Introdução às Tecnologias Web

A T2Ti

Apresentação

A T2Ti nasce do sonho de três colegas que trabalhavam no maior banco da américa latina.

Tudo começa em 2007 com o lançamento do curso Java Starter. Logo depois veio o Siscom Java Desktop seguido de outros treinamentos.

Desde então a Equipe T2Ti se esforça para produzir material de qualidade que possa formar profissionais para o mercado, ensinando como desenvolver sistemas de pequeno, médio e grande porte.

Um dos maiores sucessos da Equipe T2Ti foi o Projeto T2Ti ERP que reuniu milhares de profissionais num treinamento dinâmico onde o participante aprendia na prática como desenvolver um ERP desde o levantamento de requisitos. Foi através desse treinamento que centenas de desenvolvedores iniciaram seu negócio próprio e/ou entraram no mercado de trabalho.

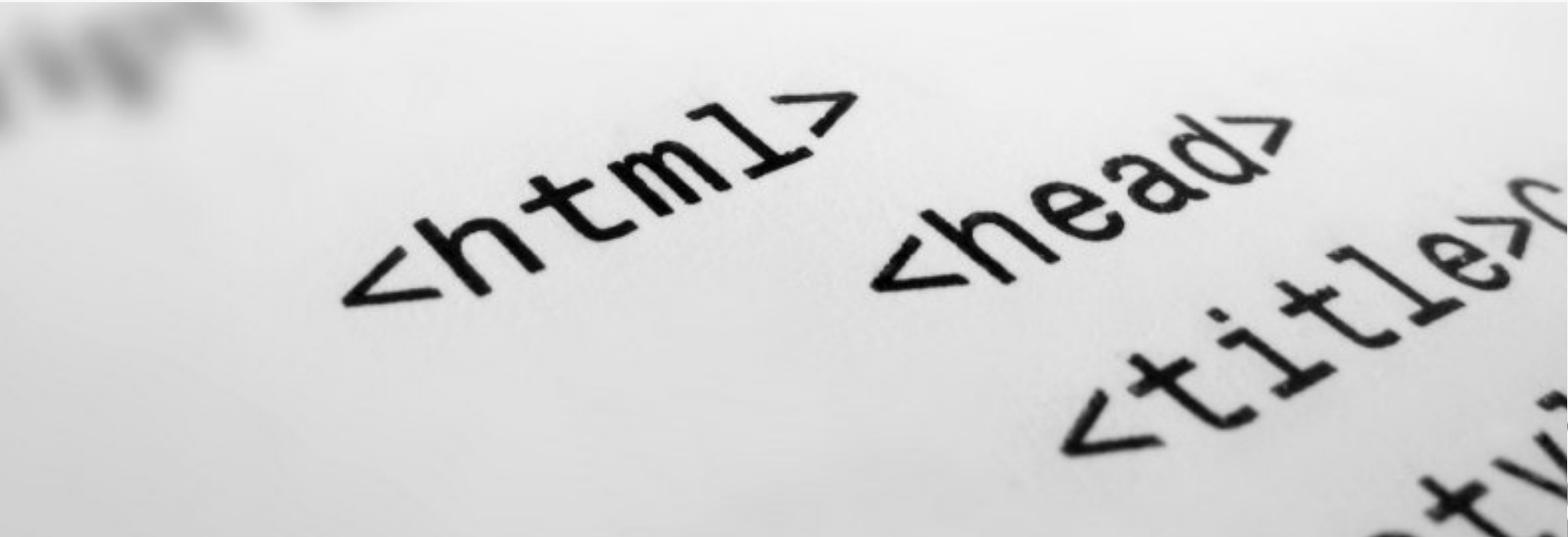
Em 2010 a T2Ti lança sua primeira aplicação para produção, o Controle Financeiro Pessoal. O sucesso foi tanto que saiu até em matéria no site Exame, ficando entre os 10 aplicativos mais baixados da semana.

Começa então a era de desenvolvimento de sistemas para alguns clientes exclusivos, pois o foco ainda era em desenvolvimento de treinamentos. A T2Ti desenvolve sistemas para o mercado nacional e internacional.

Atualmente a T2Ti se concentra nas duas vertentes: desenvolver sistemas e produzir treinamentos.

_____

HTML – HyperText Markup Language



Introdução

Muitos desenvolvedores Web dominam linguagens do lado servidor, tais como PHP, ASP, JSP, etc.

No entanto, uma boa parte deles esquece do design, da interface. Isso em muitos casos é positivo, pois o desenvolvedor se preocupa apenas com o seu trabalho e passa a bola para o designer.

Mas nem sempre é isso que ocorre. Em muitas empresas e para aqueles que são *freelancers*, saber desenvolver um site por completo é imprescindível. Para tanto a base de tudo é o HTML.

HTML (HyperText Markup Language - Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada para produzir páginas na Web. Documentos HTML podem ser interpretados por navegadores. A tecnologia é fruto do "casamento" dos padrões HyTime e SGML.

O HTML original foi criado por Tim Berners-Lee. Na época a linguagem não era uma especificação, mas uma coleção de ferramentas para resolver um problema de Tim: a comunicação e disseminação das pesquisas entre ele e seu grupo de colegas. Sua solução, combinada com a então emergente internet pública (que tornar-se-ia a Internet) ganhou atenção mundial.

Introdução

Desde a sua criação o HTML evoluiu muito e várias versões foram lançadas. Quem controla as especificações da linguagem é o W3C (World Wide Web Consortium).

Em 2000 a linguagem tornou-se uma norma internacional. A última especificação HTML lançada pelo W3C foi a recomendação HTML 4.01, publicada no final de 1999. Uma errata ainda foi lançada em 2001.

Desde a publicação do HTML 3.5 no final de 1997, o grupo de trabalho do W3C tem cada vez mais - e de 2002 a 2006, de forma exclusiva - focado no desenvolvimento do XHTML, uma especificação HTML baseada em XML que é considerada pelo W3C como um sucessor do HTML. O XHTML faz uso de uma sintaxe mais rigorosa e menos ambígua para tornar o HTML mais simples de ser processado e estendido.

Em janeiro de 2008 a W3C publicou a especificação do HTML5, a próxima versão do HTML, como *Working Draft*. Apesar de sua sintaxe ser semelhante a de SGML, o HTML5 abandonou qualquer tentativa de ser uma aplicação SGML e, definiu explicitamente sua própria serialização "html", além de uma alternativa baseada em XML, o XHTML5.

O HTML5 não será objeto de estudo nesse momento. Vamos nos concentrar pura e simplesmente no HTML.

Conceitos | Navegadores

O que significa HTML?

HyperText Markup Language.

Hyper – oposto de linear. Significa que você pode navegar através de uma página e até mesmo para outras páginas sem precisar esperar por ações ou linhas de comando.

Text – Texto.

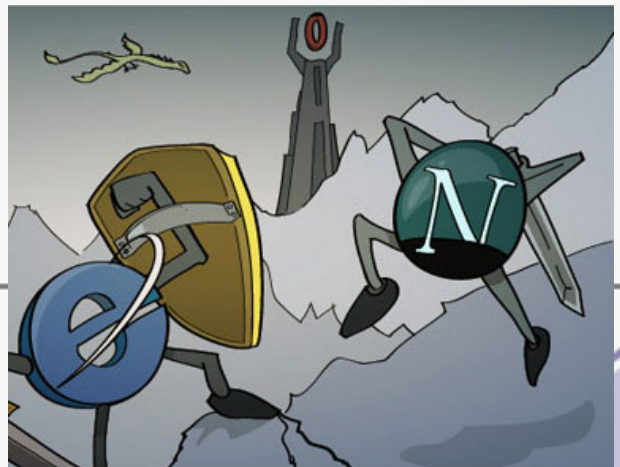
Markup – Marcações. Você realiza uma série de marcações num documento HTML. Essas marcações é que indicam o cabeçalho, título, corpo do documento, etc.

Language – Linguagem. Sim, o HTML é uma linguagem. No entanto é uma linguagem muito fácil de aprender.

Uma página HTML será apresentada num browser ou navegador. Os navegadores mais conhecidos atualmente são o Internet Explorer, o Chrome, o Safari e o Firefox.

Na década de 90 o navegador mais popular era o Netscape. A Microsof lançou o IE e com isso começou a chamada 'Guerra dos Browsers'. Como toda guerra tem suas sequelas não seria diferente com essa. Não é garantido que uma página HTML seja apresentada da mesma forma em diferentes navegadores.

É importante que você sempre siga os padrões do W3C para evitar dores de cabeça.



TAGs

Tags, etiquetas ou marcadores são rótulos usados para informar ao navegador como deve ser apresentado o website. As tags são os comandos de formatação da linguagem HTML.

Todas as tags começam com um sinal de menor "<" e acabam com um sinal de maior ">". (também conhecidos como parênteses angulares).

Exemplo:
<comando> texto livre </comando>

Observe que toda tag deve ser "fechada", isto é, sempre deve existir uma tag de abertura e uma de fechamento. A diferença entre elas é que a tag de fechamento tem uma barra "/" após o sinal de menor "<". Existe, porém, uma exceção: algumas tags possuem o fechamento na própria tag, como no exemplo:
<comando />

As tags HTML não são "case sensitive", ou seja
 é igual a <BR>. Muitos desenvolvedores escrevem suas tags em maiúsculo, outros preferem em minúsculo e existem aqueles que misturam tudo.

O W3C recomenda que você escreva suas tags em minúsculo (lowercase). Essa é a recomendação para o HTML 4. Já para o XHTML (próxima geração do HTML) isso é obrigatório.

Portanto, escreva todas as tags em caixa baixa (lowercase).

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
 <head>
  <title>sample</title>
 </head>
 <body>
  <p>Voluptatem accusantium
   totam rem aperiam.</p>
 </body>
</html>
```

HTML

Atributos

Muitas vezes a tag em si só não diz muita coisa. Ela precisa ser complementada com atributos. Os atributos fornecerão informação adicional à tag.

Exemplo:
<table border="0" name="dados">

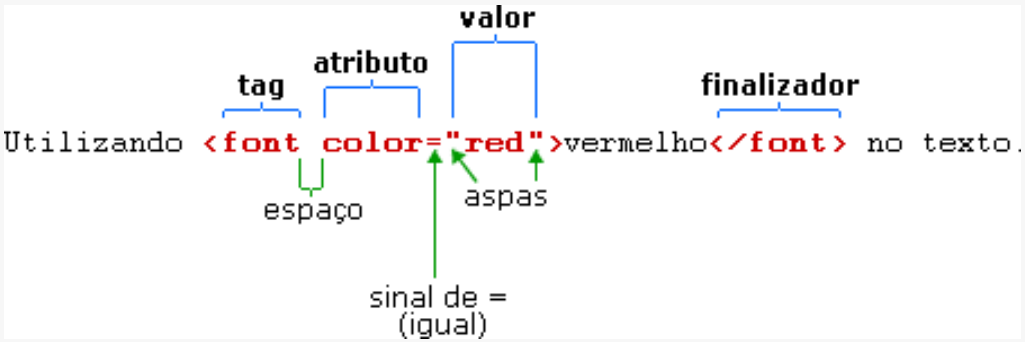
Observe que o atributo também está em lowercase, assim como o seu valor.

No caso acima a tag é a <table>. Os atributos são **border** e **name**. Os valores dos atributos vêm logo após o sinal de igual e estão entre aspas. É possível informar o valor sem as aspas, mas é uma recomendação do W3C que se utilize as aspas. É comum que se use as aspas duplas, mas as aspas simples (apóstrofos) também são aceitas.

As tags HTML não são "case sensitive", ou seja
 é igual a <BR>. Muitos desenvolvedores escrevem suas tags em maiúsculo, outros preferem em minúsculo e existem aqueles que misturam tudo.

O W3C recomenda que você escreva suas tags em minúsculo (lowercase). Essa é a recomendação para o HTML 4. Já para o XHTML (próxima geração do HTML) isso é obrigatório.

Portanto, escreva todas as tags em caixa baixa (lowercase).



Editores | Extensão

Que programa deve-se utilizar para escrever documentos HTML?

De preferência editores que não formatam o texto, tais como Notepad, Notepad++, Textpad, etc.

Durante os exemplos apresentados utilizaremos o Notepad++.

Serão apresentadas algumas ferramentas visuais que ajudam muito no desenvolvimento das páginas.

Mas para o aprendizado é melhor escrever seu código HTML em um editor simples. Depois que tiver dominado a linguagem poderá passar para uma dessas ferramentas visuais.

Que extensão deve-se utilizar ao salvar a página? Html ou htm?

No passado era comum salvarmos arquivos com uma extensão de 3 letras. Herança do DOS.

Portanto não existe nenhum problema em salvar os arquivos com a extensão html. Apenas programas bem antigos é que reconhecem extensões com apenas 3 letras.

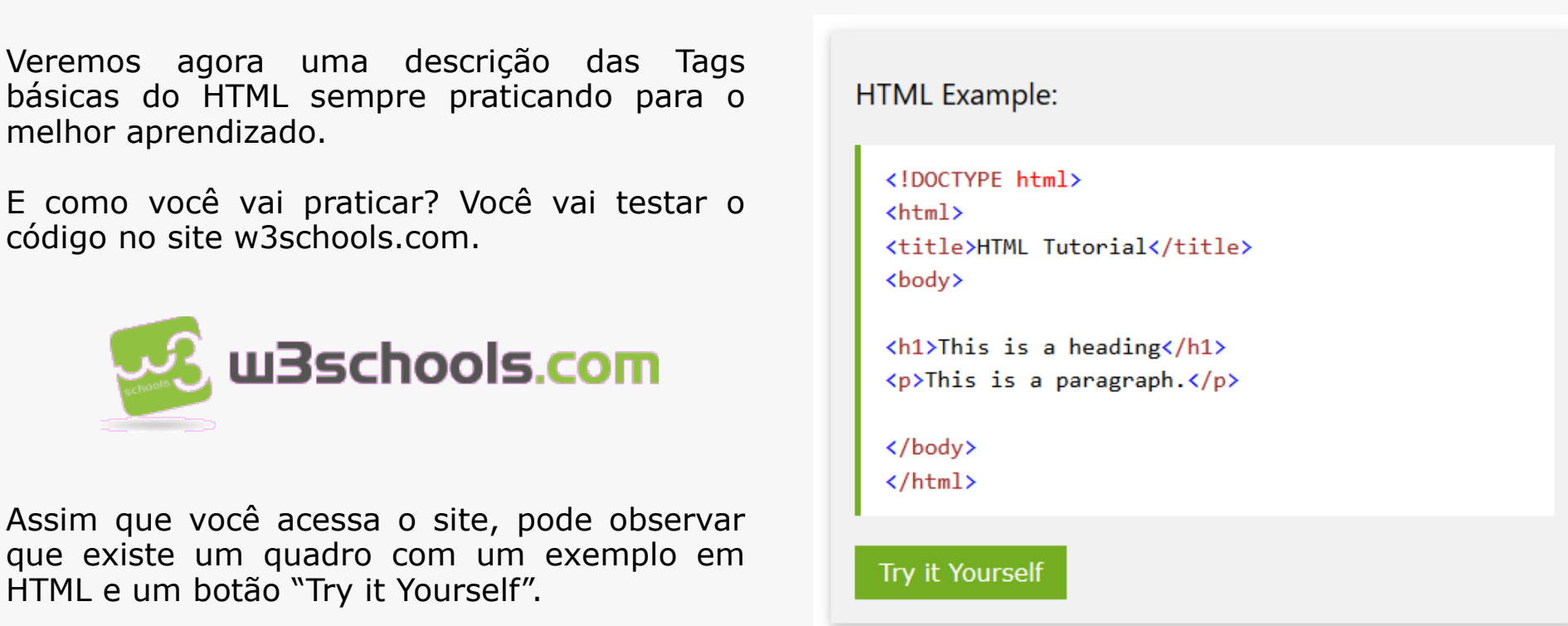
TAGs Básicas – w3schools

Veremos agora uma descrição das Tags básicas do HTML sempre praticando para o melhor aprendizado.

E como você vai praticar? Você vai testar o código no site w3schools.com.

Assim que você acessa o site, pode observar que existe um quadro com um exemplo em HTML e um botão "Try it Yourself".

Quando clicar no botão, você verá que é possível alterar o código e observar o resultado em HTML. É essa ferramenta que você utilizará durante a leitura para realizar testes e aprender HTML.



Após clicar no botão "Try it Yourself", o editor de HTML do w3schools será exibido, conforme imagem a seguir. Para ver o resultado do texto editado, basta clicar no botão "See Result".

TAGs Básicas – w3schools

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<head>
<title>Albert Eije</title>
</head>
<body>

<h1>AlbertEije.COM</h1>
<p>Este é um parágrafo.</p>

</body>
</html>
```

Result:

AlbertEije.COM

Este é um parágrafo.

TAGs Básicas

Vamos começar com a tag mais básica de todas.

<html>

Essa tag informa ao browser que se trata de uma página HTML. Na linguagem HTML padrão não existe utilização de atributos para essa tag.

Quando trabalharmos com XHTML será necessário informarmos alguns atributos.

Teste (copie o texto abaixo para o editor do w3schools):

```
<html>
AlbertEije.COM
</html>
```

<body>

Essa tag define o corpo da página. Ela contém todo o conteúdo da página (texto, imagens, cores, gráficos, etc).

Essa tag tem diversos atributos que já estão "deprecated", ou seja, ainda são suportados pelos browsers, mas podem tornar-se obsoletos no futuro e aí já não há garantias de que serão suportados. Veremos esses atributos por uma questão de aprendizado e suporte. Você aprenderá que eles devem ser utilizados em folhas de estilo (CSS) e não na própria tag.

Faça o teste abaixo com a tag body.

```
<html>
<body link="blue" alink="red" vlink="green">
      <a href="http://www.t2ti.com">T2Ti.com</a> <br />
      <a href="http://www.uol.com">UOL</a> <br />
      <a href="http://www.alberteije.com">alberteije.com</a> <br />
      <a href="http://www.google.com">Google</a> <br />
</body>
</html>
```

TAGs Básicas

<h1> a <h6>

Essas tags servem para definir cabeçalhos no texto. <h1> define o maior cabeçalho e <h6> o menor.

Essas tags tem um atributo align que também encontra-se "deprecated" e deve ser utilizado com folhas de estilo.

```
<html>
<body>
      <h1 align="center">AlbertEije.com</h1>
      <h2 align="center">AlbertEije.com</h2>
      <h3 align="center">AlbertEije.com</h3>
      <h4 align="center">AlbertEije.com</h4>
      <h5 align="center">AlbertEije.com</h5>
      <h6 align="center">AlbertEije.com</h6>
</body>
</html>
```

<p>

Define um parágrafo. Possui uma tag de abertura de uma de fechamento.

Assim como as tags anteriores seus atributos estão "deprecated", sendo que para estilizar o parágrafo deve-se utilizar CSS.

```
<html>
<body>
      Albert
      <p>AlbertEije.com</p>
      <p align="center">AlbertEije.com</p>
      Eije
</body>
</html>
```

TAGs Básicas

Esta tag serve para inserir uma linha em branco. Não adianta escrever com vários espaços e nem inserir várias linhas entre as tags.

