

DESENVOLVIMENTO WEB 2

Jair C Leite



Aula 2

1. Tecnologias fundamentais para o lado cliente – 5/9 e 12/9
 1. HTML 5
 2. CSS
 3. DOM
 4. JS



HTML 5

Objetivos de HTML

- definir meta informações sobre a página (head)
 - como o browser ou máquinas de busca podem saber informações importantes sobre a página
- definir a estrutura do texto de uma página (body)
 - os elementos que formam o texto:
 - cabeçalhos, parágrafos, listas, tabelas, links, imagens, etc.
- Os aspectos estéticos (estilo) e o layout são especificados usando a linguagem CSS e são independentes do arquivo HTML

HTML - Histórico

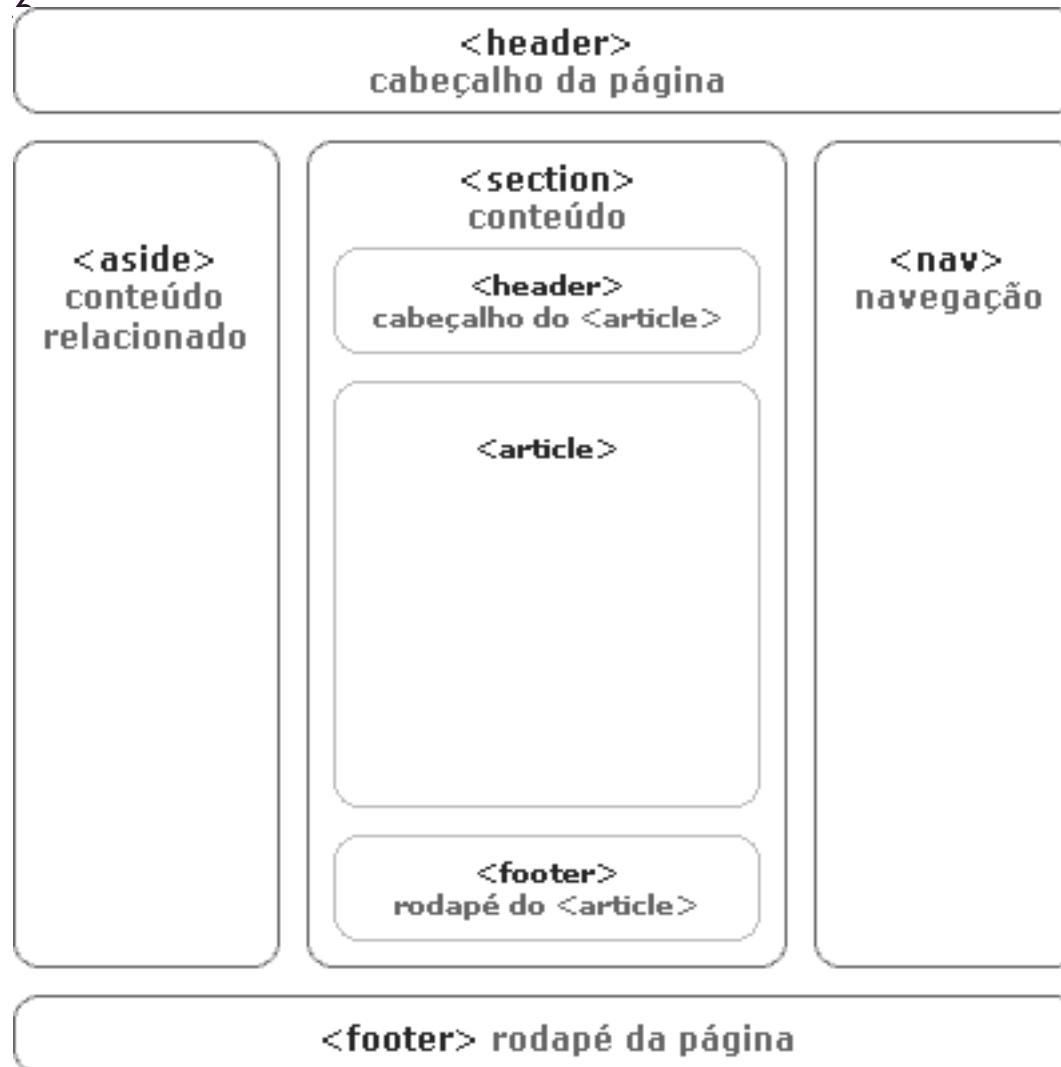
- HTML – 1991
 - não existia browsers gráficos
- HTML 2 – 1995
 - web comercial
- HTML 3.2 – 1997
 - guerra dos browsers IE e Navigator (Netscape)
- HTML 4.01 – 1999
 - estabilidade
- XHTML – 2000
 - HTML no padrão XML
- HTML 5 – 2012
 - gráficos, animações, multimídia e programação

