto e é mais conveniente.

Acesso a atributos

```
obj.name = "Simon"
```

```
var name = obj.name;
```

ou:

```
obj["name"] = "Simon";
```

```
var name = obj["name"];
```

- Equivalentes.
- O segundo usa strings, podendo ser decidido em tempo de execução e usado para palavras reservadas.

Sintaxe Literal de Objetos

```
var obj = {  
  name: "Carrot",  
  "for": "Max",  
  details: {  
    color: "orange",  
    size: 12  
  }  
}
```

Para acessar:

```
> obj.details.color  
orange  
> obj["details"]["size"]  
12
```

for (var attr in obj)

- Pode-se iterar pelas chaves de um objeto:

```
var obj = { 'name': 'Simon', 'age': 25 };  
for (var attr in obj) {  
    print (attr + ' = ' + obj[attr]);  
}
```

Arrays

- Tipo especial de objeto: as chaves são números e não strings.
- Sintaxe []:

```
> var a = new Array();
```

```
> a[0] = "dog";
```

```
> a[1] = "cat";
```

```
> a[2] = "hen";
```

```
> a.length
```

```
3
```

Arrays

```
> var a = ["dog", "cat", "hen"];
```

```
> a.length
```

```
3
```

```
var palavras = ["UFC", "Ensino"];
```

```
palavras.push("Inovação");
```

```
// adiciona a string "Inovação"
```

Array.length

```
> var a = ["dog", "cat", "hen"];
```

```
> a[100] = "fox";
```

```
> a.length
```

```
101
```

```
> typeof(a[90])
```

```
undefined
```

Append seguro:

```
a[a.length] = item;
```

Iteração em Array

```
for (var i = 0; i < a.length; i++) {  
    // Do something with a[i]  
}
```

```
for (var item in a) {  
    // Do something with item  
}
```

```
["dog", "cat", "hen"].forEach(function(currentValue, index,  
array) {  
    // Do something with currentValue or array[index]  
});
```

Arrays

Array	<code>length</code>	Sets or returns the number of members in an array
	<code>concat()</code>	Joins two or more arrays and returns the result
	<code>join()</code>	Puts all the members into a string, separated by the specified delimiter
	<code>pop()</code>	Removes and returns the last element of an array
	<code>push()</code>	Adds one or more members to the end of an array and returns the new length
	<code>reverse()</code>	Reverses the order of the members in an array
	<code>shift()</code>	Removes and returns the first member of an array
	<code>slice()</code>	Returns selected members from an existing array
	<code>sort()</code>	Sorts the members of an array
	<code>splice()</code>	Removes and adds new members to an array
	<code>unshift()</code>	Adds one or more members to the beginning of an array and returns the new length

Funções (Function Declaration)

```
function add(x, y) {  
    var total = x + y;  
    return total;  
}
```

Se nada for explicitamente retornado, o valor de retorno é undefined.

Passagem de parâmetros:

```
> add()
```

```
Nan // You can't perform addition on undefined
```

```
> add(2, 3, 4)
```

```
5 // added the first two; 4 was ignored
```

Parâmetros

```
function add() {  
    var sum = 0;  
    for (var i = 0, j = arguments.length; i < j; i++) {  
        sum += arguments[i];  
    }  
    return sum;  
}
```

```
> add(2, 3, 4, 5)  
14
```

Funções anônimas (function expression)

- funções declaradas como conteúdo de variáveis.

```
var somaDoisNumeros = function(numero1, numero2) {  
    return numero1 + numero2;  
};
```

```
somaDoisNumeros(10, 20);
```

Funções anônimas (function expression)

```
var avg = function() {  
    var sum = 0;  
    for (var i = 0, j = arguments.length; i < j; i++) {  
        sum += arguments[i];  
    }  
    return sum / arguments.length;  
}
```

Funções temporais

```
// executa a minhaFuncao daqui um segundo  
setTimeout(minhaFuncao, 1000);
```

```
// executa a minhaFuncao de um em um segundo  
// util para banner rotativo  
var timer = setInterval(minhaFuncao, 1000);
```

```
// cancela execução  
clearInterval(timer);
```

Classes???