Observe que esta é uma "empty tag", ou seja, uma tag vazia, ela não tem fechamento. No XHTML todas as tags deverão ser fechadas. Como se faz isso com a
? Simples: inserimos a barra de fechamento no final da tag <br />. Utilize essa tag para entrar com linhas em branco e não para separar parágrafos.

```
<html>
<body>
      Albert <br>
      AlbertEije.COM <br />
      Eije <br />
</body>
</html>
```

<hr>

Insere uma linha horizontal. Da mesma forma que a tag <br> esta tag não possui fechamento.

Todos os atributos devem ser estilizados com CSS. Observe no código de teste como usar atributos diretamente no HTML.

```
<html>
<body>
      Albert <br>
      AlbertEije.COM
      <hr width="400" align="left" size="10">
      Eije
</body>
</html>
```

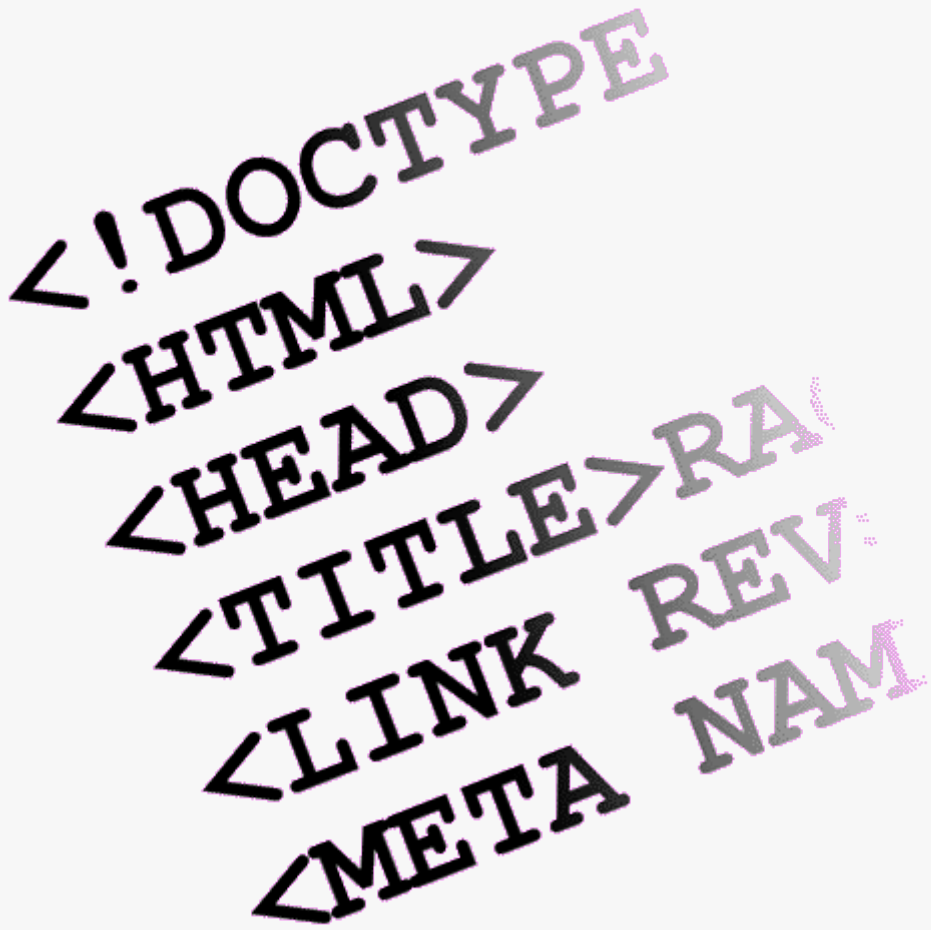
TAGs Básicas

<!-- comentário -->

Toda linguagem tem um recurso para que o desenvolvedor insira comentários. Não seria diferente com o HTML. Utilize a tag acima para inserir comentários em suas páginas.

O texto entre as tags não aparecerá para o usuário. No entanto se o usuário visualizar o código fonte da página o comentário poderá ser visto.

```
<html>
<body>
      <!--Título da página -->
      Albert <br>
      AlbertEije.com <br>
      <!-- Menu da página -->
      Eije
      <!-- complemente esta página -->
</body>
</html>
```



TAGs para Formação de Texto

É importante saber que a formatação deve ficar por conta da folha de estilo, ou CSS. No entanto, como estamos estudando pura e simplesmente o HTML, veremos quais tags servem para formatar o texto.

<b>

Formata um texto para negrito.

<i>

Formata um texto para itálico.

```
<html>
<body>
    <b>
        <i>www.alberteije.com</i>
    </b>
    <br />
    <i>Albert Eije</i>
</body>
</html>
```

<big>

Aumenta o tamanho de um texto.

<small>

Diminui o tamanho de um texto.

Texto enfatizado. Parecido com itálico.

Texto "forte". Parecido com negrito.

```
<html>
<body>
    <big>www.alberteije.com</big><br />
    <small>Albert Eije</small><br />
    <em>T2Ti</em><br />
    <strong>Estou aprendendo HTML!
</strong>
</body>
</html>
```

TAGs para Formação de Texto

<sub>

Formata o texto em subscrito.

<sup>

Formata o texto em sobrescrito.

```
<html>
<body>
	Pedro foi o 5<sup>o</sup> colocado
	<br />
	<br />
	X<sup>2</sup> + Y<sup>2</sup> = 100
	<br />
	<br />
	A<sub>01</sub> + A<sub>02</sub> + a<sub>03</sub> + ... + a<sub>0n</sub> é a soma de alguns termos de uma matriz.
	<br />
	<br />
	Teste <sup>Albert Eije</sup>
</body>
</html>
```

TAGs para Formação de Texto

<ins> e <del>

Essas tags são muito utilizadas em textos oficiais, como leis, por exemplo. A tag <ins> indica que um texto foi inserido no documento e a tag <del> indica que o texto foi removido. Ambas possuem atributos que indicam a data da modificação e direcionam para uma URL que explica a razão da modificação.

```
<html>
<body>
	<ins cite="http://www.google.com" datetime="20151225">Pedro foi o 5<sup>o</sup> colocado</ins>
	<br />
	<br />
	<del>A lei afirma que o universo é a soma de alguns termos de uma matriz.</del>
</body>
</html>
```

TAGs para Formação de Texto

<s> e <strike>

Essas tags têm o mesmo efeito da tag <del>. Estão "deprecated" e deve-se utilizar a <del> em seu lugar.

<u>

Tag que serve para sublinhar um texto. Essa tag também está "deprecated", ou seja, é bom que não seja utilizada. Além disso sublinhar um texto confunde um usuário que pode achar que é um link.

```
<html>
<body>
	<strike>Pedro foi o 5<sup>o</sup> colocado</strike>
	<br />
	<br />
	<s>A lei afirma que o universo é a soma de alguns termos de uma matriz.</s>
	<br />
	<br />
	<u>www.alberteije.com</u>
</body>
</html>
```

TAGs – Computer Output

Existem algumas tags que simulam a "saída do computador". Como se fosse um texto pré-formatado. Vejamos.

<code> e <kbd>

Formata um texto para código de computador.

<samp>

De sample. Uma amostra de código de computador.

<tt>

'Teletype text'. Todas essas tags tem uma saída semelhante.

TAGs – Computer Output

```
<html>
<body>
	Saída na tag code: <br />
	<code>
		for(int i=0;i<10;i++){<br />
			     print(i);<br />
		}
	</code>
	<hr />

	Saída na tag kbd: <br />
	<kbd>
		for(int i=0;i<10;i++){<br />
			     print(i);<br />
		}
	</kbd>
	<hr />
</body>
</html>
```

TAGs – Computer Output

```
<html>
<body>
	Saída na tag samp: <br />
	<samp>
		for(int i=0;i<10;i++){<br />
			     print(i);<br />
		}
	</samp>
	<hr />

	Saída na tag tt: <br />
	<tt>
		for(int i=0;i<10;i++){<br />
			     print(i);<br />
		}
	</tt>
</body>
</html>
```

TAGs – Computer Output

<pre>

Define um texto pré-formatado. O texto escrito nesta tag preserva os espaços e as quebras de linha.

Esta tag possui um atributo width que está "deprecated". Veremos que esse atributo formata o número máximo de caracteres por linha.

```
<html>
<body>
	Código fonte do programa laço:
	<pre width="20">
int i;
for(i=0; i<100; i++) {
	System.out.println("Número a ser mostrado: " + i);
}
	</pre>
</body>
</html>
```

TAGs – Citação e Trecho

Nas tags de formatação de texto existem aquelas que servem para fazer citações e separar trechos de texto.

<abbr> e <acronym>

<abbr> indica um texto abreviado (etc., Inc.) e <acronym> indica o uso de acrônimos (www). Por marcar um texto com essas tags você fornecerá informações úteis para os browsers, verificadores ortográficos, sistemas de tradução e ferramentas de busca.

Em alguns browsers utiliza-se o atributo title para mostrar a expressão completa quando o mouse passar por cima da abreviatura ou do acrônimo.

```
<html>
<body>
	Exemplo de utilização das tags abbr e acronym
	<br />
	<abbr title="Incorporation">Inc.</abbr>
	<br />
	<br />
	<acronym title="World Wide Web">www</acronym>
</body>
</html>
```

TAGs – Citação e Trecho

<address>

Tag usada para definir endereços, assinaturas ou informações sobre o autor de determinado documento.

Normalmente o browser mostra o texto desta tag em itálico. A maioria vai inserir uma linha antes e depois do elemento.

<bdo>

Usada para definir a direção do texto. Atributo a ser utilizado: dir.

```
<html>
<body>
	Água mole em pedra dura tanto bate até que fura
	<address>
	Provérbio sei lá de onde <br />
	Autor Desconhecido
	</address>
	<br /><br />
	<bdo dir="rtl">Posso ler este texto, mas se eu quisesse ler em hebraico?</bdo>
</body>
</html>
```

TAGs – Citação e Trecho

<blockquote>

Define um bloco de texto. Essa tag vai inserir espaços em branco em ambos os lados do texto.

É comum utilizar essa tag quando a página precisa citar algo dito por alguém ou escrito em outra página.

Existe o atributo *cite* onde pode-se informar a URL de onde a citação foi retirada.

```
<html>
<body>
	Provérbios
	<blockquote cite="http://www.proverbios.com.br">
	Água mole em pedra dura tanto bate até que fura.
	Água mole em pedra dura tanto bate até que fura.
	Água mole em pedra dura tanto bate até que fura.
	Água mole em pedra dura tanto bate até que fura.
	</blockquote>
	Fim dos provérbios
</body>
</html>
```

TAGs – Citação e Trecho

<q>

Tem o mesmo objetivo da tag <blockquote>, sendo que a primeira é usada para uma citação longa e a última para uma citação curta.

Segundo as especificações, esta tag deveria mostrar o texto entre aspas duplas. Isso funciona perfeitamente nos browsers da família mozzila (firefox e opera), mas o IE ignora.

Possui o mesmo atributo *cite* que a tag <blockquote>.

```
<html>
<body>
    Assim como disse o sábio <q> Água mole em pedra dura tanto bate até que fura </q>, tu
    deves caminhar sempre pela sombra...
</body>
</html>
```

Entities

Existem alguns caracteres que são tratados de forma diferente pelos browsers. Por exemplo, os sinais de menor e maior servem para formar as tags. Para mostrar um sinal como esse deve-se utilizar um caractere especial (*character entity*). Um caractere especial é formado de 3 partes:

1 – um "e" comercial (&);
2 – o nome da entity ou um # acompanhado do número da entity;
3 – e no final um ponto-e-vírgula.

A vantagem de usar o nome é que é fácil de lembrar, mas nem todos os browsers suportam o uso do nome, sendo que a maioria suporta o uso do código. Observe abaixo uma lista dos caracteres especiais mais comuns:

Resultado	Descrição	Nome da Entity	Número da Entity
	Espaço em branco		
<	Menor que	&lt;	&#60;
>	Maior que	&gt;	&#62;
&	E comercial	&amp;	&#38;
"	Aspas duplas	&quot;	&#34;
'	Apóstrofo	' (não funciona no IE)	&#39;

Links

Uma das coisas mais fascinantes do HTML é a possibilidade de navegar pela Internet através de cliques. Isso é possível graças aos links.

Um link pode apontar para qualquer recurso da web: uma página, uma imagem, um som, um vídeo, um arquivo, etc.

É possível até mesmo apontar para outras seções da mesma página. Para conseguir tudo isso vamos estudar apenas uma tag e alguns atributos.

<a>

"a" de Anchor (âncora). É isso mesmo que essa tag faz: cria uma âncora ou link para outro recurso.

A tag não pode ser utilizada sozinha pois não terá efeito. Ela sempre deve vir acompanhada do atributo href.

Exemplo:
 clique aqui

Links

Outro atributo a ser utilizado é o target. Com ele podemos forçar o conteúdo do link a ser carregado numa janela alvo.

Exemplo:
 clique aqui

Alguns valores do atributo target são os seguintes:

target="_blank" - O documento será carregado em uma nova janela.

target="_self" – O documento será carregado na mesma janela.

Podemos usar a tag <a> com o atributo name. Neste caso estaremos criando um link para uma seção da própria página.

Exemplo:
 Seção Teste

Dessa forma definimos uma seção na página. Para direcionarmos a página diretamente para essa seção fazemos como no exemplo.

Exemplo:
 clique para ir para teste

Links

```
<html>
<body>
	<a href="http://www.alberteije.com" target="_blank"> Visite o site AlbertEije.COM</a>
	<br /><br />
	<a	href="http://upload.wikimedia.org/wikipedia/meta/2/2a/Nohat-logo-nowords-bgwhite-200px.jpg">Imagem Wikipedia</a>
	<br /><br />
	<a href="http://www.alberteije.com/livros/gratis/aprenda_a_programar_henrique_dias.pdf">Aprenda a Programar</a>
	<h1>Utilização de links para seções</h1>
	<a href="#secao1">Seção 01</a><br />
	<a href="#secao2">Seção 02</a><br />
	<a href="#secao3">Seção 03</a>
	<br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
	<h1><a name="secao1">Aqui fica a primeira seção</a></h1>
	seção 01 <br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
	<h1><a name="secao2">Aqui fica a segunda seção</a></h1>
	seção 02 <br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
	<h1><a name="secao3">Aqui fica a terceira seção</a></h1>
	seção 03 <br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
</body>
</html>
```

Frames

O objetivo dos frames é mostrar mais de uma página HTML na mesma janela do browser.

Cada frame deverá conter uma página HTML diferente.

Existe uma discussão sobre o uso de frames. Uns defendem seu uso, mas a maioria dos desenvolvedores o condenam.

Vantagens:

- Navegação da página mais rápida. O primeiro carregamento será igual, mas em sucessivas chamadas de páginas já teremos algumas janelas salvas, evitando novamente o carregamento das mesmas.
- Processo de desenvolvimento de páginas se torna mais rápido. Isso se dá porque não há a necessidade de fazer um novo menu ou um novo título.
- Partes da página que se tornam visíveis constantemente, se contiverem links, podem servir muito bem para melhorar a navegação pelo site.

Desvantagens:

- Tiram espaço da tela.
- Forçam o visitante a entrar pela declaração de frames. Se o internauta entrar em uma página interior não verá o título e nem o menu e isso dará uma má impressão.
- Só deve-se promover a página com os frames, pois promover as páginas interiores causaria o problema anterior.
- Adicionar a página aos favoritos pode se tornar um problema.
- Em muitos casos será necessário o uso de JavaScript, que pode se comportar de forma diferente entre os browsers e isso pode se tornar um problema.

Frames

<frameset>

Esta tag serve para dividir a janela em frames. Cada frameset define um conjunto de linhas ou colunas.

Para o uso correto desta tag precisaremos utilizar os atributos rows e cols.

Exemplo (abaixo o código fonte da página animais.html):

```
<html>
<frameset cols = "25%, 25%,*">
  <frame src ="cachorro.html" />
  <frame src ="gato.html" />
  <frame src ="cavalo.html" />
</frameset>
</html>
```

No caso do exemplo anterior, será preciso ter as demais páginas para testar. Para realizar o teste no w3schools, troque o "scr" por outras três páginas existentes na Internet. Cada página será carregada num frame.

Frames

<frame>

É uma sub-janela. Observe no exemplo abaixo que um frameset é formado por vários frames. Esta tag possui vários atributos.

Exemplo (abaixo o código fonte da página animais.html):

```
<html>
<frameset cols = "25%, 25%,*">
  <frame src ="cachorro.html" />
  <frame src ="gato.html" />
  <frame src ="cavalo.html" />
</frameset>
</html>
```

No caso do exemplo anterior, será preciso ter as demais páginas para testar. Para realizar o teste no w3schools, troque o "scr" por outras três páginas existentes na Internet. Cada página será carregada num frame.

Frames

<noframes>

Vimos que quando utilizamos frames a página HTML não possui o elemento <body>. No entanto pode existir um browser que não tem suporte a frames. Neste caso utilizamos a tag <noframes>. Observe o exemplo.

Exemplo (abaixo o código fonte da página animais.html):

```
<html>
<frameset cols = "50%, 50%">
  <noframes>
      <body>Este browser não suporta frames</body>
  </noframes>
  <frame src ="cachorro.html" />
  <frame src ="gato.html" />
</frameset>
</html>
```

No caso do exemplo anterior, será preciso ter as demais páginas para testar. Para realizar o teste no w3schools, troque o "scr" por outras três páginas existentes na Internet. Cada página será carregada num frame.

Frames

<iframe>

Esta tag cria um frame dentro da sua página HTML normal. Não é necessário utilizar a tag <frameset> para isso.

Assim como a tag <frame> esta tag possui vários atributos.

Exemplo:

```
<html>
  <body>
      utilização de iframe.
      <br />
      <iframe src ="http://www.alberteije.com/livros/xdk/" width="100%"> </iframe>
  </body>
</html>
```

Tabelas (Tables)

Uma Table nada mais é do que uma tabela. Para criar uma Table precisamos abrir linhas e em cada linha inserir células que conterão os dados.

Cada célula pode conter texto, imagens, forms, outras tabelas, etc.

A Table também possui tags específicas para o Cabeçalho e para o Rodapé.

Muitos desenvolvedores ainda utilizam Tables para fazer o layout de suas páginas. Nós faremos isso! Mas o objetivo das Tables é apresentar dados. Veremos como fazer layouts rebuscados quando estudarmos CSS. Vejamos as tags que formam a Table.

<table>

Esta é a tag que abre a Table. Ela possui diversos atributos.

<tr>

Tag que serve para inserirmos linhas na Table.

<td>

Tag para a inserção de células nas linhas.

Tabelas (Tables)

<caption>

Esta tag define um título para a Table. Esta tag deve ser inserida imediatamente após a tag <table>.

<th>

Define um cabeçalho para as células da Table, ou seja, cada coluna terá um cabeçalho. Por padrão o texto desta tag será apresentado em negrito e centralizado.