Estrutura de um documento HTML5

- A estrutura é a mesma. Muda o DOCTYPE

```
1 <!DOCTYPE HTML>
2 <html lang="pt-br">
3 <head>
4 <meta charset="UTF-8">
5 <link rel="stylesheet" type="text/css" href="estilo.css">
6 <title></title>
7 </head>
8 <body>
9
10 </body>
11 </html>
```


Elementos semânticos – layout e navegação



Video

[Edit and Click Me »](#)

Your Result:



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

<video width="320" height="240"
controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  Your browser does not support the video tag.
</video>

</body>
</html>
```


Video interativo

Edit and Click Me »

```

<!DOCTYPE html>
<html>
<body>

<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br />
  <video id="video1" width="420">
    <source src="mov_bbb.mp4" type="video/mp4" />
    <source src="mov_bbb.ogv" type="video/ogg" />
    Your browser does not support HTML5 video.
  </video>
</div>

<script type="text/javascript">
var myVideo=document.getElementById("video1");


function playPause()
{
  if (myVideo.paused)
    myVideo.play();
  else
    myVideo.pause();
}

function makeBig()

```

Your Result:

Play/Pause Big Small Normal



Video courtesy of [Big Buck Bunny](#).

Novidades - audio


[Edit and Click Me »](#)

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

<audio controls="controls">
  <source src="song.ogg" type="audio/ogg" />
  <source src="song.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
</audio>

</body>
</html>
```

Your Result:



Interactive

- Define elementos para interação com o usuário
 - Tem intersecção com phrasing e embed
- a
 - audio
 - button
 - details
 - embed
 - iframe
 - img
 - keygen
 - label
 - menu
 - object
 - select
 - textarea
 - video

Novos controles

- Objetivos
 - Diminuir o esforço de programação de scripts de validação
 - Facilitar a integração com outros serviços
- Problemas
 - Ainda tem muito pouca coisa implementada nos principais browsers (2012)
- Exemplos
 - Output
 - Usado para resultados javascript
- Datas e horas
 - O campo de formulário pode conter qualquer um desses valores no atributo type:
 - datetime
 - date
 - month
 - week
 - time
 - datetime-local
- Números
 - Intervalos com valores específicos
 - Intervalos com valores indefinidos

Exemplo


[Edit and Click Me »](#)

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

<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
<input type="range" name="a" value="50" />100
+<input type="number" name="b" value="50" />
=<output name="x" for="a b"></output>
</form>

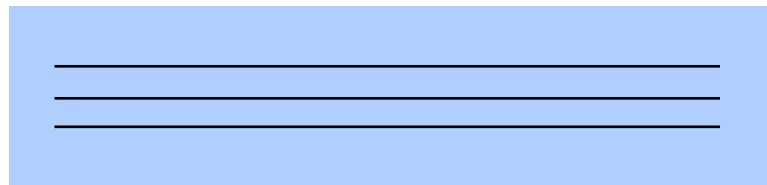
</body>
</html>
```

Your Result:

0  100 + =145

Blocos e elementos inline

- HTML estrutura a página em blocos (quadrados) e linhas (fluxo)
 - blocos tem são áreas com limites definidos
 - fluxos são caracteres de texto e outros elementos “em linha” (inline)



Blocos e elementos inline

- Alguns elementos são blocos
 - `<h1>`, `<p>`, ``, `<table>`, `<div>`
- Outros são colocados dentro de um fluxo de linhas
 - ``, `<a>`, ``, `<td>`
- **Div** é um elemento que define um bloco a ser referenciado posteriormente (por exemplo CSS)
 - `<div><p>Estes dois parágrafos</p>`
 - `<p>formam um bloco</p></div>`
- **Span** é um elemento que define um elemento em linha

O elemento SPAN

- SPAN permite definir trechos de um fluxo de forma que possam ser referenciados e formatados
 - `<p>Este parágrafo tem um elemento em linha chamado meuspan que posso referenciar quando quiser.</p>`

O Elemento DIV

- Usado para definir divisões (ou seções) no documento HTML
 - `<div id="nome" style="CSS">`