- JavaScript não possui classes.
- Funcionalidade semelhante é obtida através de protótipos de objetos.
- JavaScript usa funções como classes.
- A palavra reservada **new** cria um novo objeto e o atribui a palavra chave **this** de dentro do escopo da função invocada.
 - Pode-se então adicionar atributos a esse objeto.

Java	JavaScript
Strongly-typed	Loosely-typed
Static	Dynamic
Classical	Prototypal
Classes	Functions
Constructors	Functions
Methods	Functions

Construtores

```
function Person(first, last) {  
    this.first = first;  
    this.last = last;  
    this.fullName = function() {  
        return this.first + ' ' + this.last;  
    };  
    this.fullNameReversed = function() {  
        return this.last + ', ' + this.first;  
    };  
}  
  
var s = new Person("Lemmy", "Kilmister");
```

Construtores

```
var Pessoa = function(nome, email) {  
  console.log("criando nova pessoa");  
  console.log(typeof(this));  
  this.nome = nome;  
  this.email = email;  
}  
  
// criando nova pessoa  
var joao = new Pessoa("João da Silva", "joao@da.silva");  
console.log(joao.nome); // João da Silva  
console.log(joao.email); // joao@da.silva
```


Construtores

```
var Curso = function(nome) {  
    this.nome = nome;  
    return "curso "+ nome;  
}  
  
// Invocando como função  
var stringParaCS01 = Curso("CS01");  
typeof(stringParaCS01); // "string"  
console.log(stringParaCS01); // curso CS01  
  
// Invocando como construtor  
var objetoParaWD47 = new Curso("WD47");  
typeof(objetoParaWD47); // object  
console.log(objetoParaWD47.nome); // WD47
```

Protótipo

- Qualquer atributo ou função adicionado ao protótipo de uma dessas funções ficará disponível em qualquer objeto do tipo gerado por elas.

```
String.prototype.paraNumero = function() {  
    if(this == "um") {  
        return 1;  
    }  
}  
console.log("um".paraNumero()); // 1
```

Protótipo

```
var Pessoa = function(nome, email) {  
    this.nome = nome;  
  
    // verifica se o e-mail foi preenchido  
    if (email) {  
        this.email = email;  
    }  
}  
  
Pessoa.prototype.email = "contato@ufc.br"  
  
var ricardo = new Pessoa("Ricardo");  
console.log(ricardo.email); // contato@ufc.br  
  
var joao = new Pessoa("Joao da Silva",  
    "joao@da.silva");  
console.log(joao.email); // joao@da.silva
```

Protótipo

```
var Pessoa = function(nome, email) {  
    this.nome = nome;  
  
    // verifica se o e-mail foi preenchido  
    if (email) {  
        this.email = email;  
    }  
};  
  
Pessoa.prototype.fala = function(){  
    console.log("Olá, meu nome é "+this.nome+" e meu email é  
"+this.email);  
};  
  
Pessoa.prototype.anda = function(){  
    console.log("Estou andando");  
};
```

Herança

- Modos de implementar herança em JavaScript:
 - Prototype-chaining Inheritance
 - Parasitic Combination Inheritance
 - Functional Inheritance
- Ver:
 - <http://blog.caelum.com.br/reaproveitando-codigo-com-javascript-heranca-e-prototipos/>

Closures

- Função interna.
- Objeto de escopo.