Title	Title	Title	Title	Title	Title
Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data

Tabelas (Tables)

```
<html>
<body>
    <table border="1">
        <caption>Tabela Exemplo</caption>
        <tr>
            <th>Coluna 1</th>
            <th>Coluna 2</th>
        </tr>
        <tr><td>
                T2ti.com
            </td>
            <td>
                Albert Eije Barreto Mouta
        </td></tr>
        <tr><td>
                Tabelas
            </td>
            <td>
                 
        </td></tr>
    </table>
</body>
</html>
```

Tabelas (Tables)

<colgroup>

Esta tag define grupos de colunas em uma Table. É útil agrupar colunas para formatação.

<col>

Tag utilizada em conjunto com a <colgroup>. O objetivo é definir valores para atributos de uma ou mais colunas na Table.

```
<html><body>
    <table border="1"><caption>Tabela Exemplo</caption>
        <colgroup span="3">
            <col width="100" align="center"></col>
            <col width="250" align="center"></col>
            <col width="200" align="center"></col>
        </colgroup>
        <tr><th>Coluna 1</th><th>Coluna 2</th><th>Coluna 3</th><th>Coluna 4</th>
        </tr>
        <tr><td>T2ti.com</td><td>Albert Eije</td><td>Tabelas</td><td>Teste</td>
        </tr>
    </table>
</body></html>
```

Tabelas (Tables)

<thead> <tbody> <tfoot>

Estas três tags são utilizadas para agrupar linhas na Table, otimizando sua estrutura. O principal objetivo é que você separe as linhas de cabeçalho daquelas do corpo e do rodapé. Numa Table muito grande, ao imprimir seu conteúdo, o browser repetiria automaticamente o cabeçalho e o rodapé em todas as páginas. Infelizmente este recurso não funciona em alguns dos mais recentes browsers.

```
<html><body>
    <table border="1">      <caption>Tabela Exemplo</caption>
        <thead>  <tr><th>Coluna 1</th><th>Coluna 2</th></tr>  </thead>
        <tfoot>
        <tr><th>Rodapé 1</th><th>Rodapé 2</th></tr>
        </tfoot>
        <tbody>
        <tr> <td>AlbertEije.COM</td> <td>Albert Eije Barreto Mouta</td> </tr>
        <tr> <td>Tabelas</td> <td> </td> </tr>
        <tr> <td>AlbertEije.COM</td> <td>Albert Eije Barreto Mouta</td> </tr>
        <tr> <td>Tabelas</td> <td> </td> </tr>
        </tbody>
    </table>
</body></html>
```

Listas

Vamos aprender sobre três tipos de listas: ordenadas (ordered – ol), não ordenadas (unordered – ul) e de definição (definition – dl). Cada uma destas listas tem uma característica que a distingue das demais.

Unordered List – lista não ordenada. Esta lista será formada por bullets (símbolos gráficos), na maioria das vezes pequenos círculos pretos. A lista é formada pela tag <ul> e cada ítem da lista pela tag <li>. Observe o exemplo:

```
<html>
<body>
	<!--Lista não ordenada-->
	<h1>Lista não ordenada</h1>
	<ul type="disc">
		<li>Cachorro</li>
		<li>Gato</li>
		<li>Cavalo</li>
		<li>Leão</li>
	</ul>
</body>
</html>
```

Listas

Ordered List – lista ordenada. Esta lista será formada números. A lista é formada pela tag <ol> e cada ítem da lista pela tag <li>. Observe o exemplo:

```
<html>
<body>
	<!--Lista ordenada-->
	<h1>Lista ordenada</h1>
	<ol type="1" start="2">
		<li>Cachorro</li>
		<li>Gato</li>
		<li>Cavalo</li>
		<li>Leão</li>
	</ol>
</body>
</html>
```

Listas

<dl> <dt> <dd>

Definition List – lista de definição. Esta não é uma lista de itens, mas uma lista de termos e explanações desses termos.
<dl> define a lista;
<dt> uma entrada na lista;
<dd> definição do termo.

```
<html>
<body>
	<!--Lista de definição-->
	<h1>Lista de definição</h1>
	<dl>
		<h3><dt>Segunda-Feira</dt></h3>
			<dd>É o primeiro dia da semana. Dia em que muita gente fica
			chateada porque tem que ir trabalhar!</dd>

		<h3><dt>Terça-Feira</dt></h3>
			<dd>É o segundo dia da semana. Já se começa a aucostumar-se
			com o trabalho, mas a maioria tá esperando pela sexta!</dd>
	</dl>
</body>
</html>
```

Forms

Em linguagens desktop, como o Delphi ou o VB, você pode inserir diversos itens como Labels, Edits, Comboboxes, Botões, etc.

Como fazer isso no HTML? Isso foi pensado e este recurso é disponibilizado através dos forms. Isso mesmo, é dentro dos forms HTML que inserimos todos esses elementos.

Além disso os forms têm um papel imprescindível na interatividade da página. Através deles os usuários poderão enviar informações para o site e tais informações poderão ser armazenadas em bancos de dados, por exemplo. Vejamos em detalhes as características dos forms.

<form>

Esta tag serve para inserir um form na página. Ela sozinha não faz nada. São necessários os outros elementos para completar o form.

<input>

A maioria das tags dentro do form são do tipo <input>. Para identificar cada input utilizamos o atributo type.

O código do exemplo está dividido nas próximas três páginas. Copie o código de cada página até que fique completo no editor do w3schools para estudá-lo.

Forms

```
<html>
<body>
	<h1>Formulários</h1>
	<hr />
	<form name="formulario" action="http://websro.correios.com.br/sro_bin/txect01$.QueryList"
method="post" target="_blank">
		<table>
			<tr>
				<td align="right">Tipo:</td>
				<td><input type="text" name="P_TIPO" value="alberteije.com"
readonly="readonly"> </input></td>
			</tr>
			<tr>
				<td align="right">Senha:</td>
				<td><input name="senha" type="password" maxlength="10" size="10">
</input></td>
			</tr>
			<tr>
				<td align="right">Lingua:</td>
				<td><input type="text" name="P_LINGUA"> </input></td>
			</tr>
			<tr>
				<td align="right">Código:</td>
				<td><input type="text" name="P_COD_UNI"> </input></td>
			</tr>
```

Forms

```
<tr>
                    <td align="right">Sexo:</td>
                    <td>
                        <input type="radio" name="sexo" value="Masculino"
checked="checked">Masculino </input>
                        <input type="radio" name="sexo" value="Feminino">Feminino </input>
                    </td>
                </tr>
                <tr>
                    <td align="right" valign="top">    Bens:</td>
                    <td>
                        <input type="checkbox" name="bens" value="Carro"
checked="checked">Carro</input><br />
                        <input type="checkbox" name="bens" value="Moto">Moto
</input><br />
                        <input type="checkbox" name="bens" value="TV"
checked="checked">TV</input><br />
                        <input type="checkbox" name="bens"
value="Micro">Micro</input><br />
                    </td>
                </tr>
```

Forms

```
<tr>
                    <td align="right">Upload:</td>
                    <td><input type="file"></input></td>
               </tr>
               <tr>
                    <td align="right">Imagem:</td>
                    <td><input                                                    type="image"
src="http://www.alberteije.com/infra/logo_alberteije.png"></input></td>
               </tr>
               <tr>
                    <td><input type="hidden" value="teste"></input></td>
               </tr>
               <tr>
                    <td>
                         <input type="button" value="button"></input>
                         <input type="submit" value="submit"></input>
                         <input type="reset" value="reset" disabled="disabled"></input>
                    </td>
               </tr>
          </table>
     </form>
</body>
</html>
```

Forms

<textarea>

Um elemento interessante e muito importante para se utilizar nos forms. Trata-se de uma área para digitar um texto mais longo.

Esta tag é muito utilizada nos formulários de contato onde o usuário digita seu nome, e-mail e logo após um texto descritivo. O local onde o usuário digita este texto descritivo é a textarea.

```
<textarea rows="4" cols="50">
Exemplo de área de texto. Teste lá no w3schools.
</textarea>
```

Forms

```
<html><body>
	<h1>Formulário de Contato</h1><hr />
	<form name="formulario" action="#" method="post">
		<table>
			<tr>
				<td align="right">Nome:</td>
				<td><input type="text" name="nome" /></td>
			</tr><tr>
				<td align="right">E-Mail:</td>
				<td><input name="email" type="text" /></td>
			</tr><tr>
				<td align="right">Texto:</td>
				<td><textarea name="texto" rows="5" cols="20">Digite o texto
aqui</textarea></td>
			</tr><tr align="center">
				<td>
					<input type="submit" value="submit" /></input>
					<input type="reset" value="reset" /></input>
				</td>
			</tr>
		</table>
	</form>
</body></html>
```

Forms

<fieldset>

Esta tag desenha uma borda ao redor dos elementos contidos na mesma.

<legend>

Esta tag define um caption para o <fieldset>.

```
<html><body>
	<h1>Formulário de Contato</h1><hr />
	<form name="formulario" action="#" method="post">
		<fieldset>
		<legend>Informações Gerais</legend>
			Nome:
			<input type="text" name="nome" />
			E-Mail:
			<input name="email" type="text" />
		</fieldset>
		<br />
		<input type="submit" value="submit" /></input>
		<input type="reset" value="reset" /></input>
	</form>
</body></html>
```

Forms

<select>

Cria uma lista do tipo drop-down (combobox). É possível, no entanto, transformá-la em uma lista aberta utilizando um de seus atributos.

<option>

Cada opção que aparece na tag <select>.

<optgroup>

Serve para agrupar options. Quando se tem diversas opções pode ser interessante agrupá-las.

```
<html><body><h1>Formulário de Contato</h1><hr />
    <form name="formulario" action="#" method="post">
        <select name="UF" size="5" multiple="multiple">
            <optgroup label="Estados com a Letra A">
                <option value="AC">AC</option>
                <option value="AL">AL</option>          </optgroup>
            <optgroup label="Estados com a Letra M">
                <option value="MA">MA</option>
                <option value="MG">MG</option>    </optgroup>  </select><br />
        <input type="submit" value="submit" /></input>
        <input type="reset" value="reset" /></input>   </form>      </body></html>
```

Forms

<label>

Esta tag é utilizada em conjunto com outros elementos do form para facilitar o acesso a esses elementos.

<button>

Existe um input do tipo button, mas esta tag tem uma diferença: você pode inserir conteúdo tais como textos ou imagens.

```
<html><body>
	<h1>Formulário de Contato</h1> <hr />
	<form name="formulario" action="#" method="post">
		<fieldset>
		<legend>Informações Gerais</legend>
			<label for="nome">Nome:</label>
			<input type="text" name="nome" id="nome"/>
			E-Mail:
			<input name="email" type="text" />
		</fieldset>
		<button><img src="http://www.alberteije.com/infra/logo_alberteije.png" /></button>
	</form>
</body></html>
```

Forms

O que seria da Internet se não fossem as imagens? Imagine que navegássemos acessando apenas texto! Seria muito chato.

As imagens são um recurso interessante para incrementar a visualização de um site. Existem sites que só existem por causa de imagens, como sites que vendem fotografias, site de álbuns ou de imobiliárias onde as fotos dos imóveis são apresentadas.

Veremos agora como fazer para trabalhar com imagens numa página HTML.

Tag para inserir uma imagem na página. Como esta é uma empty tag, será necessário um atributo para identificar a imagem. Este atributo é o src de source (origem).

Existem alguns atributos no estado "deprecated", ou seja, devem ser evitados e ao invés de usá-los no próprio código HTML, devem ser inseridos num arquivo CSS a parte.

```
<html><body>
	<a href="http://www.alberteije.com">
	<img src="http://www.alberteije.com/infra/logo_alberteije.png" alt="Logotipo AlbertEije" title="Logo AlbertEije" border="0" />
	</a>
</body></html>
```

Forms

<map>

Este é um recurso bem interessante. Você pode mapear partes da imagem e em cada uma dessa parte inserir um link para um lugar diferente!

<area>

Define uma região na imagem mapeada. Essa região será definida através de coordenadas.

```
<html>
<body>

<p>Clique num dos astros para vê-lo mais de perto:</p>
<img src="planets.gif" width="145" height="126" alt="Planetas" usemap="#planetmap">
<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sol" href="sun.htm">
  <area shape="circle" coords="90,58,3" alt="Mercúrio" href="mercur.htm">
  <area shape="circle" coords="124,58,8" alt="Vênus" href="venus.htm">
</map>

</body>
</html>
```

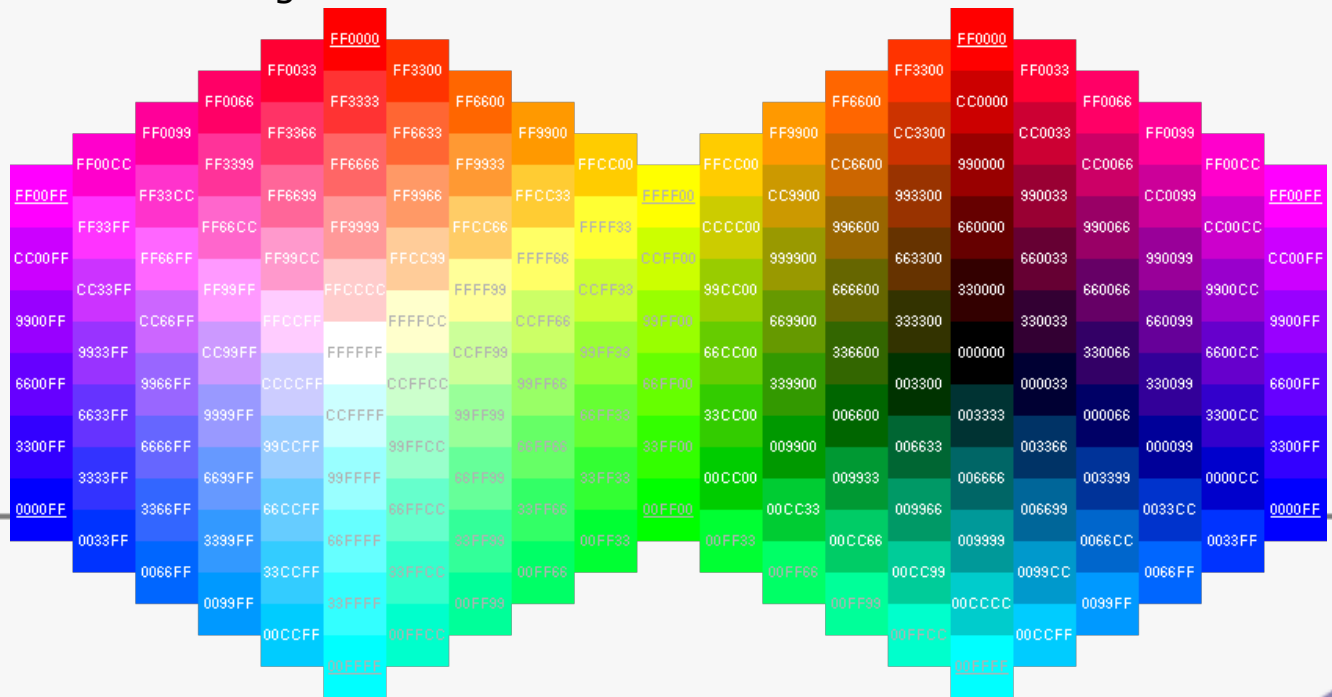
Cores

Outro recurso importantíssimo para as páginas HTML. Imagine se todas as páginas fossem com um fundo branco e a fonte preta! Tão ruim como não puder usar imagens! Felizmente isso não é assim.

Para mostrar tantas cores em nossas páginas devemos usar uma combinação das cores Vermelho, Azul e Verde. No inglês essas cores são Red, Green, Blue. Perceba as iniciais: RGB. Combinando as retículas dessas cores obteremos milhões de outras cores.

Para combinarmos as cores usaremos uma notação Hexadecimal. Dessa forma o menor valor a ser utilizado será o zero (#00 em hexa) e o maior valor será o 255 (#ff em hexa).

A combinação será feita da seguinte maneira: RRGGBB.



Cores

Abaixo podemos observar uma tabela com algumas cores. Ao lado vemos o respectivo código hexa e o número RGB.

Cor	Código Hexa da Cor	Número RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

Cores

Existe ainda a possibilidade de usar o nome da cor.

Por exemplo: aqua, black, blue, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow, etc.

É bom ter cuidado ao utilizar o nome da cor porque não é garantido que todos os browsers suportam o nome. É melhor usar o código hexa.

Ao combinar os códigos RGB você terá nada menos que 16 milhões de cores para escolher! Na verdade um pouco mais! Faça a conta: 256x256x256.

```
<html>
<body>

    <table>
        <tr><td bgcolor="#922209"> Uma Cor </td></tr>
        <tr><td bgcolor="#70A1DD"> Outra Cor  </td></tr>
    </table>

</body>
</html>
```

HEAD

<head>

Esta tag contém informações sobre a página. Tudo que é escrito na tag <head> não deveria ser exibido ao usuário pelo browser. Dentro da tag <head> existem várias tags para serem utilizadas. Eis algumas delas:

<base> <meta> <title> <script>

Vejamos as explicações sobre cada uma dessas tags.

<base>

Define uma URL base para toda a página. Por exemplo, digamos que na página exista referência para várias imagens em outro site.

```
<img src="http://www.alberteije.com/imagens/um.gif" />
<img src="http://www.alberteije.com/imagens/dois.gif" />
```

Você poderia usar a tag <base> para informar a URL acima:

```
<base href="http://www.alberteije.com/imagens/" />
```

E quando fosse mostrar as imagens não precisaria repeti-la:

```
<img src="um.gif" />
<img src="dois.gif" />
```

HEAD

<meta>

Esta tag provê informações sobre a página. Meta significa “informação sobre”. Nesta tag você deverá inserir descrições e palavras-chave para as ferramentas e robôs de busca, como o do google, por exemplo. Esta tag possui muitos atributos.

```
<html>
    <head>
        <title> Você está aprendendo HTML! </title>

        <meta name="author" content="AlbertEije.COM" />
        <meta name="description" content="Site de e-books" />
        <meta name="keywords" content="web interface, javascript, html, html5, css" />
        <meta name="keywords" content="php, python, jsf, asp" />

        <meta name="robots" content="index,follow">
        <meta name="robots" content="noarchive">
<body>
    Teste
</body>
</html>
```

HEAD

<title>

Esta tag define o título da página.