```
<html>
<body>

<h3>This is a header</h3>
<p>This is a paragraph.</p>

<div style="color:#00FF00">
  <h3>This is a header</h3>
  <p>This is a paragraph.</p>
</div>

</body>
</html>
```

This is a header

This is a paragraph.

This is a header

This is a paragraph.

Layout em HTML – elemento DIV

```
<html>
<body>

<div id="container" style="width:500px">

<div id="header" style="background-color:#FFA500;">
<h1 style="margin-bottom:0;">Main Title of Web Page</h1> </div>

<div id="menu" style="background-color:#FFD700;height:200px;width:100px;float:left;">
<b>Menu</b> <br />
HTML<br />
CSS<br />
JavaScript</div>

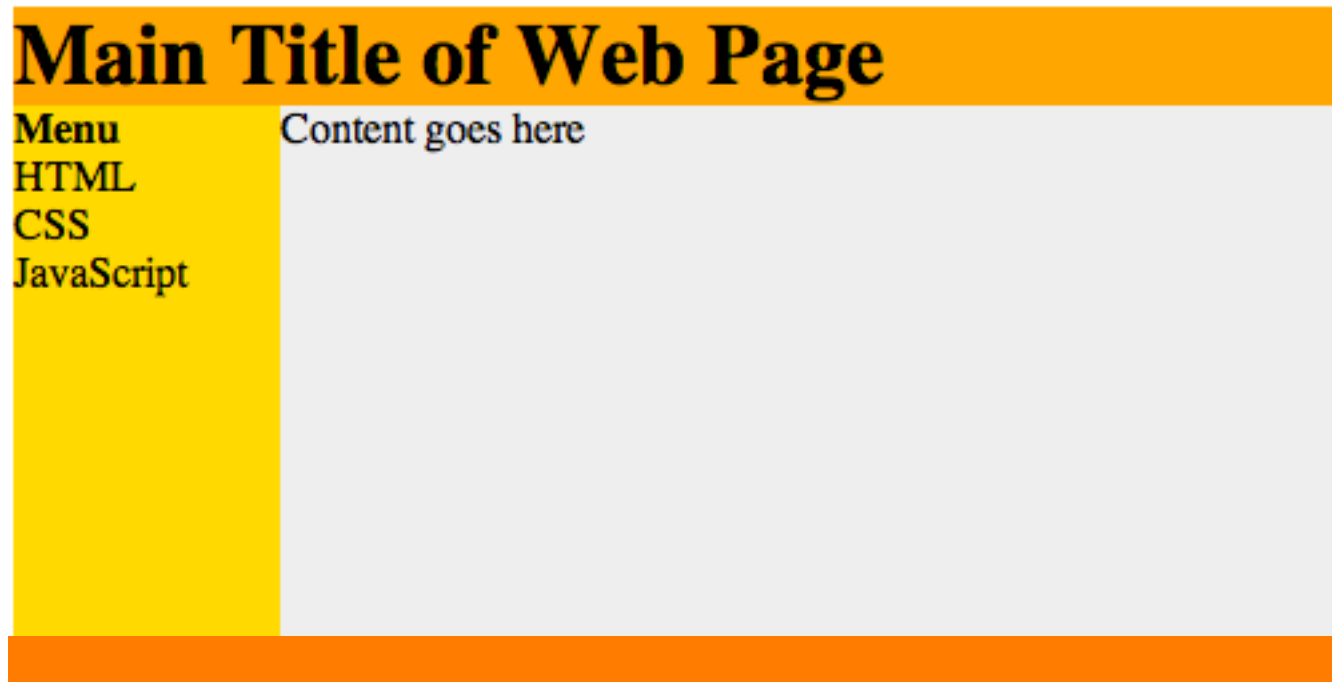
<div id="content" style="background-color:#EEEEEE;height:200px;width:400px;float:left;">
Content goes here</div>