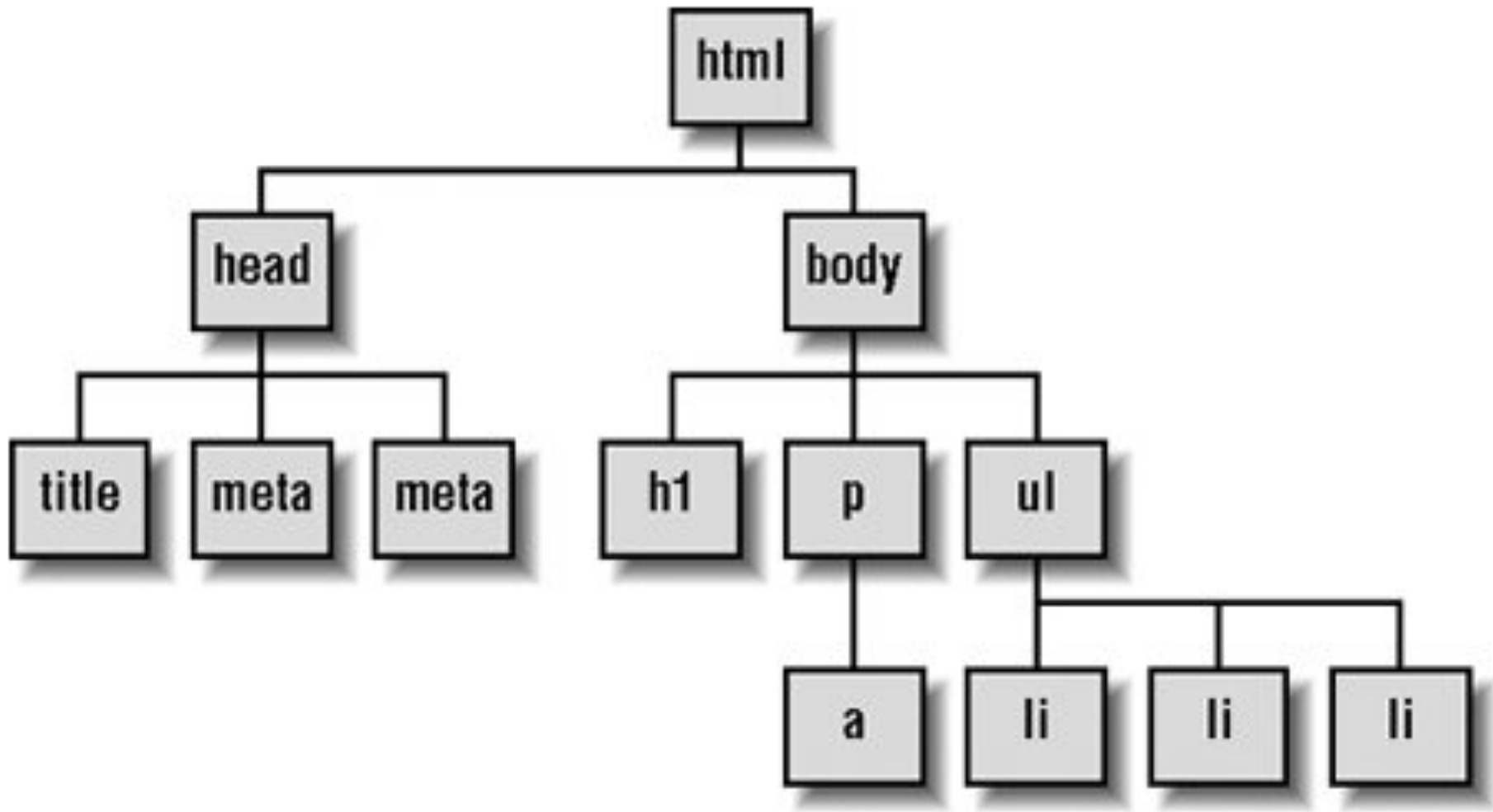
```
function makeAdder(a) {  
  return function(b) {  
    return a + b;  
  };  
}
```

```
var x = makeAdder(5);  
var y = makeAdder(20);  
x(6); // 11  
y(7); // 27
```

DOM - Document Object Model

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample XHTML</title>
  <meta charset="utf-8">
  <meta http-equiv="Content-Language" content="en-us" />
  <title>Aula de JS</title>
</head>
<body>
  <h1>This is a heading, level 1</h1>
  <p>This is a paragraph of text with a
    <a href="/path/to/another/page.html">link</a>.</p>
  <ul>
    <li>This is a list item</li>
    <li>This is another</li>
    <li>And another</li>
  </ul>
</body>
</html>
```

Árvore de documento



Interatividade na Web - querySelector

```
var titulo = document.querySelector("#titulo");  
titulo.textContent = "Agora o texto do elemento mudou!";
```

```
var painelNovidades = document.querySelector(  
    "section#main.painel#novidades");  
painelNovidades.style.color = "red"
```

Interatividade na Web - querySelectorAll