<script>

Embora o HTML possua muitas funções, existem coisas que não dá pra fazer apenas com ele. São necessários scripts para isso. A linguagem de scripts mais utilizada é a JavaScript. Para informar ao browser que vamos inserir um script na página precisamos desta tag. Veja o exemplo:

```
<script type="text/javascript">
      document.write("www.alberteije.com")
</script>
```

HEAD

Segue mais um exemplo para você testar no w3schools:

```
<html>
	<head>
		<title> Você está aprendendo HTML! </title>

		<script type="text/javascript">
			function teste()
			{
				alert('AlbertEije.COM');
			}
		</script>
	</head>
<body>
	<h1>AlbertEije.COM</h1>
	<form>
		<input type="button" value="clique aqui" onclick="teste();" />
	</form>
</body>
</html>
```

DOCTYPE

<!DOCTYPE>

Esta tag deve ser a primeira da página e aparecer antes mesmo da tag <html>. Ela informa ao browser que especificação de HTML ou XHTML o documento está usando. O HTML 4.01 especifica 3 tipos de documentos:

Transitional
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

Strict
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Frameset
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

DOCTYPE

Transitional

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

Deve ser usada na transição de HTML anterior à versão 4 para as tags atuais. Contém 11 elementos de apresentação e um conjunto de atributos já não utilizados na versão Strict. O Transitional faz com que a maioria dos browsers trabalhe em modo de compatibilidade. Isso não é bom. Evite utilizar o Transitional.

Strict

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Enfatiza a separação entre conteúdo e forma e comportamento do usuário; é recomendada pelo W3C para novos arquivos.Deve ser utilizada juntamente com CSS.

Todos os atributos de apresentação devem ficar por conta do CSS. Portanto, este modo é o que deve ser utilizado. Quando você aprender XML e CSS use sempre o modo Strict.

DOCTYPE

Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

Usada em páginas com Frames.

Observe que em cada uma das opções mostradas aparece um arquivo com a extensão DTD (Document Type Definition). Trata-se de um arquivo texto em XML que fornece ao browser informações suplementares sobre a interpretação do código.

Nele estão descritos os elementos que podem ser usados, quais elementos podem estar dentro de outros elementos, que tags descrevem esses elementos, se um elemento tem conteúdo ou não, se é necessário ter tag de abertura e/ou de fechamento. Enfim, tudo que pode ou não ser usado dentro do documento.

FONT

<font>

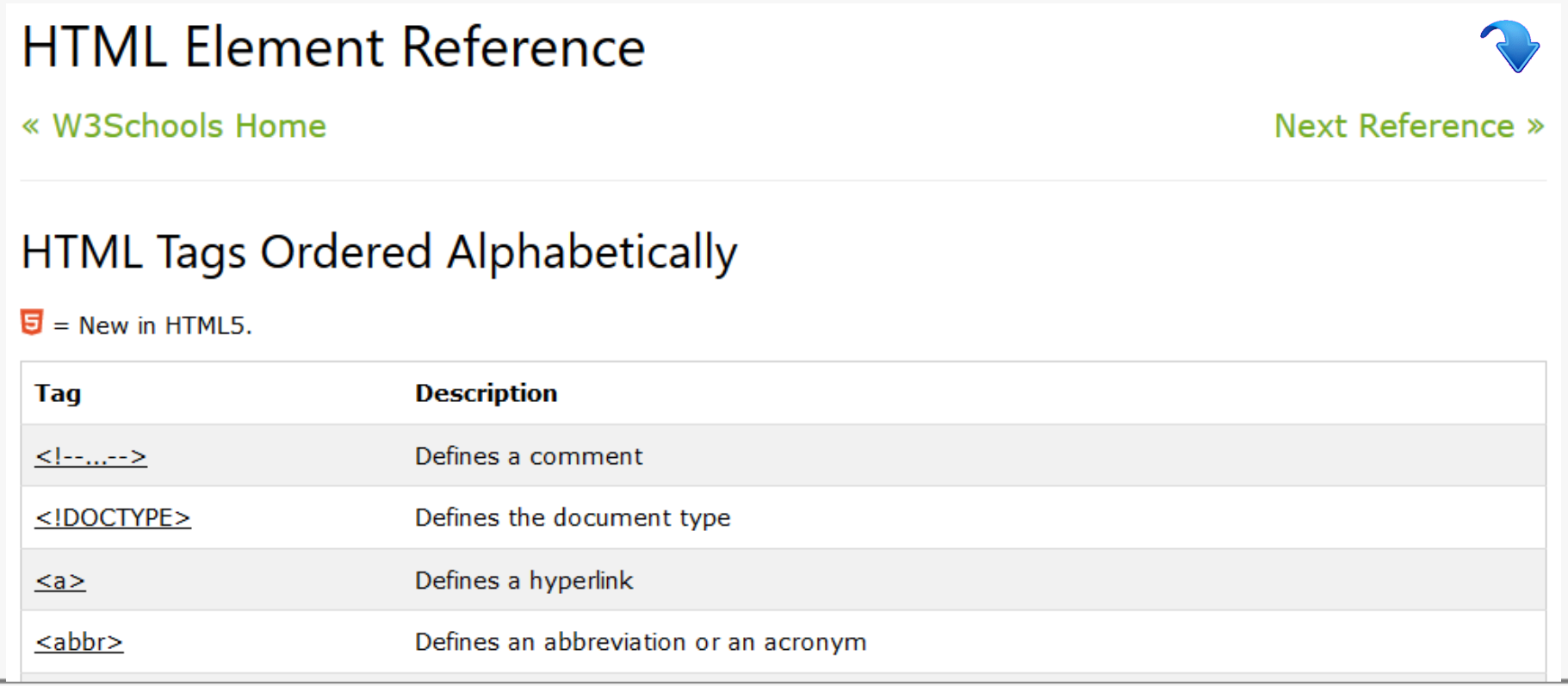
Esta é uma tag que está completamente "deprecated". Isso mesmo! No início o texto era definido com esta tag. Mudar o layout de um site se tornava um pesadelo! Muitas vezes era melhor fazer outro. Veremos agora como utilizar essa tag. Ela será mostrada por uma questão de suporte, já que você pode um dia precisar manter um site antigo onde ela apareça. Não se engane, muita gente ainda usa essa preciosidade.

E como os textos devem ser formatados hoje em dia? CSS!

```
<html>
<body>
	<h1>Trabalhando com a tag Font que está deprecated</h1>	<hr />
	<font face="Arial" size="4" color="#c0c0c0">Fonte Arial</font>
	<hr />
	<font face="Arial Black" size="6" color="#ff0000">Fonte Arial Black</font>
</body>
</html>
```

Explorando as TAGs

E então, vimos todas as TAGs e seus atributos? Não. Você vai explorá-las no site w3schools. Existe uma seção nesse site onde é possível explorar as TAGs já sabendo quais delas estão depreciadas, quais não funcionam no HTML5, etc. Você também verá o suporte por navegador. Clique na imagem abaixo para acessar a página de referência e selecione a TAG que for objeto de estudo.



HTML Element Reference

« W3Schools Home Next Reference »

HTML Tags Ordered Alphabetically

= New in HTML5.

Tag	Description
<!--...-->	Defines a comment
<!DOCTYPE>	Defines the document type
<a>	Defines a hyperlink
<abbr>	Defines an abbreviation or an acronym

Atributos Padrões

Vimos que cada tag possui atributos específicos. Existem porém alguns atributos que todas as tags possuem, com algumas exceções.

Esses atributos são conhecidos como:

Core Attributes – (atributos de núcleo, principais);
Language Attributes – (atributos de linguagem);
Keyboard Attributes – (atributos de teclado);

Atributos Padrões

Core Attributes – (atributos de núcleo, principais)

Atributo	Valor	Descrição
class	*class_rule* ou *style_rule*	A classe do elemento (CSS)
id	*id_name*	Um ID único para um elemento
style	*style_definition*	Estilo definido na tag – inline (CSS)
title	*tooltip_text*	Um texto para mostrar uma dica (hint)

Language Attributes – (atributos de linguagem)

Atributo	Valor	Descrição
dir	ltr | rtl	Define a direção do texto
lang	*language_code*	Define o código da linguagem

Keyboard Attributes – (atributos de teclado)

Atributo	Valor	Descrição
accesskey	*character*	Define uma tecla de atalho para acesso a um elemento
tabindex	*number*	Define a ordem de tabulação de um elemento

Eventos

Podemos utilizar diversos eventos para interagir com o usuário. Dentre os eventos disponíveis temos os seguintes:

Keyboard Events: onkeydown, onkeypress, onkeyup;

Mouse Events: onclick, ondblclick, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup;

Windows Events: onload, onunload;

Form Element Events: onchange, onsubmit, onreset, onselect, onblur, onfocus.

Tais eventos serão abordados quando estudarmos JavaScript.

URL

URL – Uniform Resource Locators

É o endereço de alguma coisa na internet. A URL é formada da seguinte forma:

protocolo://host:porta/path;parâmetros?consulta#fragmento

- Protocolo: o mais comum é o HTTP;
- Host: nome de host (servidor) ou endereço IP;
- Porta: número da porta TCP que o provedor está usando;
- Path: o caminho e a referência do objeto nesse servidor;
- Parâmetros: string com parâmetros de comunicação;
- Consulta: string com parâmetros de um programa;
- Fragmento: referência a um subconjunto de um objeto.

Editores

Existem vários editores para trabalhar com HTML. Talvez você esteja se perguntando se existe algum editor visual, onde você arrasta os elementos e o editor já cria o código. Existe!

Mas fica a sugestão para você estudar HTML usando o NotePad++. Use só o texto mesmo, a princípio, para compreender como a linguagem funciona.

Mas para você não me chamar de malvado, segue um site interessante, que permite você criar tudo de forma visual e online. Não precisa instalar nada na sua máquina. Divirta-se.

HTML 5

HTML 5

Introdução

HTML5 (Hypertext Markup Language, versão 5) é uma linguagem para estruturação e apresentação de conteúdo para a World Wide Web e é uma tecnologia chave da Internet originalmente proposto pela Opera Software.

É a quinta versão da linguagem HTML. Esta nova versão traz consigo importantes mudanças quanto ao papel do HTML no mundo da Web, através de novas funcionalidades como semântica e acessibilidade.

Página do W3C

Possibilita o uso de novos recursos antes possíveis apenas com a aplicação de outras tecnologias. Sua essência tem sido melhorar a linguagem com o suporte para as mais recentes mídias, enquanto a mantém facilmente legível por seres humanos e consistentemente compreendida por computadores e outros dispositivos.

O HTML5 será o novo padrão para HTML, XHTML, e HTML DOM. Atualmente, está em fase de esboço, porém diversos navegadores já implementam algumas de suas funcionalidades.

HTML 5

História

O Web Hypertext Application Technology Working Group (WHATWG) iniciou o trabalho do novo padrão HTML em 2004, quando o World Wide Web Consortium (W3C) estava se concentrando no futuro desenvolvimento do XHTML 2.0, e o HTML 4.01 não tinha sido atualizado desde 2001.

Em 2009, o W3C decidiu que o Grupo de Trabalho do XHTML 2.0 deveria parar seus trabalhos, e assim, descontinuar o padrão. Desta forma o W3C e o WHATWG passaram a trabalhar juntos no desenvolvimento do HTML5.

Em abril de 2010 Steve Jobs emitiu uma carta pública intitulada "Reflexões sobre o Adobe Flash", onde ele conclui o Adobe Flash não seria mais necessário para assistir vídeos ou mesmo exibir qualquer conteúdo web.

No início de novembro de 2011 a Adobe anunciou que iria interromper o desenvolvimento de Flash para dispositivos móveis e redirecionar seus esforços para o desenvolvimento de ferramentas utilizando HTML5.

HTML 5

História

No início de 2008 o W3C – consórcio de empresas de tecnologia que coordena os padrões da internet quanto à linguagem – anunciou a primeira especificação do HTML5.

Foram feitas grandes alterações, que incluem:

- Novas API's, entre elas uma para desenvolvimento de gráficos bidimensionais.
- Controle embutido de conteúdo multimídia.
- Aprimoramento do uso offline.
- Melhoria na depuração de erros.

Após dez anos sem atualizações, a forma como se escreve páginas na internet passa por uma boa transformação. O HTML5 oferece uma experiência web totalmente diferente para usuários e embora exista um longo caminho para ser finalizado, muitos navegadores importantes já implementaram grandes partes da linguagem, incluindo tags de vídeo e suporte à tecnologia Canvas.

Com a evolução da linguagem, os navegadores passam da categoria "mostradores" de páginas para um renderizador de "web software".

Camadas de Desenvolvimento

O desenvolvimento no lado do cliente é baseado em 3 camadas principais: informação, formatação e comportamento.

Embora sejam independentes, a evolução de cada camada influencia o caminho da outra. O CSS não consegue evoluir se o HTML manter-se congelado no tempo.

A camada de informação é a mais importante. Ela vem antes de todas as outras e fica sob o controle do HTML. O HTML marca a informação dando-lhe significado.

A informação precisa ser acessível a qualquer hora, de qualquer lugar e principalmente, por qualquer dispositivo e meio de acesso. Informação é tudo o que o usuário consome.

Desde o início, quando a internet foi planejada e criada, seu objetivo era claro: compartilhar informação com pessoas do mundo inteiro, de forma rápida e dinâmica.

A informação deve ser entregue, não importa se o visual tenha sido prejudicado por falta do CSS ou se o Javascript está desligado no navegador do usuário.

Camadas de Desenvolvimento

A segunda camada é responsável por controlar o visual da informação exibida pelo HTML. É esta camada que deixa o visual legal, bonito. Atualmente essa camada é controlada pelo CSS.

O CSS é a linguagem responsável por controlar o visual da informação exibida pelo HTML. O CSS formatará o conteúdo de forma que seja visualmente agradável em qualquer meio de acesso. O CSS é o responsável por formatar a informação para que ela seja consumida em qualquer meio de acesso de forma simples.

Se a informação deve ser acessada em qualquer lugar, o CSS precisa cuidar para que essa informação apareça de maneira adequada em cada um destes meios de acesso: computadores, TVs, tablets, smartphones etc. Essa é sua principal responsabilidade.

Camadas de Desenvolvimento

A terceira decide quais serão os comportamentos dos elementos. O Javascript é o principal responsável por essa camada.

Com o Javascript é possível definir se os elementos serão arrastados, dimensionados, rotacionados, reformatados etc.

O Javascript controla tudo isso manipulando as características dos elementos pelo CSS.

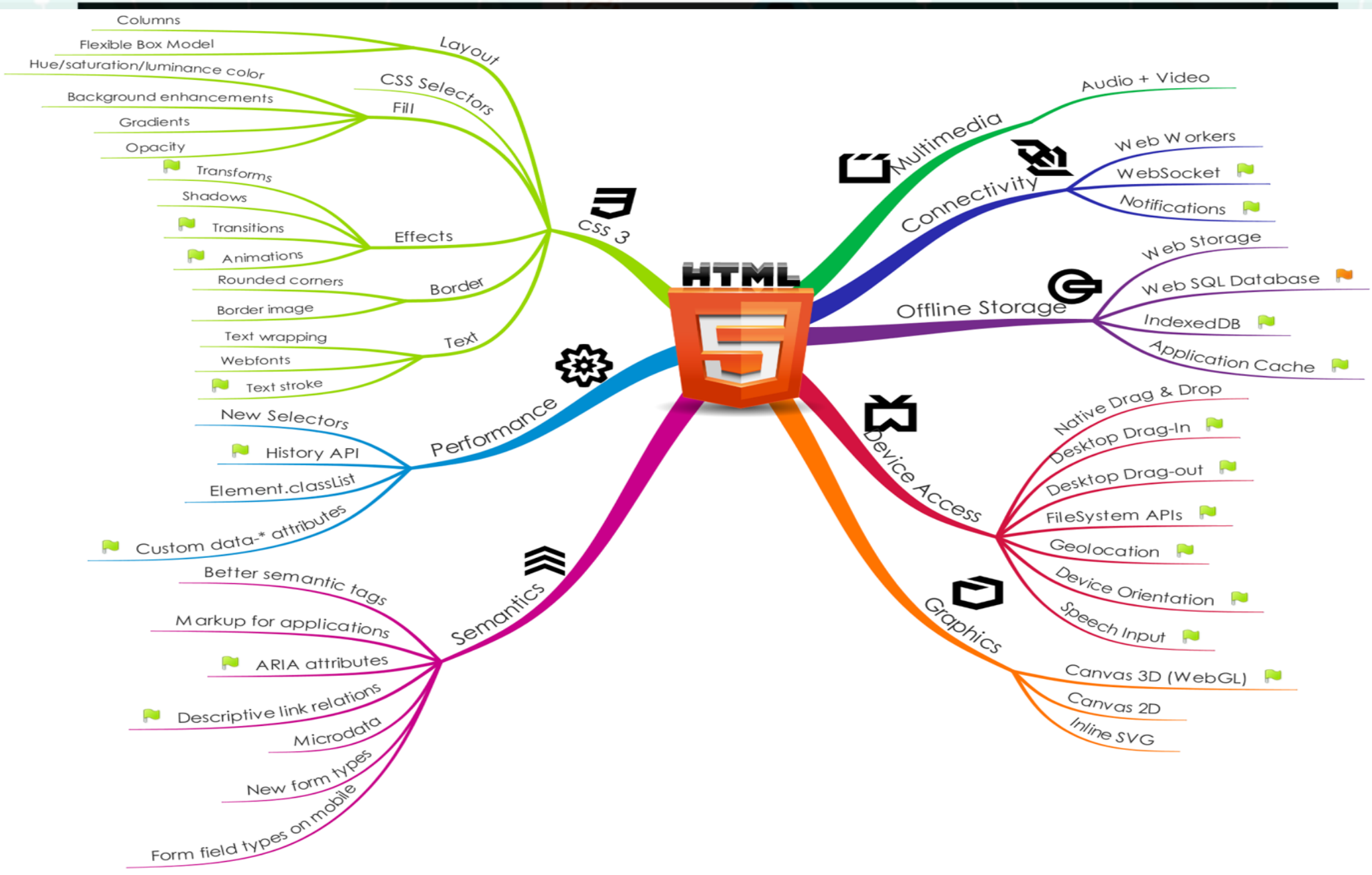
Resumindo: o Javascript controla os valores definidos pelo CSS e manipula estas propriedades.

Nessa camada você pode usar o JavaScript puro ou utilizar um framework como o Jquery.

O CSS também tem um papel nessa camada. Com o CSS3 é possível controlar comportamentos simples dos elementos, começando com animações e transições.

JavaScript

HTML 5


HTML
CSS 3
Layout
Columns
Flexible Box Model
CSS Selectors
Hue/saturation/luminance color
Fill
Background enhancements
Gradients
Opacity
Effects
Transforms
Shadows
Transitions
Animations
Border
Rounded corners
Border image
Text
Text wrapping
Webfonts
Text stroke
Performance
New Selectors
History API
Element.classList
Custom data-* attributes
Semantics
Better semantic tags
Markup for applications
ARIA attributes
Descriptive link relations
Microdata
New form types
Form field types on mobile
Multimedia
Audio + Video
Connectivity
Web Workers
WebSocket
Notifications
Offline Storage
Web Storage
Web SQL Database
IndexedDB
Application Cache
Device Access
Native Drag & Drop
Desktop Drag-In
Desktop Drag-out
FileSystem APIs
Geolocation
Device Orientation
Speech Input
Graphics
Canvas 3D (WebGL)
Canvas 2D
Inline SVG

HTML 5

Visão Geral

De acordo com o W3C a Web é baseada em 3 pilares:

- Um esquema de nomes para localização de fontes de informação na Web, esse esquema chama-se URI.
- Um Protocolo de acesso para acessar estas fontes, hoje o HTTP.
- Uma linguagem de Hypertexto, para a fácil navegação entre as fontes de informação: o HTML.