<div id="footer" style="background-color:#FFA500;clear:both;text-align:center;">
Copyright © 2011 W3Schools.com</div>

</div>

</body>
</html>
```

Resultado do exemplo anterior



JAVASCRIPT

Utilizando JS em HTML

```
<html>
<head>
<title> ... </title>
<meta http-equiv="content-script-type" content="text/javascript">
<script type="text/javascript" src="http://www.site.com/arq.js">
</script>
</head>
<body>
  <p> Elementos e textos em HTML </p> ....
  <script type="text/javascript">
... aqui vem o código javascript ...
</script>
</body>
</html>
```

Execução de *scripts*

- Existem dois instantes no qual um *script* pode ser associado a um documento HTML e executado pelo *browser*
- Aqueles que são executados quando o documento é **carregado**
 - Estes são declarados usando o elemento **script**
- Aqueles que são executados quando ocorre um **evento intrínseco** associado a um elemento HTML.
- **Funções** podem ser associadas a eventos intrínsecos

Variáveis

- Variáveis em JS

- `var x;`
- `var nome="Maria";`
- `var primNome = "Maria", sobreNome = "Silva", idade = 20;`
- `var x = 2 + 3 + "5";`
- `var x = "5" + 2 + 3;`

- Nomes

- Devem começar com letra ou `_` (underscore)
- São sensíveis a maiúscula e minúsculas
- Podem ser locais (a funções) ou globais (página)

Tipos de dados

- Tipos primitivos: `number`, `string`, `boolean`, `undefined`, `null`
- Objetos, Arrays e Funções
- Operador `typeof`
- `typeof "Maria"` `(string)`
- `typeof 5` `(number)`
- `typeof 3.14` `(number)`
- `typeof true` `(boolean)`
- `typeof x` `(undefined)`
- `typeof [1, 3, 5, 7]` `(object)`
- `typeof function calc() { }` `(function)`

Operadores

- Operadores
 - Javascript permite vários operadores comuns a C/C++ e Java
 - Aritmética e atribuição
 - `+, -, *, /, %, ++, --, =, +=, -=, *=, /=, %=`
 - Comparação e lógica
 - `==, !=, !==, <, >, <=, >=, ===, &&, ||, !, ?`
- `var txt = "<h2>";`
- `txt += "Maria";`
- `txt += " ";`
- `txt += "Silva";`
- `txt += "</h2>";`

Estruturas de controle – If e Switch

- If ... else

```
if (condition) {  
    Executa código se verdadeiro  
}  
else {  
    Executa código se não for verdadeiro  
}
```

- Switch

```
switch(n) {  
    case 1: "código" break;  
    case 2: "código" break;  
    default: "código"  
}
```

Estruturas de controle – for e while

- for

```
for (i=0; i<=10; i++) {  
    "código"  
}
```

- While

```
while (variable<=endvalue) {  
    "código"  
}
```

- Do-while

```
do {  
    "código"  
} while (variable<=endvalue);
```

Arrays

- `var carros = ["Fiat", "Ford", "VW"];`
- `document.write(carros);`
- `document.write(carros[1]);`

- `var pessoas, text, fLen, l;`
- `pessoas= ["Maria", "Marisa", "Marina", "Marilena"];`
- `text = "";`
- `fLen = pessoas.length;`
- `for (i = 0; i < fLen; i++) {`
- `text += "" + pessoas[i] + "";`
- `}`

Objetos em Javascript

- `var pessoa = {
 primNome : "Maria", sobreNome : "Silva", idade : 20
};`
 - `pessoa.priNome`
 - `pessoa["primNome"]`
- Métodos
`var pessoa = {
 primNome : "Maria",
 sobreNome : "Silva",
 idade : 55,
 completo : function() {
 return this.primNome + " " + this.sobreNome + " tem " +
 this.idade + " anos.";
 }
};`

Construtores de objetos

- Criando construtores