```
var paragrafos = document.querySelectorAll("div p");  
paragrafos[0].textContent = "Primeiro parágrafo da seleção";  
paragrafos[1].textContent = "Segundo parágrafo da seleção";
```

Funções + Eventos

```
function mostraAlerta() {  
    alert("Funciona!");  
}
```

```
// obtendo um elemento através de um seletor de ID  
var titulo = document.querySelector("#titulo");  
  
titulo.onclick = mostraAlerta;
```

Funções + Eventos

```
document.querySelector("#titulo").onclick = function() {  
    alert("Funciona!");  
}
```

Limitação: cada seleção só pode receber um evento.
E se quisermos vincular mais eventos à mesma seleção?

Funções + Eventos

```
document.querySelector("#titulo").addEventListener('click',  
function() {  
    alert("Funciona!");  
})
```

Eventos principais

- onclick: clica com o mouse
- ondblclick: clica duas vezes com o mouse
- onmousemove: mexe o mouse
- onmousedown: aperta o botão do mouse
- onmouseup: solta o botão do mouse (útil com os dois acima para gerenciar drag'n'drop)
- onkeypress: ao pressionar e soltar uma tecla
- onkeydown: ao pressionar uma tecla.
- onkeyup: ao soltar uma tecla. Mesmo acima.
- onblur: quando um elemento perde foco
- onfocus: quando um elemento ganha foco
- onchange: quando um input, select ou textarea tem seu valor alterado
- onload: quando a página é carregada
- onunload: quando a página é fechada
- onsubmit: disparado antes de submeter o formulário. Útil para realizar validações

Propriedades e funções

Property/method	Description
<code>event</code>	Represents the state of an event
<code>history</code>	Contains the URLs the user has visited within a browser window
<code>location</code>	Gives read/write access to the URI in the address bar
<code>opener</code>	Sets or returns a reference to the window that created the window
<code>parent</code>	Returns the parent window
<code>screenLeft</code>	Returns the x-coordinate of the upper-left corner of the browser relative to the upper-left corner of the screen
<code>screenTop</code>	Returns the y-coordinate of the top corner of the browser relative to the top corner of the screen
<code>status</code>	Sets or returns the text in the status bar of the window
<code>alert()</code>	Displays an alert box with a specified message and an OK button
<code>close()</code>	Closes the current window
<code>confirm()</code>	Displays a dialog box with a specified message and an OK and a Cancel button
<code>focus()</code>	Sets focus on the current window
<code>open()</code>	Opens a new browser window
<code>print()</code>	Prints the contents of the current window
<code>setTimeout()</code>	Calls a function or evaluates an expression after a specified number of milliseconds

Exercício

Em um formulário, receber preço e quantidade. Exibir o valor total.
Quando um deles for alterado, alterar automaticamente o valor total.

Referências e Links importantes

- Caelum WD-43 - **Desenvolvimento Web com HTML, CSS e JavaScript.**

Disponível em:

<https://www.caelum.com.br/apostila-html-css-javascript/>

- www.w3schools.com
- FREEMAN Eric & FREEMAN Elisabeth. **Use a Cabeça! HTML com CSS e XHTML.** Alta Books, 1ª Edição, 2006.
- Apostila da K19 - **Desenvolvimento Web com HTML, CSS e Javascript.**

Referências e Links importantes

- Uma reintrodução ao JavaScript (Tutorial de JS)
 - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/A_re-introduction_to_JavaScript
- Karloespiritu javascript cheatsheet
 - <http://karloespiritu.github.io/cheatsheets/javascript/>
- A (Re)-Introduction to JavaScript
 - Simon Willison - <http://simonwillison.net/>
 - <http://simon.incutio.com/slides/2006/etech/javascript/js-tutorial.001.html>
 - <http://simon.incutio.com/slides/2006/etech/javascript/js-reintroduction-notes.html>
- ROBBINS, Jennifer. *Web Design in a Nutshell*. O'Reilly. 3ª Ed .2006.

Obrigado!
Dúvidas, comentários, sugestões?

Regis Pires Magalhães
regismagalhaes@ufc.br