HTML é uma abreviação de Hypertext Markup Language - Linguagem de Marcação de Hypertexto. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc) na Web.

O HTML é baseado no conceito de Hipertexto. Hipertexto são conjuntos de elementos – ou nós – ligados por conexões. Estes elementos podem ser palavras, imagens, vídeos, áudio, documentos etc. Estes elementos conectados formam uma grande rede de informação.

HTML 5

Visão Geral

Para distribuir informação de uma maneira global, é necessário haver uma linguagem que seja entendida universalmente por diversos meios de acesso. O HTML se propõe a ser esta linguagem

Desenvolvido originalmente por Tim Berners-Lee o HTML ganhou popularidade quando o Mosaic - browser desenvolvido por Marc Andreessen na década de 1990 - ganhou força. A partir daí, desenvolvedores e fabricantes de browsers utilizaram o HTML como base, compartilhando as mesmas convenções.

Entre 1993 e 1995, o HTML ganhou as versões HTML+, HTML2.0 e HTML3.0, onde foram propostas diversas mudanças para enriquecer as possibilidades da linguagem.

Em 1997, o grupo de trabalho do W3C responsável por manter o padrão do código, trabalhou na versão 3.2 da linguagem, fazendo com que ela fosse tratada como prática comum.

HTML 5

Visão Geral

Desde o começo o HTML foi criado para ser uma linguagem independente de plataformas, navegadores e outros meios de acesso. Interoperabilidade significa menos custo.

Você cria apenas um código HTML e este código pode ser lido por diversos meios, ao invés de versões diferentes para diversos dispositivos.

O HTML deve ser entendido universalmente, dando a possibilidade para a reutilização dessa informação de acordo com as limitações de cada meio de acesso.

Enquanto o W3C focava suas atenções para a criação da segunda versão do XHTML, um grupo chamado Web Hypertext Application Technology Working Group (WHATWG) trabalhava em uma versão do HTML que trazia mais flexibilidade.

O **WHATWG** foi fundado por desenvolvedores de empresas como Mozilla, Apple e Opera em 2004. Eles não estavam felizes com o caminho que a Web tomava e nem com o rumo dado ao XHTML. Por isso, estas organizações se juntaram para escrever o que seria chamado hoje de HTML5.

HTML 5

Visão Geral

Por volta de 2006, o trabalho do WHATWG passou a ser conhecido pelo mundo e principalmente pelo W3C – que até então trabalhavam separadamente – que reconheceu todo o trabalho do grupo.

Em Outubro de 2006, Tim Berners-Lee anunciou que trabalharia juntamente com o WHATWG na produção do HTML5 em detrimento do XHTML 2. Contudo o XHTML continuaria sendo mantido paralelamente de acordo comas mudanças causadas no HTML. O grupo que estava cuidando especificamente do XHTML 2 foi descontinuado em 2009.

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final.

Ao contrário das versões anteriores, o HTML5 fornece ferramentas para o CSS e o Javascript fazerem seu trabalho da melhor maneira possível. O HTML5 permite, por meio de suas APIs, a manipulação das características destes elementos, de forma que o website ou a aplicação continue leve e funcional.

HTML 5

Visão Geral

O HTML5 também cria novas tags e modifica a função de outras. As versões antigas do HTML não continham um padrão universal para a criação de seções comuns e específicas como rodapé, cabeçalho, sidebar, menus e etc.

Não havia um padrão de nomenclatura de IDs, Classes ou tags. Não havia um método de capturar de maneira automática as informações localizadas nos rodapés dos websites. Há outros elementos e atributos onde sua função e significado foram modificados e que agora podem ser reutilizados de forma mais eficaz.

O HTML5 modifica a forma de como escrevemos código e organizamos a informação na página.

Seria mais semântica com menos código. Seria mais interatividade sem a necessidade de instalação de plugins e perda de performance.

É a criação de código interoperável, pronto para futuros dispositivos e que facilita a reutilização da informação de diversas formas.

Visão Geral

A estrutura básica do HTML5 continua sendo a mesma das versões anteriores da linguagem, há apenas uma exceção na escrita do Doctype. Segue abaixo como a estrutura básica pode ser seguida:

```
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>
```

Visão Geral – Doctype

O Doctype deve ser a primeira linha de código do documento antes da tag HTML.

<!DOCTYPE html!>

O Doctype indica para o navegador e para outros meios qual a especificação de código utilizar.

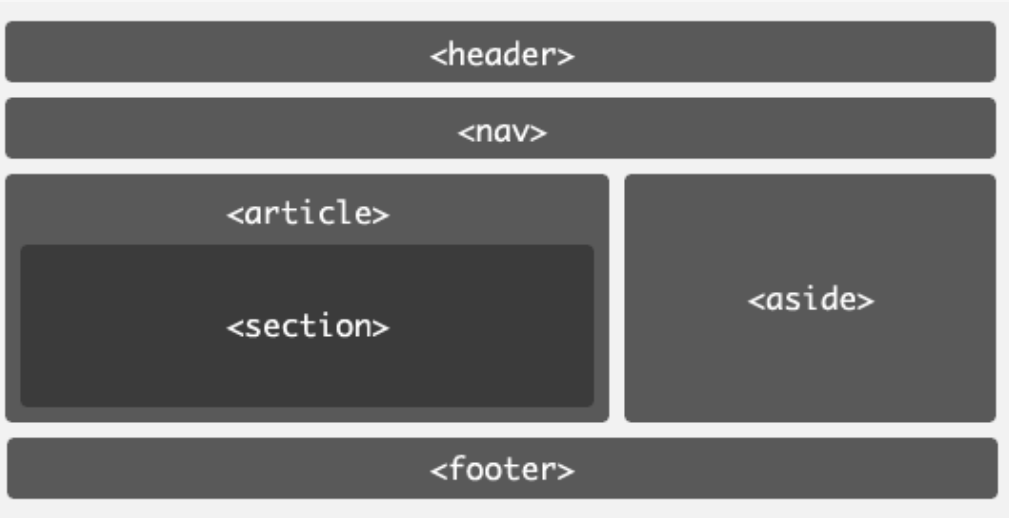
Em versões anteriores, era necessário referenciar o DTD diretamente no código do Doctype. Com o HTML5, a referência por qual DTD utilizar é responsabilidade do navegador.

O Doctype não é uma tag do HTML, mas uma instrução para que o navegador tenha informações sobre qual versão de código a marcação foi escrita.

Visão Geral – Elementos para o Conteúdo

O HTML5 introduz todo um conjunto de elementos que tornam muito fácil estruturar uma página. Muitas páginas HTML4 incluíam várias estruturas comuns, como cabeçalhos, rodapés e colunas. Ainda é comum criar várias tags 'div', dando a cada uma delas um id ou um class descritivo.

O uso dos elementos div é largamente utilizado porque falta ao HTML4 a semântica necessária para descrever essas partes de forma mais especifica. O HTML5 corrige esse problema ao introduzir novos elementos para representar cada uma dessas seções diferentes.



Veremos nas próximas páginas vários exemplos de código. Será dada a mesma orientação fornecida no Livro 01 para testar o código. Acompanhe a sugestão na página seguinte.

Testando o Código – w3schools

Veremos nas próximas páginas vários exemplos de código. Você deve praticar para o melhor aprendizado.

E como você vai praticar? Você vai testar o código no site w3schools.com.

Assim que você acessa o site, pode observar que existe um quadro com um exemplo em HTML e um botão "Try it Yourself".

Quando clicar no botão, você verá que é possível alterar o código e observar o resultado em HTML. É essa ferramenta que você utilizará durante a leitura para realizar testes e aprender HTML.

Após clicar no botão "Try it Yourself", o editor de HTML do w3schools será exibido, conforme imagem a seguir. Para ver o resultado do texto editado, basta clicar no botão "See Result".

Visão Geral – Elementos para o Conteúdo

O elemento 'header' representa o cabeçalho de uma seção. Cabeçalhos podem conter mais do que apenas o titulo da seção – por exemplo, seria razoável que o cabeçalho incluísse subtítulos, históricos de versões ou assinaturas.

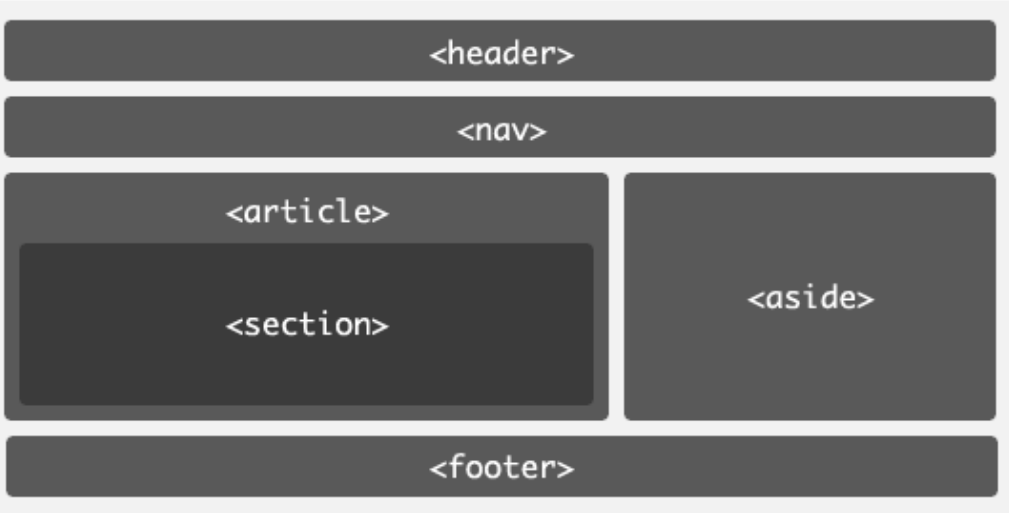
```
<header>
<h1>Visão Geral do HTML5</h1>
<p class="byline">By Albert Eije</p>
</header>
<header>
<h1>Teste H1</h1>
<h2>Teste H2</h2>
</header>
```

O elemento 'footer' representa o rodapé da seção para o qual ele se aplica. Um rodapé tipicamente contém informações sobre sua seção como quem escreveu, links para documentos relacionados, dados de copyright, e assim por diante.

```
<footer>© 2016 Albert Eije</footer>
```

O elemento 'nav' representa uma seção de links de navegação. É adequado tanto para navegação do site quanto para um índice.

```
<nav>
<ul>
<li><a href="/">Principal</a></li>
<li><a href="#">Produtos</a></li>
<li><a
href="/services">Serviços</a></li>
<li><a href="/about">Sobre</a></li>
</ul>
</nav>
```



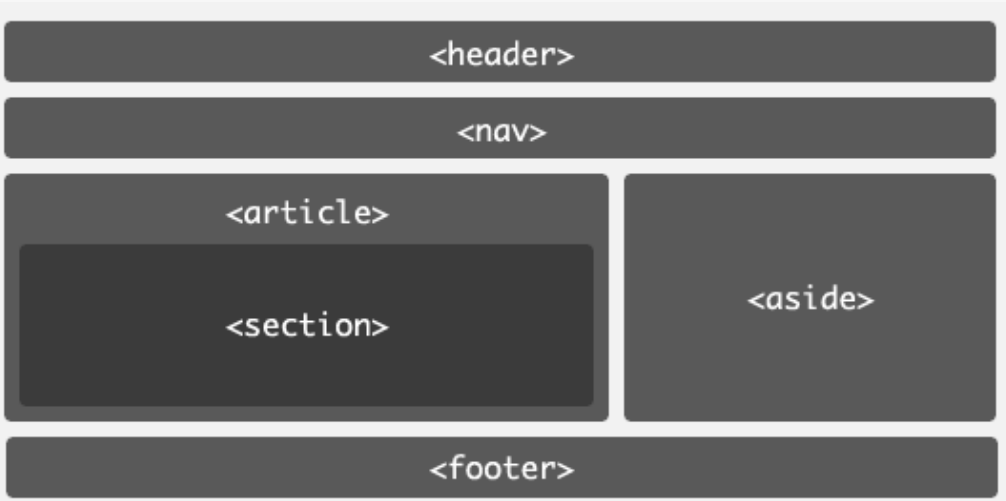
Visão Geral – Elementos para o Conteúdo

O elemento 'aside' é para conteúdos que sejam tangencialmente relacionados ao conteúdo em torno dele. É útil para definição de barras laterais.

```
<aside>
<h1>Arquivos</h1>
<ul>
<li><a href="#">Setembro 2016</a></li>
<li><a href="#">Julho 2016</a></li>
<li><a href="#">Agosto 2016</a></li>
</ul>
</aside>
```

O elemento 'section' representa uma seção genérica de um documento ou aplicação, como um capítulo, por exemplo.

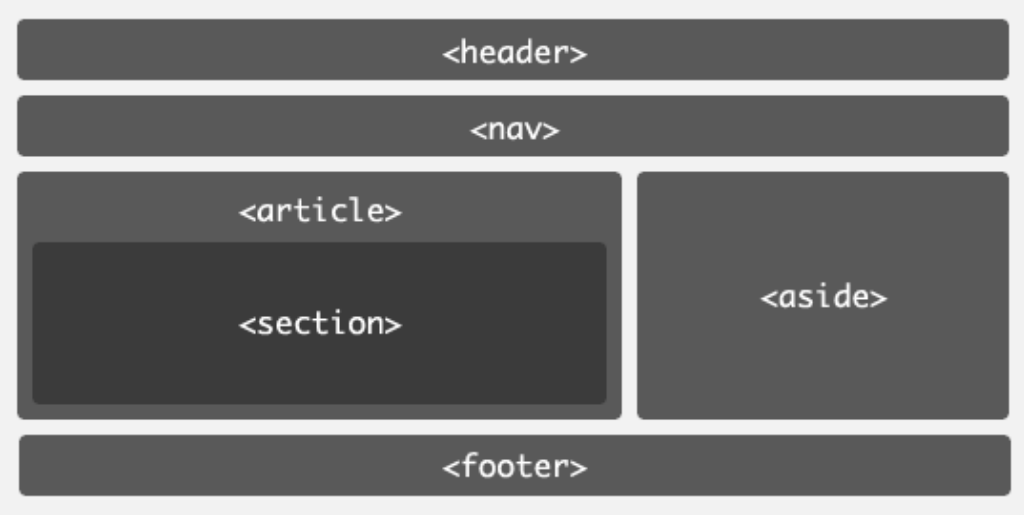
```
<section>
<h1>Capítulo 1: L3F0V</h1>
<p>E se um hacker agisse por conta
própria para resolver os problemas do
país? Qual seria a sua primeira
atitude?

O que seria da sociedade? E dos
políticos? Acompanhe a trajetória de
L3F0V nessa sua louca viagem.
...</p>
</section>
```



Visão Geral – Elementos para o Conteúdo

O elemento 'article' representa uma seção independente de um documento, página ou site. É adequado para conteúdo como notícias ou artigos de blogs, post em fóruns ou comentários individuais.

```
<article id="comment-2">
<header>
<h4><a href="#comment-2"
rel="bookmark">Comment #2</a>
by <a href="http://example.com/">Albert
Eije</a></h4>
<p><time datetime="2016-08-
29T13:58Z">Fevereiro 29th, 2016 at
13:58</time>
</header>
<p>Muito bom o artigo!</p>
</article>
```



Visão Geral – Categorias

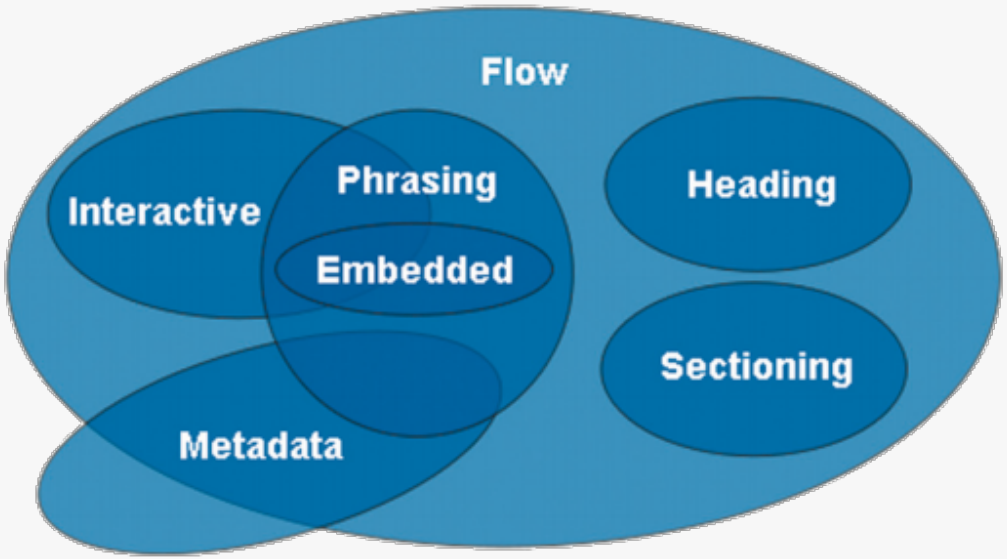
Cada elemento no HTML pode ou não fazer parte de um grupo de elementos com características similares. As categorias serão listadas a seguir.

Manterei os nomes das categorias em inglês para que haja um melhor entendimento:

- **Metadata content**
- **Flow content**
- **Sectioning content**
- **Heading content**
- **Phrasing content**
- **Embedded content**
- **Interactive content**

Na imagem ao lado observamos como as categorias estão relacionadas de acordo com o WHATWG.

Cada categoria possui seu conjunto de tags. Clique em cada um dos links para acessar a página do W3C na posição específica da categoria para observar quais tags a compõe.



Visão Geral – Formulários

O HTML5 introduz vários atributos novos, tipos de entrada e outros elementos para seus formulários.

A seção de formulários do HTML5 foi originalmente uma especificação chamada Web Forms 2.0 que adicionou novos tipos de controle para formulários.

Uma das melhores coisas sobre os formulários HTML5 é que você pode usar todos esses novos tipos de entrada e atributos de imediato.

Eles fazem coisas interessantes e úteis nos navegadores modernos que os suportam – e são ignorados sem problemas em navegadores que não os suportam. Isso se deve aos princípios de projeto do HTML5.

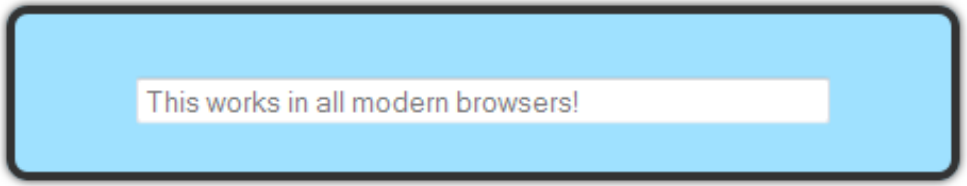
Veremos agora alguns dos novos atributos para observar como a coisa evoluiu. É bom estudar a especificação com atenção para conhecer de perto todos os novos elementos do HTML5.