```
function Pessoa(primNome, sobreNome, idade) {  
  this.primNome = primNome;  
  this.sobreNome = sobreNome;  
  this.idade = idade;  
}
```

- Usando

- `var pessoa = new Pessoa("Maria", "Silva", 20);`

- Atributos e métodos podem ser adicionados

- `pessoa.profissao = "professor";`
- `pessoa.nomeCompleto = function() {
 return nome = primNome + sobreNome;`

Objetos

- **Objects** são objetos
- **Arrays** são objetos (mas não devem ser criado com **new**)
- **Functions** são objetos
- **Dates** são objetos
- **Maths** são objetos
- **Regular expressions** são objetos
- Tipos primitivos
 - **Booleans** podem ser objetos (quando definidos com **new**)
 - **Numbers** podem ser objetos (quando definidos com **new**)
 - **Strings** podem ser objetos (quando definidos com **new**)
 - Devem ser evitados!

Funções

- Funções agrupam funcionalidades do código, organizando-o.
- Permitem o controle de quando o código deve ser executado.
- Pode ser definida em qualquer local do HTML
- Recomenda-se coloca-la em arquivos externos .js ou no final do body
- Sintaxe

```
function functionname(var1,var2,...,varX) {  
    código  
}
```


Exemplos – funções

```
<html>
<head>
<script type="text/javascript">
function product(a,b) {
    return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

Eventos intrínsecos

- É possível associar uma ação a um certo número de eventos que ocorrem quando o usuário interage com o *browser*
- Estes eventos são chamados de **eventos intrínsecos**
- Um evento intrínseco pode ser associado a um elemento HTML
- Quando o evento ocorre uma ação associada pode ser executada
- Um função JS pode ser associada ao evento

Associando *scripts* a eventos intrínsecos de controles

- Os seguintes controles respondem a eventos intrínsecos:
 - INPUT, SELECT, BUTTON, TEXTAREA, LABEL
- Pode-se associar um *script* à ocorrência de um determinado evento em um controle

`<elemento... evento="script">`

`<input type="text" name="nome"
onblur="validenome(this.value)">`

Exemplo – Javascript com evento

```
<html>
<head>
  <script type="text/javascript">
    function displayDate() {
      document.getElementById("demo").innerHTML=Date();
    }
  </script>
</head>
<body>
  <h1>My First Web Page</h1>
  <p id="demo">This is a paragraph.</p>

  <button type="button" onclick="displayDate()">Display Date</button>

</body>
</html>
```

Exemplos de eventos intrínsecos

- onload, onunload
- onclick, ondblclick
- onmousedown, onmouseup, onmouseover, onmousemove, onmouseout
- onfocus, onblur
- onkeypress, onkeydown, onkeyup
- onsubmit, onreset
- onselect, onchange