HTML 5

Visão Geral – Formulários | placeholder

O atributo placeholder permite que configuremos um texto alternativo, parecido como que faríamos com o atributo value no HTML4. Ele deve ser usado apenas para descrições curtas. Para qualquer texto mais longo deve-se usar o atributo title.

A diferença do HTML4 é que o texto apenas é exibido quando o campo estiver vazio e não estiver em foco. Uma vez que o campo receba foco *e você comece a digitação*, o texto desaparece.



Testando

```
<input type="text" name="user-name" id="user-name" placeholder="Texto que vai aparecer para o usuário">
```

Visão Geral – Formulários | datalist

O elemento datalist é novo no HTML5 e representa uma lista pré-definida de opções para controles de formulário. Ele funciona de forma similar às caixas de pesquisa dos navegadores que autocompletam a informação que está sendo digitada.

Result:

Duplo Clique ou Digite:

- Aba
- Baba
- Cabra

```
<label>Duplo Clique ou Digite:
<datalist id="testes">
 <option value="Aba">Aba</option>
 <option value="Baba">Baba</option>
 <option value="Cabra">Cabra</option>
  <!-- ... -->
</datalist>
<input type="text" name="teste"
list="testes">
</label>
```

Visão Geral – Armazenamento Local

O armazenamento local do HTML5 torna possível armazenar valores no navegador que podem ser recuperados mesmo após o encerramento da sessão, como os cookies. O armazenamento local pode também permitir a transmissão de eventos entre janelas do navegador, um recurso muito útil.

O armazenamento local oferece um depósito simples de pares chave – valor, como uma tabela hash ou um dicionário. O armazenamento local vem em duas versões:

- Armazenamento de sessão (sessionStorage)
- Armazenamento local (localStorage)

O armazenamento de sessão está disponível dentro da mesma janela do navegador enquanto ela estiver aberta. O armazenamento local está disponível no navegador para todas as janelas com a mesma origem (domínio).

Visão Geral – Armazenamento Local

Os objetos sessionStorage e localStorage são acessados da mesma forma. Apenas seus tempos de vida e visibilidade dos dados que são diferentes.

Você pode armazenar apenas Strings nas propriedades do armazenamento de sessão e local. Nada mais.

Se precisa armazenar objetos JavaScript, terá que convertê-los em uma String JSON primeiro.

Os objetos de armazenamento disparam eventos que sua aplicação pode querer tratar. Um evento de armazenamento é disparado quando você insere, atualiza ou remove uma propriedade de seu armazenamento local ou da sessão.

Próximos Passos para o Aprendizado

Até aqui demos uma "pincelada" no HTML5. Não trataremos de todos os elementos do HTML5 no livro, pois seria enfadonho e contraproducente. Chegou a hora da ação, do estudo online.

O primeiro passo você já deveria ter dado. Caso não tenha estudado ainda o HTML, pare a leitura deste livro e leia o Livro 01. Faça os exercícios lá descritos. Realize os testes. Aprender HTML é pré-requisito obrigatório para aprender HTML5.

Tenha em mente ainda o seguinte:

O desenvolvimento no lado do cliente é baseado em 3 camadas principais: informação (HTML), formatação (CSS) e comportamento (JavaScript).

Ou seja, mesmo depois de estudar HTML e HTML5 você ainda não estará preparado para desenvolver aplicações web. Sendo assim, encarre esses livros e tais conhecimentos como uma parte pertencente a um todo, que será completado quando você estudar CSS, JavaScript e Ajax.

Próximos Passos para o Aprendizado | W3C

Após estudar o conteúdo HTML do primeiro livro, você irá ler o material disponibilizado pelo W3C no site brasileiro no seguinte link:

Material W3C – HTML5

Estude o material acima com atenção. Teste todo e qualquer código no editor do w3schools. Acostume-se com os novos elementos do HTML5 e com a maneira de implementar o código. Será essencial aprender isso para entender o material que virá sobre CSS e JavaScript.

Após estudar o material do W3C, é importante que você tenha um guia de referência.

Seguem dois links para você continuar os estudos. Entre em cada tag. Comece por aquelas que mais lhe chamam atenção. Teste os códigos no editor do w3schools.

Lista de Elementos do HTML5

HTML Element Reference

CSS

CSS

Introdução

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. O seu principal benefício é a separação entre o formato e o conteúdo de um documento.

Em vez de colocar a formatação dentro do documento, o desenvolvedor cria um link (ligação) para uma página que contém os estilos, procedendo de forma idêntica para todas as páginas de um portal.

Com a variação de atualizações dos navegadores, o suporte ao CSS pode variar. O Internet Explorer 6, por exemplo, tem suporte total a CSS1 e praticamente nulo a CSS2. Navegadores mais modernos como Google Chrome e Mozilla Firefox tem suporte maior, inclusive até a CSS3.

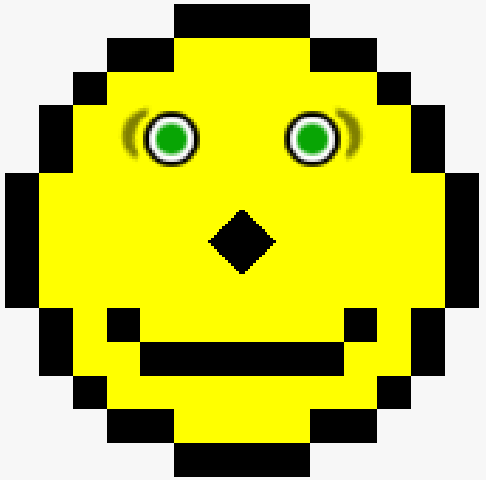
A interpretação dos navegadores pode ser avaliada com o teste Acid2, que se tornou uma forma base de revelar quão eficiente é o suporte do CSS.

CSS

Acid2

Acid2 é o nome dado a uma página de teste de compatibilidade com padrões web (Web Standards) W3C com navegadores, publicada e promovida pelo Web Standards Project para expor problemas de compatibilidade com os padrões web existentes em alguns navegadores. O objetivo do teste é que o navegador testado exiba exatamente um rosto amarelo junto das palavras “Hello World”.

Clique na imagem ao lado para acessar a página do Acid2 e observe o resultado no seu navegador.



CSS

Sintaxe

O CSS tem uma sintaxe simples e utiliza uma série de palavras em inglês para especificar os nomes de diferentes estilos de propriedade de uma página. Uma folha de estilo consiste de uma lista de regras. Cada regra ou conjunto de regras consiste de um ou mais seletores e um bloco de declaração. Uma declaração de bloco é composta por uma lista de declarações entre chaves. Cada declaração em si é uma propriedade, dois pontos (:), um valor, então um ponto e vírgula (;).

```
seletor {
 propriedade: valor;
}
```

Os seletores são usados para declarar a quais elementos de marcação um estilo se aplica. Os seletores podem ser aplicados a todos os elementos de um tipo específico, ou apenas aqueles elementos que correspondam a um determinado atributo; elementos podem ser combinados, dependendo de como eles são colocados em relação uns aos outros no código de marcação, ou como eles estão aninhados dentro do objeto de documento modelo.

CSS

CSS3

O CSS3 é a mais nova versão do CSS, onde se define estilos para páginas web com efeitos de transição, imagem, e outros, que dão um estilo novo às páginas Web 2.0 em todos os aspectos de design do layout.

A principal função do CSS3 é abolir as imagens de plano de fundo, bordas arredondadas, apresentar transições e efeitos para criar animações de vários tipos, como um simples relógio de ponteiros.

Isso se deve aos novos navegadores que estão chegando, com suporte à essa linguagem, como o Google Chrome, Opera, Internet Explorer 9, Safari, Firefox e Microsoft Edge.

Dessa forma, o CSS3 facilita o trabalho dos que desenvolvem para web e também dos usuários, pela variedade de transformações na apresentação de um site.

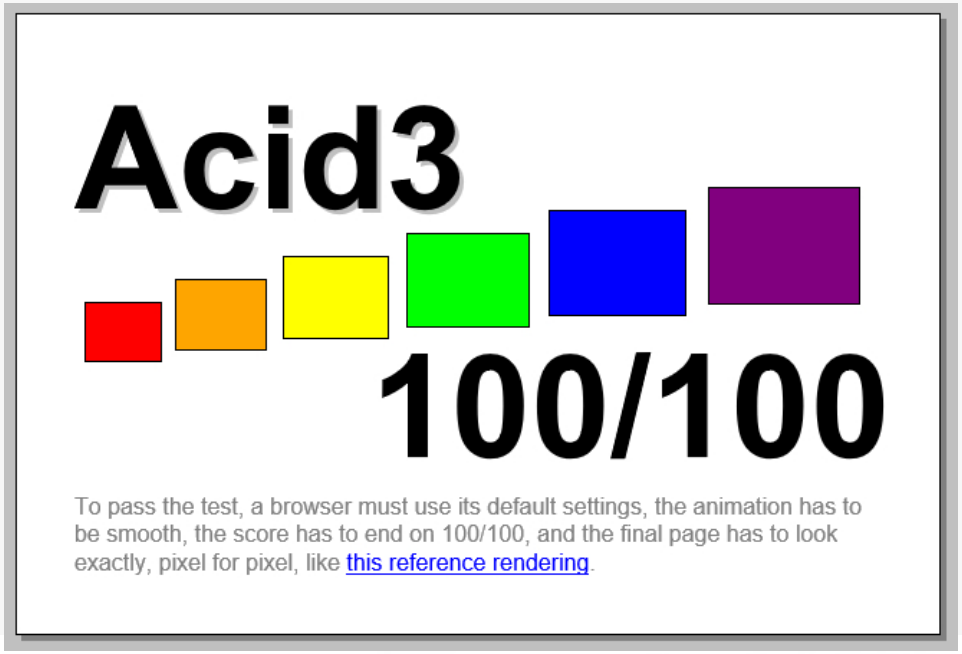
CSS

Acid3

Acid3 é o nome dado a uma suíte de testes de compatibilidade com padrões web (Web Standards) W3C com navegadores, com cem diferentes tipos de testes, sendo o sucessor do teste Acid2.

Esse teste tem um nível muito alto de exigência para os navegadores em comparação com o seu antecessor - Acid2. A sua escala vai de 1 a 100.

A imagem ao lado mostra o teste feito no Microsoft Edge. Clique na imagem para fazer o teste no seu navegador.



CSS

Seletores

O CSS define regras que fazem as definições de estilo casarem com um elemento ou grupo de elementos. O documento HTML pode conter um bloco de css em um elemento estilo (style) ou usando o elemento link apontando para um arquivo externo que contenha o bloco CSS (arquivo_qualquer.css). Para usar o CSS foi criado o atributo class que todo elemento pode conter.

As regras de casamento para o css são chamadas de seletores, uma definição de estilo pode ser casada com um seletor ou um grupo de seletores separados por vírgula, um seletor pode casar um elemento de várias maneiras:

- Elemento do tipo: element_name {style definition;}
- Elemento do tipo com a classe: element_name.class_name {style definition;}
- Todos os elementos com a classe: .class_name {style definition;}
- O elemento com o id: #id_of_element {style definition;}
- Casamento de um grupo: element_name_01, element_name_02, .class_name {style definition;}

CSS

Seletores

Exemplos:

```
p {text-align: right; color: #BC2;}
p.minhaclasse01 { color:#ADC; }
.minhaclasse02 { color:#CBD; }
#iddomeuelemento { color:#ABD; }
p.minhaclasse03 .minhaclasse04 { color:#ABD; }


/* comentário em css, semelhante aos da linguagem c */
body
{
   font-family: Arial, Verdana, sans-serif;
   background-color: #FFF;
   margin: 5px 10px;
}
```

Seletores

O código anterior define a fonte padrão como Arial. Caso ela não exista, será substituída pela Verdana. Caso a Verdana não exista, define qualquer fonte sans-serif. Define também a cor de fundo do corpo da página. Sua necessidade adveio do fato de o HTML (Hyper Text Markup Language) aos poucos ter deixado de ser usado apenas para criação de conteúdo na web, e portanto havia uma mistura de formatação e conteúdo textual dentro do código de uma mesma página. Contudo, na criação de um grande portal, fica quase impossível manter uma identidade visual, bem como a produtividade do desenvolvedor. É nesse ponto que entra o CSS.

As especificações do CSS podem ser obtidas no site da W3C "World Wide Web Consortium", um consórcio de diversas empresas que buscam estabelecer padrões para a Internet. É importante notar que nenhum navegador suporta igualmente as definições do CSS. Desta forma, o web designer deve sempre testar suas folhas de estilo em navegadores de vários fabricantes, e preferencialmente em mais de uma versão, para se certificar de que o que foi codificado realmente seja apresentado da forma desejada.

Como Estudar CSS?

A melhor forma de estudar CSS é praticando. Um dos melhores e mais conhecidos sites brasileiros sobre CSS é o MAUJOR.

Clique na imagem para explorar o site. Tem muita coisa. Você pode passar vários dias explorando esse site por conta da quantidade de material.

Testando o Código – w3schools

E onde você vai testar o código? Assim como foi feito com o HTML e como faremos com o XML, você vai testar o seu código CSS no site w3schools.com.

O site tem mais de 300 exemplos prontos para teste.

Para testar seu próprio código, basta digitar ou colar no quadro branco e clicar no botão "Try it yourself".

Divirta-se!

CSS

Guia de Referência W3C

Clique na imagem para acessar um PDF com o Guia de Referência CSS, uma publicação do W3C Brasil.

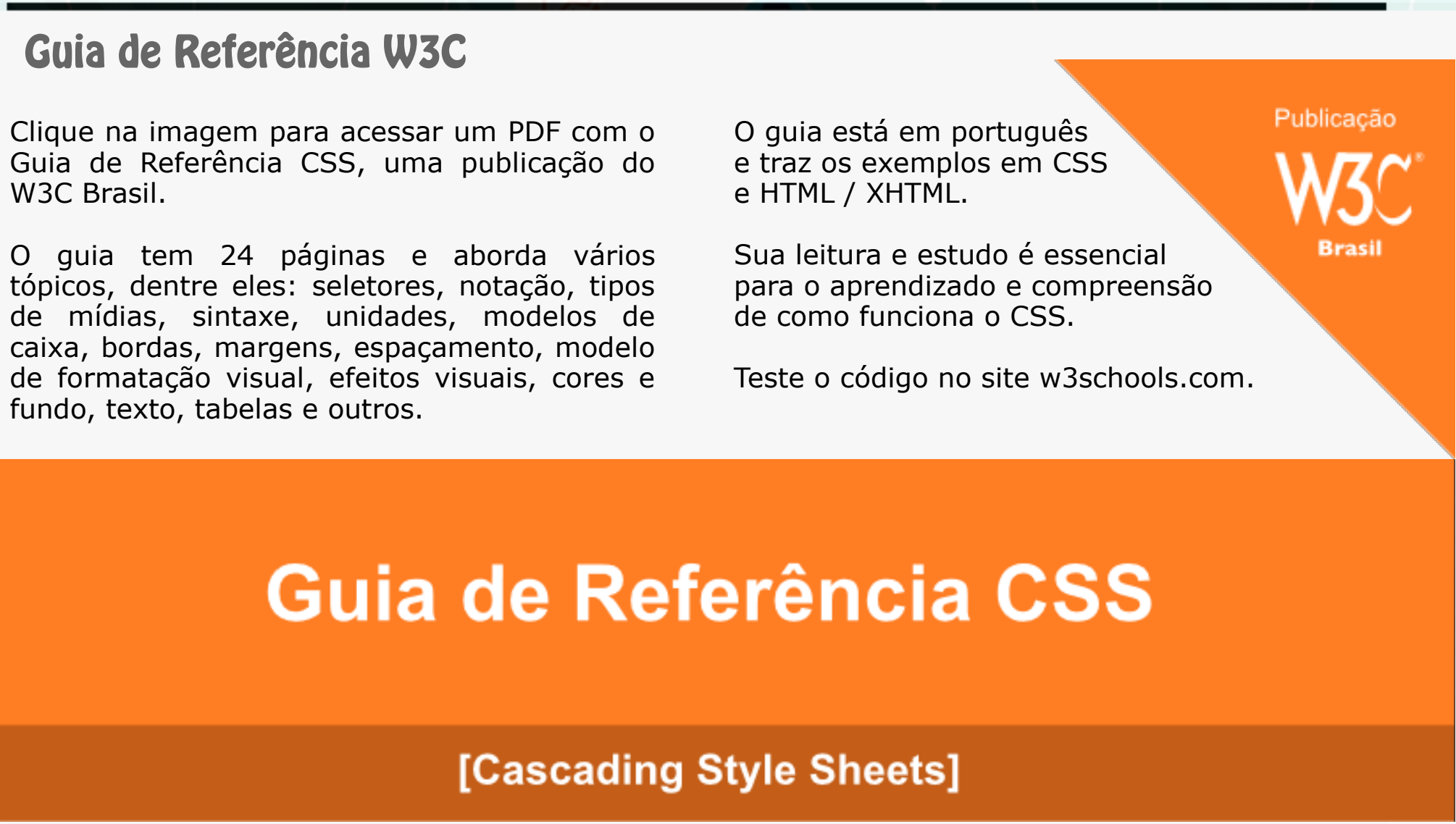
O guia tem 24 páginas e aborda vários tópicos, dentre eles: seletores, notação, tipos de mídias, sintaxe, unidades, modelos de caixa, bordas, margens, espaçamento, modelo de formatação visual, efeitos visuais, cores e fundo, texto, tabelas e outros.

O guia está em português e traz os exemplos em CSS e HTML / XHTML.

Sua leitura e estudo é essencial para o aprendizado e compreensão de como funciona o CSS.

Teste o código no site w3schools.com.

Publicação W3C Brasil



CSS

Visão Geral

O CSS formata a informação entregue pelo HTML. Essa informação pode ser qualquer coisa: imagem, texto, vídeo, áudio ou qualquer outro elemento criado. Tenha em mente que o CSS *formata a informação*. Essa formatação na maioria das vezes é visual, mas não necessariamente. É possível manipular o áudio, por exemplo, caso esteja utilizando o CSS Aural. O CSS prepara essa informação para que ela seja consumida da melhor maneira possível.

O HTML5 trouxe poucas novidades para os desenvolvedores do lado cliente. Basicamente foram criadas novas tags, o significado de algumas foi modificado e outras tags foram descontinuadas. As novidades interessantes mesmo ficaram por conta do Javascript. As APIs que o HTML5 disponibilizou são, sem dúvida, uma das características mais aguardadas por todos estes desenvolvedores. Já o CSS3 trouxe mudanças drásticas para a manipulação visual dos elementos do HTML.

CSS

Visão Geral

Com o CSS padrão podemos formatar algumas características básicas: cores, background, características de fontes, margens, paddings, posições etc. Com as atualizações do CSS3, podemos controlar diversas outras coisas:

- Selecionar primeiro e último elemento, além de elementos pares ou ímpares.
- Selecionarmos elementos específicos de um determinado grupo de elementos.
- Gradiente em textos e elementos.
- Bordas arredondadas.
- Sombras em texto e elementos.
- Manipulação de opacidade.
- Controle de rotação.
- Controle de perspectiva.
- Animação.
- Estruturação independente da posição no código HTML.