Exemplo – função, evento e alerta

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

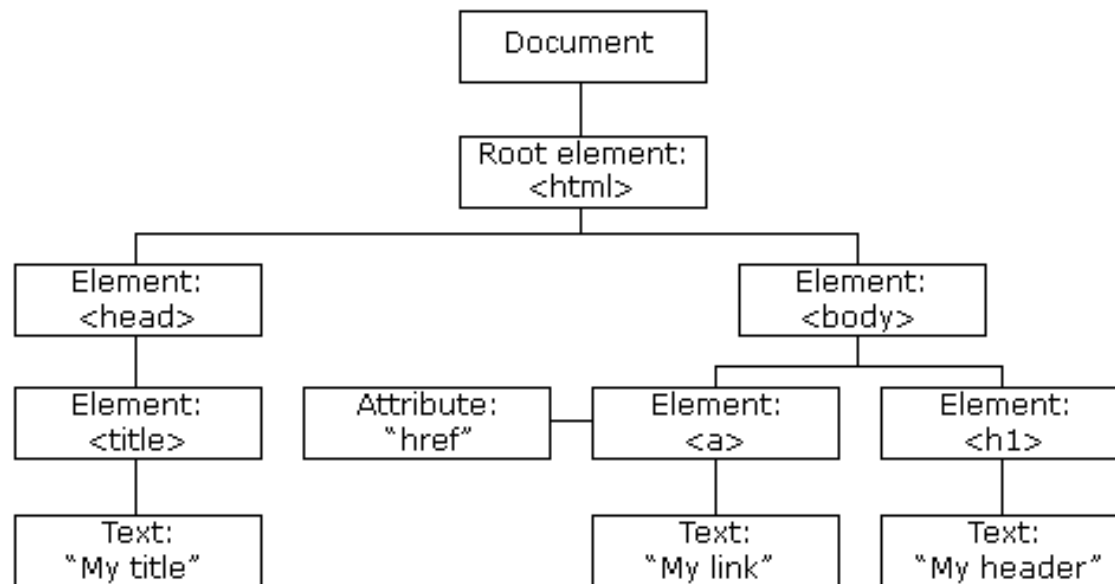
</body>
</html>
```

Modelo de objetos de um documento (DOM)

- Os elementos de um documento HTML podem ser modificados dinamicamente pelos programas scripts
- Um documento HTML segue um modelo de objetos específico (DOM)
- O modelo define uma hierarquia de objetos
- Cada objeto possui métodos e propriedades associadas.

DOM – Document Object Model

- DOM é um padrão do W3C
- Define um padrão para acessar elementos de documentos HTML e XML
- Define uma API com objetos, propriedades e métodos
- Em DOM, qualquer elemento HTML/XML é um nó numa árvore



Modificando documentos dinamicamente com *scripts*

- Os objetos são referenciados no script da seguinte forma

`Document.myform.text1.value`

`Document.form2.button2.value`

- Métodos ou propriedades associados a um objeto podem ser referenciado da mesma forma

`Document.write("texto a ser escrito")`

`Document.title="Um documento simples"`

Exemplo

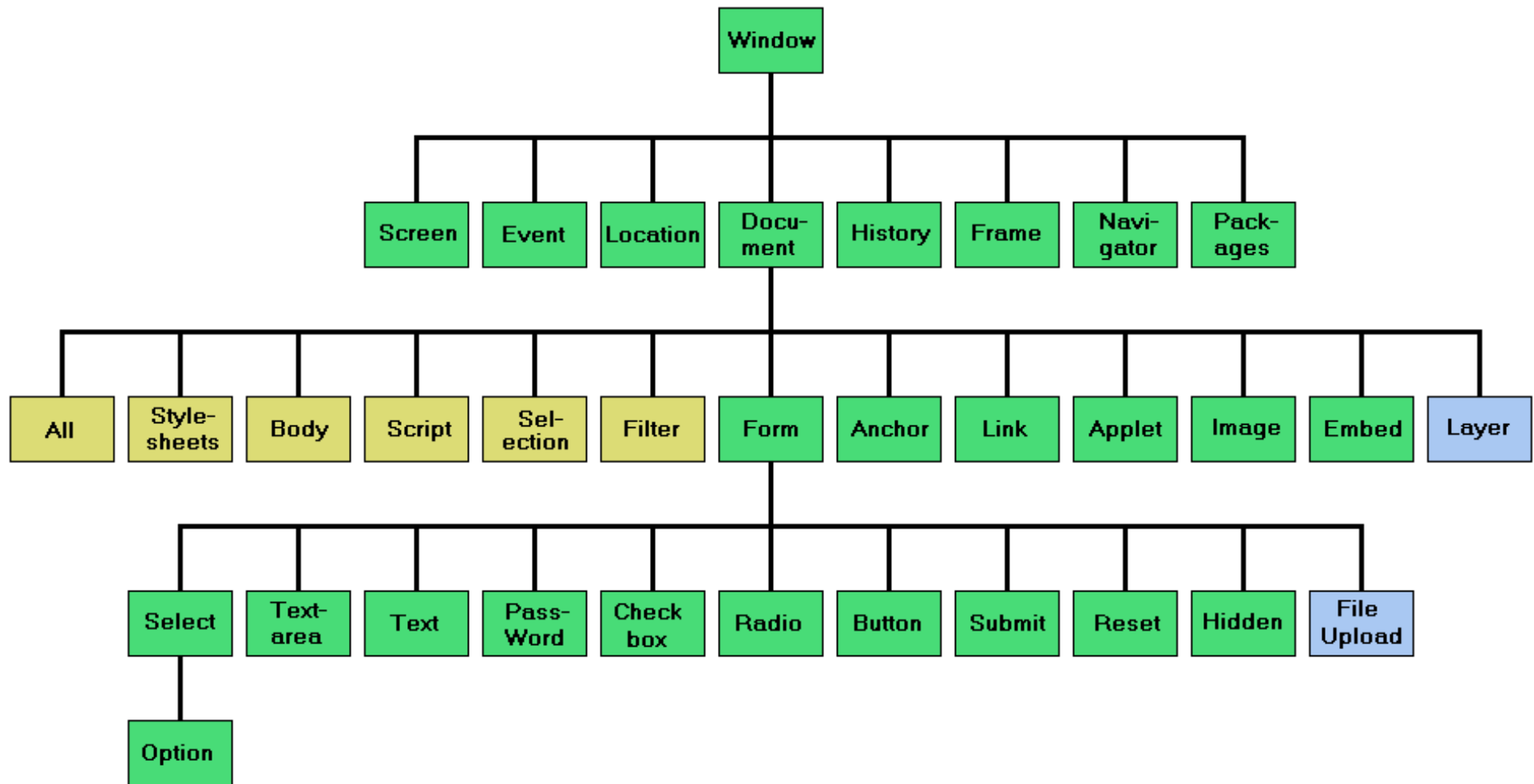
- Mudando elementos HTML

```
<html>
<body>
<p id="p1">Hello World!</p>
```

```
<script type="text/javascript">
document.getElementById("p1").innerHTML="New text!";
</script>
</body>
</html>
```

```
<html>
<body>
<script type="text/javascript">
document.body.backgroundColor="lavender";
</script>
</body>
</html>
```

DOM – Visão geral



Os objetos de um *browser*

- Toda página possui os seguintes objetos
 - Window – objeto de mais alto nível
 - Document – contém propriedades e objetos do documento corrente
 - Location – propriedades baseadas na URI atual
 - History – contem as URI previamente acessadas

Objeto navigator

- Permite obter informações sobre o Browser
- Exemplo

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

```
<script type="text/javascript">
```

```
txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
```

```
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
```

```
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
```

```
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
```

```
txt+= "<p>Platform: " + navigator.platform + "</p>";
```

```
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
```

```
document.getElementById("example").innerHTML=txt;
```

```
</script>
```

HTML DOM – propriedades e métodos

- Seja x um elemento HTML, temos:
- Propriedades
 - x.innerHTML - the text value of x
 - x.nodeName - the name of x
 - x.nodeValue - the value of x
 - x.parentNode - the parent node of x
 - x.childNodes - the child nodes of x
 - x.attributes - the attributes nodes of x
- Métodos
 - x.getElementById(id) - get the element with a specified id
 - x.getElementsByTagName(name) - get all elements with a specified tag name
 - x.appendChild(node) - insert a child node to x
 - x.removeChild(node) - remove a child node from x

Exemplo

- Mudando estilo (CSS) com função

```
<html>
<body>
<script type="text/javascript">
function ChangeBackground() {
    document.body.style.backgroundColor="lavender";
}
</script>
</head>
<body>
<input type="button" onclick="ChangeBackground()"
    value="Change background color" />
</body>
</html>
```

http://www.w3schools.com/html/dom/dom_using.asp

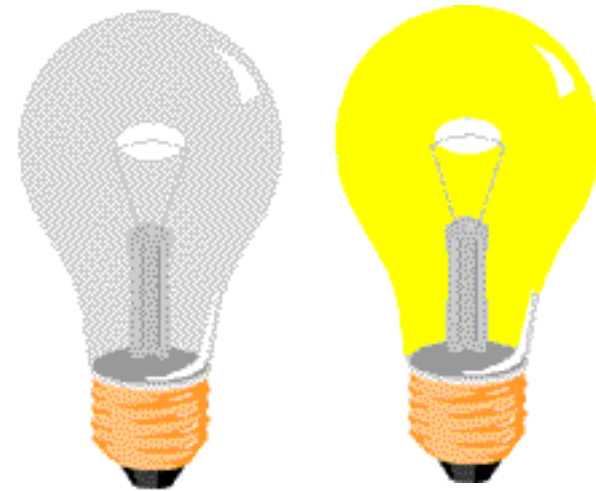
Exemplo – acendendo uma lâmpada

```
<html>
<head>
<script type="text/javascript">
cc=0;
function changeimage()
{
if (cc==0) {
  cc=1;
  document.getElementById('myimage').src="bulbon.gif";
}
else {
  cc=0;
  document.getElementById('myimage').src="bulboff.gif";
}
}
</script>
</head>
```

```
<body>


<p>Click to turn on/off the light</p>

</body>
</html>
```



AJAX E JSON

Asynchronous JavaScript and XML
JavaScript Object Notation

JSON

- JSON é string que representa objetos JS
- Objeto JS
 - `var pessoa = { primNome : "Maria", sobreNome : "Silva", idade : 20 };`
- String JSON
 - `var pessoaJson = { "primNome" : "Maria", "sobreNome" : "Silva", "idade" : 20 };`

Transformando JSON

- Transformando objeto em JSON
 - `var pessoa = { primNome : "Maria", sobreNome : "Silva", idade : 20 };`
 - `var myJSON = JSON.stringify(myObj);`
 - `document.write(myJSON);`
- Transformando JSON em objeto
 - `var pessoaJson = { "primNome" : "Maria", "sobreNome" : "Silva", "idade" : 20 };`
 - `var myObj = JSON.parse(myJSON);`
 - `document.getElementById("demo").innerHTML = myObj.name;`

Tipos de dados em JSON

- String
 - { "primNome" : "Maria" }
- Number
 - { "idade" : 20 }
- Object
 - { "pessoa" : { "primNome" : "Maria", "sobreNome" : "Silva", "idade" : 20 } }
- Boolean
 - { "falecimento" : false }
- Arrays
 - { "filhos" : ["Maria" , "Marisa" , "Marina"] }

Coleções

```
{  
  { "primNome" : "Maria", "sobreNome" : "Silva",  
    "idade" : 20 },  
  { "primNome" : "Marina", "sobreNome" : "Silva",  
    "idade" : 21 },  
  { "primNome" : "Mariana", "sobreNome" : "Silva",  
    "idade" : 22 },  
  { "primNome" : "Marisa", "sobreNome" : "Silva",  
    "idade" : 23 },  
  { "primNome" : "Marilena", "sobreNome" : "Silva",  
    "idade" : 24 },  
}
```

AJAX

- AJAX significa Asynchronous JavaScript And XML.
- AJAX não é uma linguagem de programação.
- AJAX não é um framework.
- AJAX é uma técnica para acessar servidores web a partir de uma página no cliente.

Exemplo – 1/3

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>Usando AJAX </h2>
<button type="button" onclick="ajaxRequest()">Clique para mudar</button>
</div>
<script>
function ajaxRequest() {
...
}
</script>
</body>
</html>
```

Exemplo – 2/3

```
<div id="demo">  
<h2>Usando AJAX </h2>  
<button type="button" onclick="ajaxRequest()">Clique para mudar</button>  
</div>
```

```
<script>  
...  
function ajaxRequest() {  
    var xhr = new XMLHttpRequest();  
    xhr.onreadystatechange = carregarTexto;  
    xhr.open("GET", "ajax.txt", true);  
    xhr.send();  
}  
</script>
```


Exemplo – 3/3

```
<script>
var carregarTexto = function() {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML =
            this.responseText;
    }
};
function ajaxRequest() {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = carregarTexto;
    xhr.open("GET", "ajax.txt", true);
    xhr.send();
}
</script>
```