Visão Geral - Seletores

De volta aos seletores. Os seletores são a alma do CSS e você precisa dominá-los.

É com os seletores que você irá escolher um determinado elemento dentre todos os outros elementos do site para formatá-lo.

Boa parte da inteligência do CSS está em saber utilizar os seletores de uma maneira eficaz, escalável e inteligente.

O seletor representa uma estrutura. Essa estrutura é usada como uma condição para determinar quais elementos de um grupo de elementos serão formatados.

Seletores encadeados e seletores agrupados são a base do CSS.

Visão Geral - Seletores

Seletores Encadeados

```
div p strong a { color: red; }
```

Este seletor formata o link (a), que está dentro de um strong, que está dentro de P e que por sua vez está dentro de um DIV.

Seletores Agrupados

```
strong, em, span {
 color: red;
}
```

Você agrupa elementos separados por vírgula para que herdem a mesma formatação.

Visão Geral - Seletores

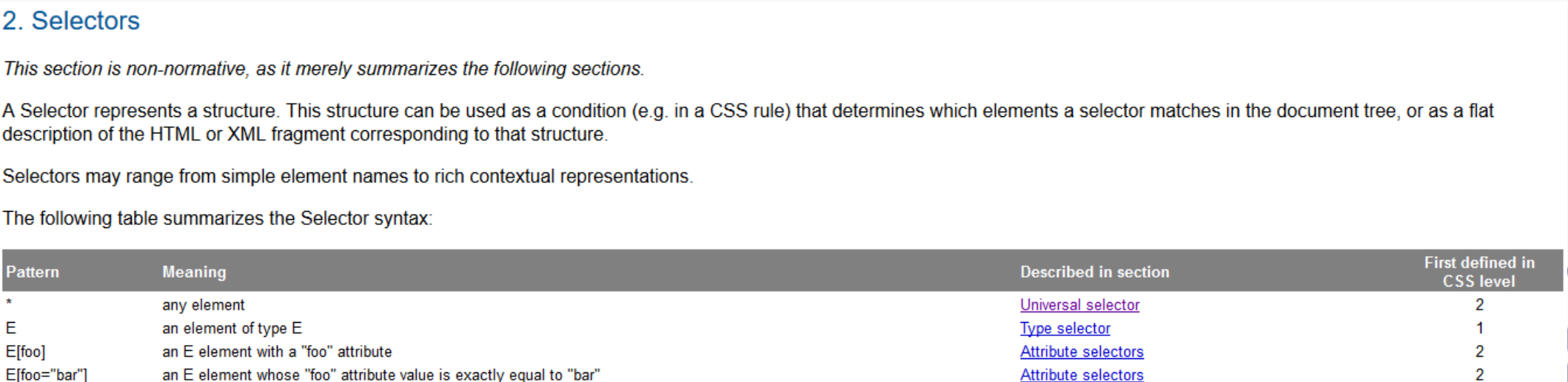
Seletores Complexos

Os seletores complexos trabalham com elementos que precisariam de scripts em Javascript para marcá-los com uma CLASS ou um ID para serem formatados. Com os seletores complexos você consegue formatar elementos que antes eram inalcançáveis.

Imagine que você tenha um título (h1) seguido de um parágrafo (p). Você precisa selecionar todos os parágrafos que vem depois de um título h1. Com os seletores complexos você fará assim

```
h1 + p {
color:red;
}
```

Clique na imagem abaixo para acessar uma lista de seletores complexos no site W3C.



2. Selectors

This section is non-normative, as it merely summarizes the following sections.

A Selector represents a structure. This structure can be used as a condition (e.g. in a CSS rule) that determines which elements a selector matches in the document tree, or as a flat description of the HTML or XML fragment corresponding to that structure.

Selectors may range from simple element names to rich contextual representations.

The following table summarizes the Selector syntax:

Pattern	Meaning	Described in section	First defined in CSS level
*	any element	Universal selector	2
E	an element of type E	Type selector	1
E[foo]	an E element with a "foo" attribute	Attribute selectors	2
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"	Attribute selectors	2

CSS

Visão Geral - Gradiente

Uma das características mais legais do CSS3 é a criação de gradientes. Todos os navegadores mais recentes aceitam essa característica. Nada melhor do que um exemplo para compreender. Clique na imagem abaixo para ir direto para um exemplo no site w3schools.com.

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 200px;
    background: red; /* For browsers that do not support gradients */
    background: -webkit-linear-gradient(red, yellow); /* For Safari 5.1 to 6.0 */
    background: -o-linear-gradient(red, yellow); /* For Opera 11.1 to 12.0 */
    background: -moz-linear-gradient(red, yellow); /* For Firefox 3.6 to 15 */
    background: linear-gradient(red, yellow); /* Standard syntax (must be last) */
}
</style>
</head>
<body>

<h3>Linear Gradient - Top to Bottom</h3>
<p>This linear gradient starts at the top. It starts red, transitioning to yellow:
</p>

<div id="grad1"></div>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support
gradients.</p>

</body>
</html>
```

Result:

Linear Gradient - Top to Bottom

This linear gradient starts at the top. It starts red, transitioning to yellow:

Note: Internet Explorer 9 and earlier versions do not support gradients.

CSS

Visão Geral - Columns

Com o controle de colunas no CSS, podemos definir colunas de texto de forma automática. A propriedade column-count define a quantidade de colunas terá o bloco de textos. Com a propriedade column-width definimos a largura destas colunas. A propriedade column-gap cria um espaço entre as colunas. Clique na imagem abaixo para ver a descrição detalhada no site w3schools.com e realizar testes com os exemplos.

Edit This Code:

See Result »

Result:

```
<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
</style>
</head>
<body>

<p><b>Note:</b> Internet Explorer 9, and earlier versions, does not support the
column-count property.</p>

<div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad
minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in
vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis
at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril
delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta
nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer
possim assum.
</div>

</body>
</html>
```

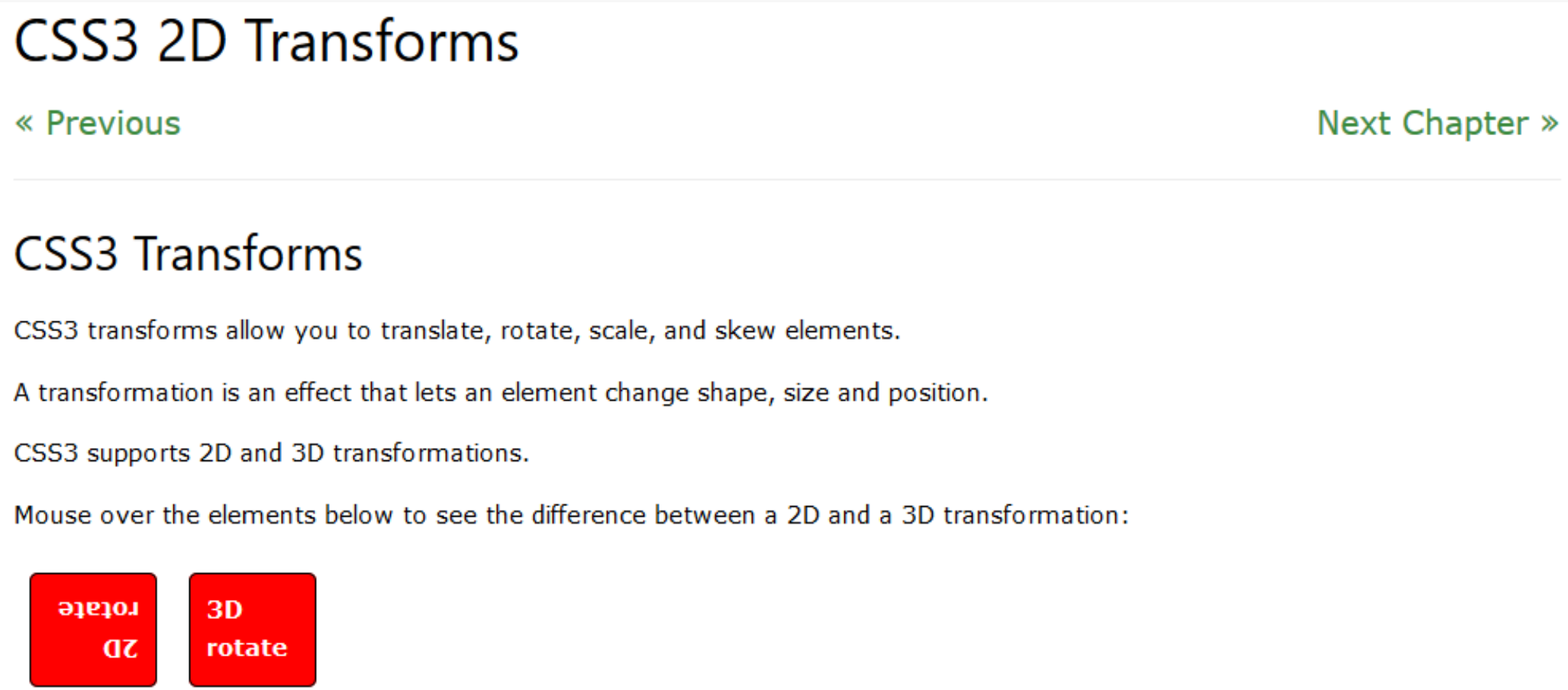
Note: Internet Explorer 9, and earlier versions, does not support the column-count property.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

CSS

Visão Geral - Transform

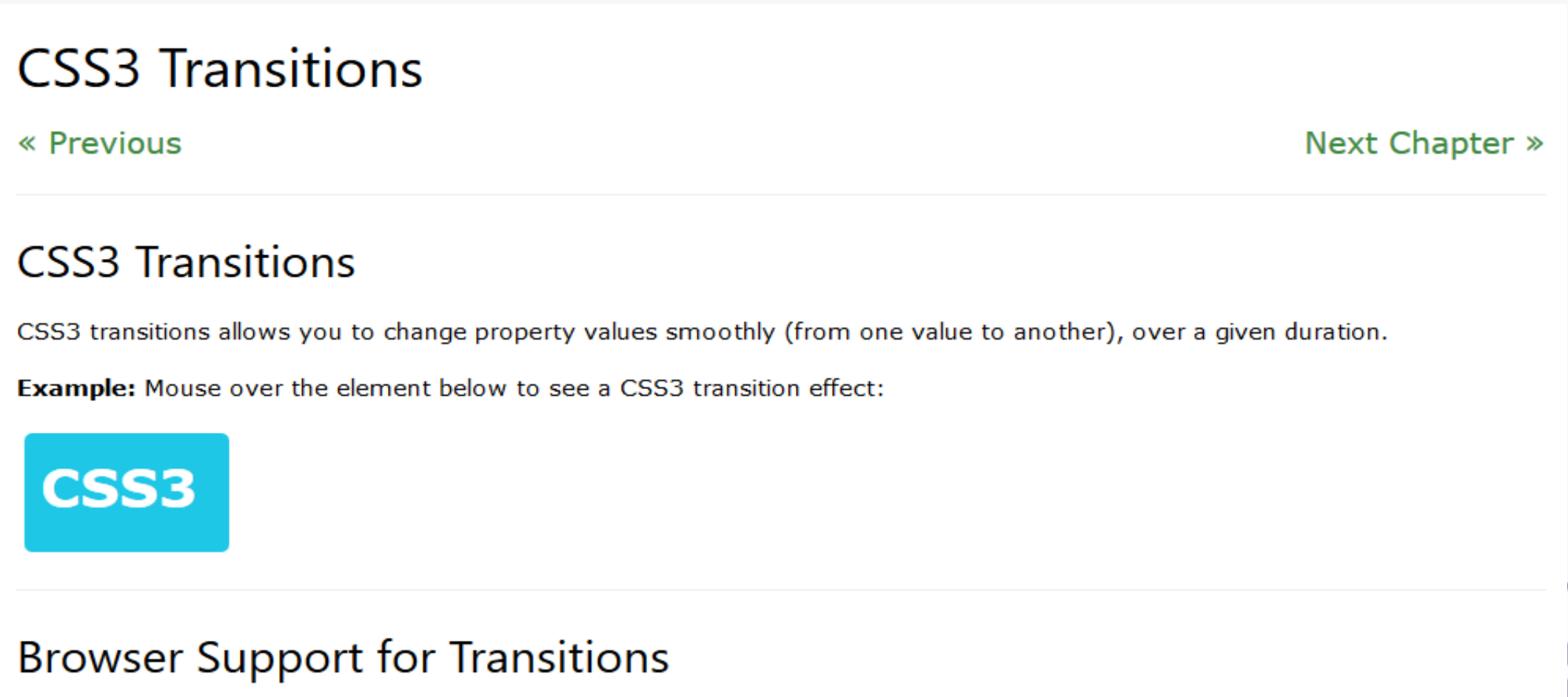
A propriedade transform manipula a forma com que o elemento aparecerá na tela. Você poderá manipular sua perspectiva, escala e ângulos. Uma transformação é especificada utilizando a propriedade transform. Clique na imagem abaixo para ver a descrição detalhada no site w3schools.com e realizar testes com os exemplos.



CSS

Visão Geral - Transições

As transições permitem alterar os valores das propriedades suavemente, durante um determinado período. Clique na imagem abaixo para ver a descrição detalhada no site w3schools.com e realizar testes com os exemplos.



CSS

Visão Geral - Animações

As animações permitem manipular e animar a maioria dos elementos HTML sem necessitar de JavaScript ou Flash. Clique na imagem abaixo para ver a descrição detalhada no site w3schools.com e realizar testes com os exemplos.

CSS3 Animations

« Previous Next Chapter »

CSS3 Animations

CSS3 animations allows animation of most HTML elements without using JavaScript or Flash!

CSS3

Browser Support for Animations

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by -webkit-, -moz-, or -o- specify the first version that worked with a prefix.

Property						

Visão Geral - RGBA

Quando desenvolvemos para web, normalmente com cores na forma de hexadecimal. É a forma mais comum e mais utilizada. Ainda assim, existem outros formatos menos comuns que funcionam sem problemas. Um deles é o RGB.

O RGB são 3 conjuntos de números que começam em 0 e vão até 255 (0% até 100%), onde o primeiro bloco define a quantidade de vermelho (Red), o segundo bloco a quantidade de verde (Green) e o último bloco a quantidade de azul (Blue). A combinação destes números formam todas as cores que você possa imaginar.

```
p {
 background:rgb(255,255,0);
 padding:10px;
 font:13px verdana;
}
```

O exemplo anterior define a cor de fundo do elemento P para amarelo.

OK. Mas esse tópico trata de RGBA. O que significa o "A" nessa sigla? Bom, ocorre que se trabalharmos apenas com o RGB sempre teremos cores sólidas. Se você colocar um elemento branco na frente de um elemento vermelho, ele vai sobrepor o elemento e não veremos mais o elemento vermelho, apenas o branco. E se fosse possível aplicar uma "lente" no elemento branco, de modo que ele ficasse um pouco transparente e pudéssemos ver tanto o elemento branco (uma retícula do branco) e ainda assim o elemento vermelho ao fundo? Seria bom! E é pra isso que serve a letra "A" da sigla RGBA.

Visão Geral - RGBA

O CSS3 nos trouxe a possibilidade de modificar a opacidade dos elementos. Lembrando que quando modificamos a opacidade do elemento, tudo o que está contido nele também fia opaco e não apenas o background ou a cor dele.

O RGBA funciona da mesma forma que o RGB, ou seja, definindo uma cor para a propriedade. No caso do RGBA, além dos 3 canais RGB (Red, Green e Blue) há um quarto canal, A (Alpha) que controla a opacidade da cor. Nesse caso, podemos controlar a opacidade da cor do background sem afetar a opacidade dos outros elementos.

XML


XML
<div class="buttonb
<a href="Albums

<asp:ID="D
<ItemTemplate>
<table bord
<tr>
<td

XML

Introdução

Em meados da década de 1990, o World Wide Web Consortium (W3C) começou a trabalhar em uma linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML. O princípio do projeto era criar uma linguagem que pudesse ser lida por software, e integrar-se com as demais linguagens. Sua filosofia seria incorporada por vários princípios importantes:

- Separação entre conteúdo e formatação;
- Simplicidade e legibilidade, tanto para humanos quanto para computadores;
- Possibilidade de criação de TAGs ilimitadas;
- Criação de arquivos para validação da estrutura do arquivo XML (Os chamados DTDs - Document Type Definition);
- Interligação de bancos de dados distintos;
- Concentração na estrutura da informação, não na sua aparência;

O XML é um formato para a criação de documentos com dados organizados de forma hierárquica. Como exemplos desses tipos de documentos podemos citar: documentos de texto formatados, imagens vetoriais e bancos de dados.

XML

O que é XML?

É uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. É um subtipo do SGML (acrônimo de Standard Generalized Markup Language, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet.

A principal característica do XML é a de criar uma única infraestrutura para a criação de outras linguagens tais como:

- XHTML – é uma reformulação da linguagem de marcação HTML, baseada em XML. Combina as TAGs de marcação HTML com regras da XML. Este processo de padronização tem em vista a exibição de páginas Web em diversos dispositivos (televisão, palm, celular, etc). Sua intenção é melhorar a acessibilidade.
- RDF – é uma linguagem para representar informação na Internet.
- MathML – é uma aplicação do XML para representar símbolos e fórmulas matemáticas.

XML

O que é XML?

- SDMX – é uma iniciativa internacional com o objetivo de desenvolver e empregar processos eficientes para troca e compartilhamento de dados e metainformação estatística entre organizações internacionais e os seus países-membros.
- XBRL – padrão emergente baseado no XML para definir informação financeira. É dirigido por um consórcio internacional sem fins lucrativos (XBRL International Incorporated) de mais de 300 organizações, entre entidades reguladoras, agências governamentais e empresas de software.
- SVG – é a abreviatura de Scalable Vectorial Graphics que pode ser traduzido do inglês como gráficos vetoriais escaláveis. Trata-se de uma linguagem XML para descrever de forma vetorial desenhos e gráficos bidimensionais, quer de forma estática, quer dinâmica ou animada. Umas das principais características dos gráficos vetoriais, é que não perdem qualidade ao serem ampliados. A grande diferença entre o SVG e outros formatos vetoriais, é o fato de ser um formato aberto, não sendo propriedade de nenhuma empresa.

XML

O que é XML?

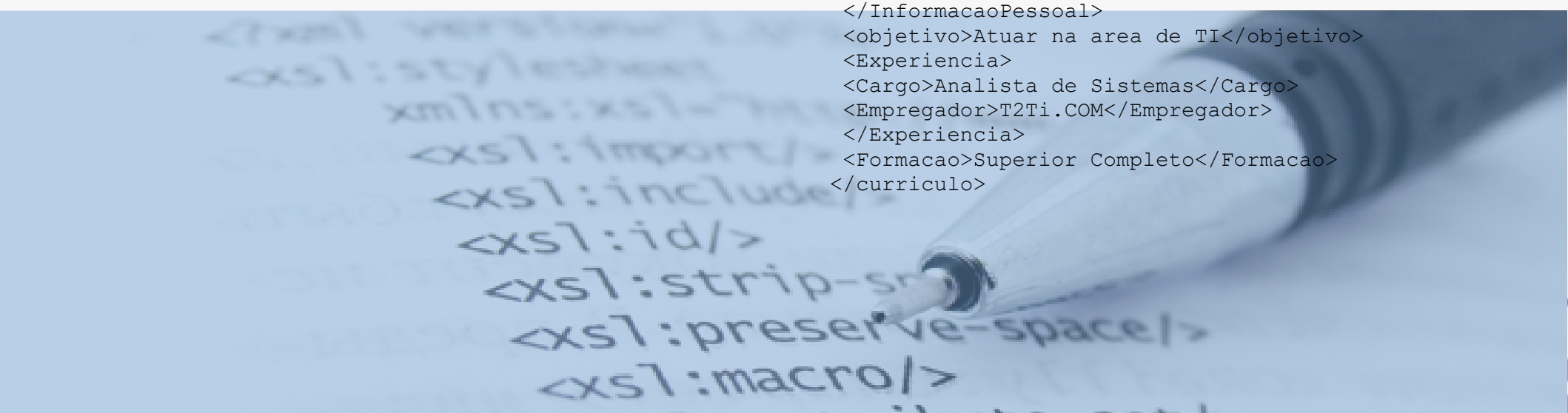
O XML não faz nada. Isso mesmo, não faz nada. Pode ser difícil de compreender no início, mas é simples: o XML foi criado para estruturar, armazenar e obter informações de transporte dos dados.

Assim como o HTML, o XML é um texto simples organizado em TAGs.

Vamos a um exemplo de XML:

Curriculum Vitae:

```
<?xml version="1.0" encoding="UTF-8"?>
<curriculo>
 <InformacaoPessoal>
 <DataNascimento>10-10-1980</DataNascimento>
      <Nomecompleto>Albert      Eije      Barreto
Mouta</Nomecompleto>
 <Contatos>
 <Residencia>
 <Rua>R. Maria Tomasia</Rua>
 <Numero>458</Numero>
 <Cidade>Sao Paulo</Cidade>
 <Pais>Brasil</Pais>
 </Residencia>
 <Telefone>8888-5555</Telefone>
 <Email>alberteije@gmail.com</Email>
 </Contatos>
 <Nacionalidade>Brasileira</Nacionalidade>
 <Sexo>M</Sexo>
 </InformacaoPessoal>
 <objetivo>Atuar na area de TI</objetivo>
 <Experiencia>
 <Cargo>Analista de Sistemas</Cargo>
 <Empregador>T2Ti.COM</Empregador>
 </Experiencia>
 <Formacao>Superior Completo</Formacao>
</curriculo>
```



Comparações entre HTML e XML

Vamos fazer uma analogia com o HTML:

HTML significa HyperText Markup Language, que traduzido quer dizer Linguagem de Marcação de Hipertexto.

XML significa eXtensible Markup Language, que traduzido quer dizer Linguagem de Marcação Extensível, ou seja, é uma extensão do HTML.

Na verdade o HTML e o XML são "parentes". Eles derivam da mesma linguagem, o SGML.

Ambos identificam elementos em uma página e ambos utilizam sintaxes similares.

Se você conhece HTML, será muito fácil compreender o XML.

A grande diferença entre HTML e XML é que o HTML descreve a estrutura e as ações em uma página enquanto o XML não descreve estrutura nem ações, mas sim o que cada trecho de dados é ou representa. Ou seja, o XML descreve o conteúdo do documento.

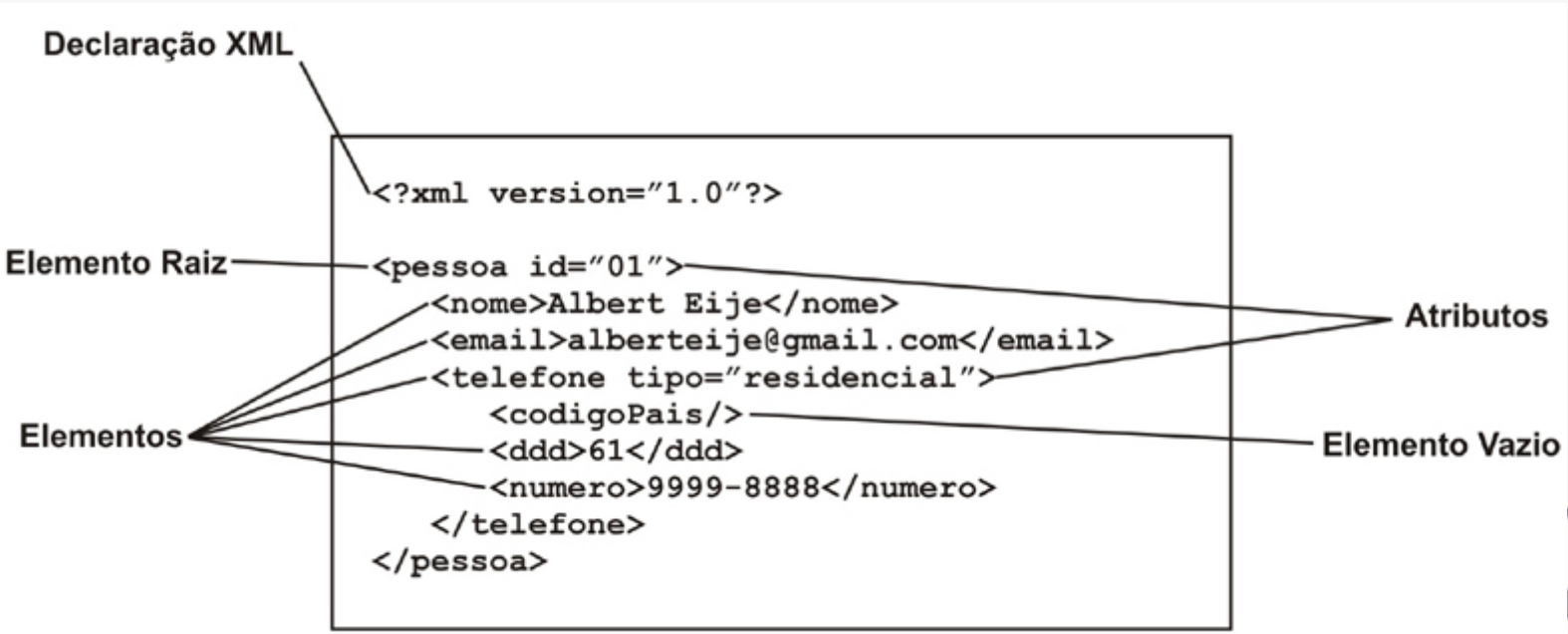
Comparações entre HTML e XML

Assim como o HTML, o XML também faz uso de TAGs e atributos:

- TAGs – palavras encapsuladas pelos sinais "<" e ">";
- Atributos – característica de determinada TAG (definidos da seguinte forma: name="value").

O HTML especifica cada sentido para as TAGs e atributos.

O XML usa as TAGs somente para delimitar trechos de dados, e deixa a interpretação do dado a ser realizada completamente para a aplicação que o está lendo. Resumindo, enquanto em um documento HTML uma TAG <p> indica um parágrafo, no XML essa TAG pode indicar um preço, um parâmetro, uma pessoa, ou qualquer outra coisa que tenha sido definida pelo criador do documento para essa TAG <p>.



Comparações entre HTML e XML

Os arquivos XML são arquivos-texto e facilitam que os programadores ou desenvolvedores "debuguem" mais facilmente as aplicações, de forma que um simples editor de textos pode ser usado para corrigir um erro em um arquivo XML.

Mas as regras de formatação para documentos XML são muito mais rígidas do que para documentos HTML. Uma TAG esquecida ou um atributo sem aspas torna o documento inutilizável, enquanto que no HTML isso é tolerado.

As especificações oficiais do XML determinam que as aplicações não podem tentar adivinhar o que está errado em um arquivo (no HTML isso acontece), mas sim devem parar de interpretá-lo e reportar o erro.

XML Mostra O QUE Significa

```
<pessoa>
   <nome>Albert Eije</nome>
   <email>alberteije@gmail.com</email>
   <telefone>
      <ddd>61</ddd>
      <numero>9999-8888</numero>
   </telefone>
</pessoa>
```

HTML Mostra COMO Apresentar

```
<html><body>
   <h1>Albert Eije</h1>
   <h2>alberteije@gmail.com</h2>
   <p>
      <b>61</b>
      <i>9999-8888</i>
   </p>
</body></html>
```

Características da Linguagem XML

Representação dos dados de forma estruturada

O XML fornece uma representação estruturada dos dados que é amplamente implementável e fácil de ser desenvolvida. O XML fornece um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações desde simples até as mais complexas, dentre elas:

- Um simples documento;
- Um registro estruturado tal como uma ordem de compra de produtos;
- Um objeto com métodos e dados como objetos Java ou controles ActiveX;
- Um registro de dados. Um exemplo seria o resultado de uma consulta a bancos de dados;
- Apresentação gráfica, como interface de aplicações de usuário;
- Entidades e tipos de esquema padrões;
- Todos os links entre informações e pessoas na Web.

Características da Linguagem XML

Representação dos dados de forma estruturada

Uma característica importante é que, uma vez tendo sido recebido o dado pelo cliente, tal dado pode ser manipulado, editado e visualizado sem a necessidade de novas requisições ao servidor. Com a redução de requisições economiza-se processamento nos servidores e banda na rede, pois menos dados serão trafegados.

O XML é considerado de grande importância na Internet e em grandes intranets porque permite que sistemas de diferentes plataformas conversem entre si, trocando informações dentro de um mesmo padrão.

Características da Linguagem XML

Separação entre apresentação e dados

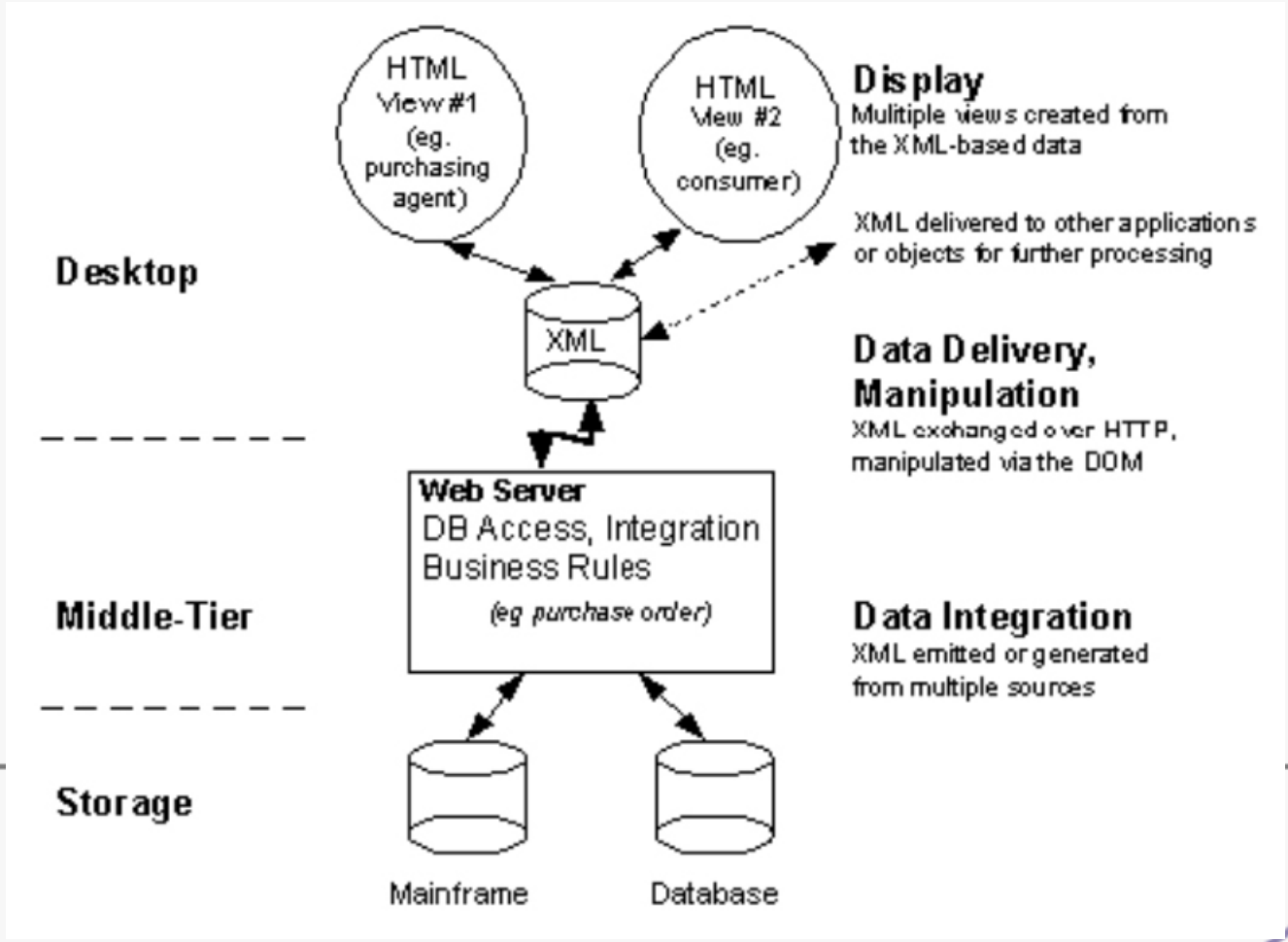
O HTML especifica como o documento deve ser apresentado na tela por um navegador. Já o XML define o conteúdo do documento. Por exemplo, em HTML são utilizadas TAGs para definir uma tabela, suas linhas e colunas. No XML você utiliza as TAGs para descrever os dados, como exemplo TAGs de assunto, título, autor, conteúdo, referências, datas, etc.

Assim como o HTML tem as folhas de estilo (o CSS) o XML também conta com esse recurso, que é conhecido como XSL (Extensible Style Language) para a apresentação de dados em um navegador. O XML separa os dados da apresentação e processo, o que permite visualizar e processar o dado como quiser, utilizando diferentes folhas de estilo e aplicações.

Características da Linguagem XML

Separação entre apresentação e dados

Observe na imagem abaixo uma aplicação Web três camadas que permite a troca de dados entre mainframes e aplicações desktop.

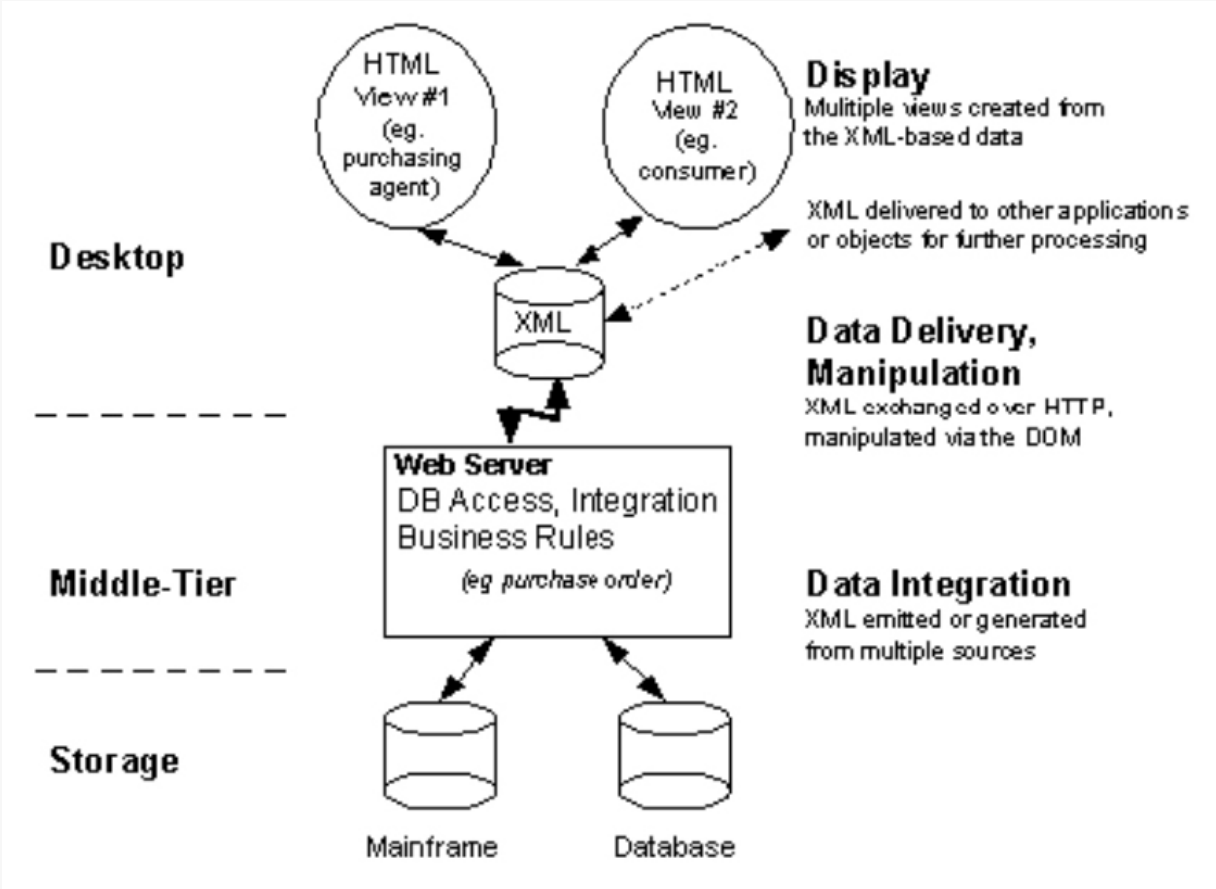


Características da Linguagem XML

Separação entre apresentação e dados

Observe que na Camada Desktop, a entrada de dados se dá através de uma interface HTML (um browser). Mas poderia ser através de outros clientes, tais como uma aplicação Swing feita em Java ou uma aplicação feita em Dot Net, por exemplo. O usuário poderia entrar com qualquer tipo de dado: compras, ordens de pagamento, resultados de busca, pedidos, catálogos, etc.

Na Camada Middle-Tier, um aplicativo rodando num servidor converte os dados em XML e a partir de então os dados poderão ser armazenados no banco de dados ou lidos por outras aplicações num computador de grande porte (mainframe).



Definição Conceitual do XML

Estrutura do Documento

Um documento XML é uma árvore rotulada onde um nó externo consiste de:

- Dados de caracteres (uma sequência de texto);
- Instruções de processamento (anotações para os processadores), tipicamente no cabeçalho do documento;
- Um comentário (nunca acompanhando com semântica);
- Uma declaração de entidade (simples macros);
- Nós DTDs (Document Type Definition).

Definição Conceitual do XML

Estrutura do Documento

Um nó interno é um elemento, o qual é rotulado com:

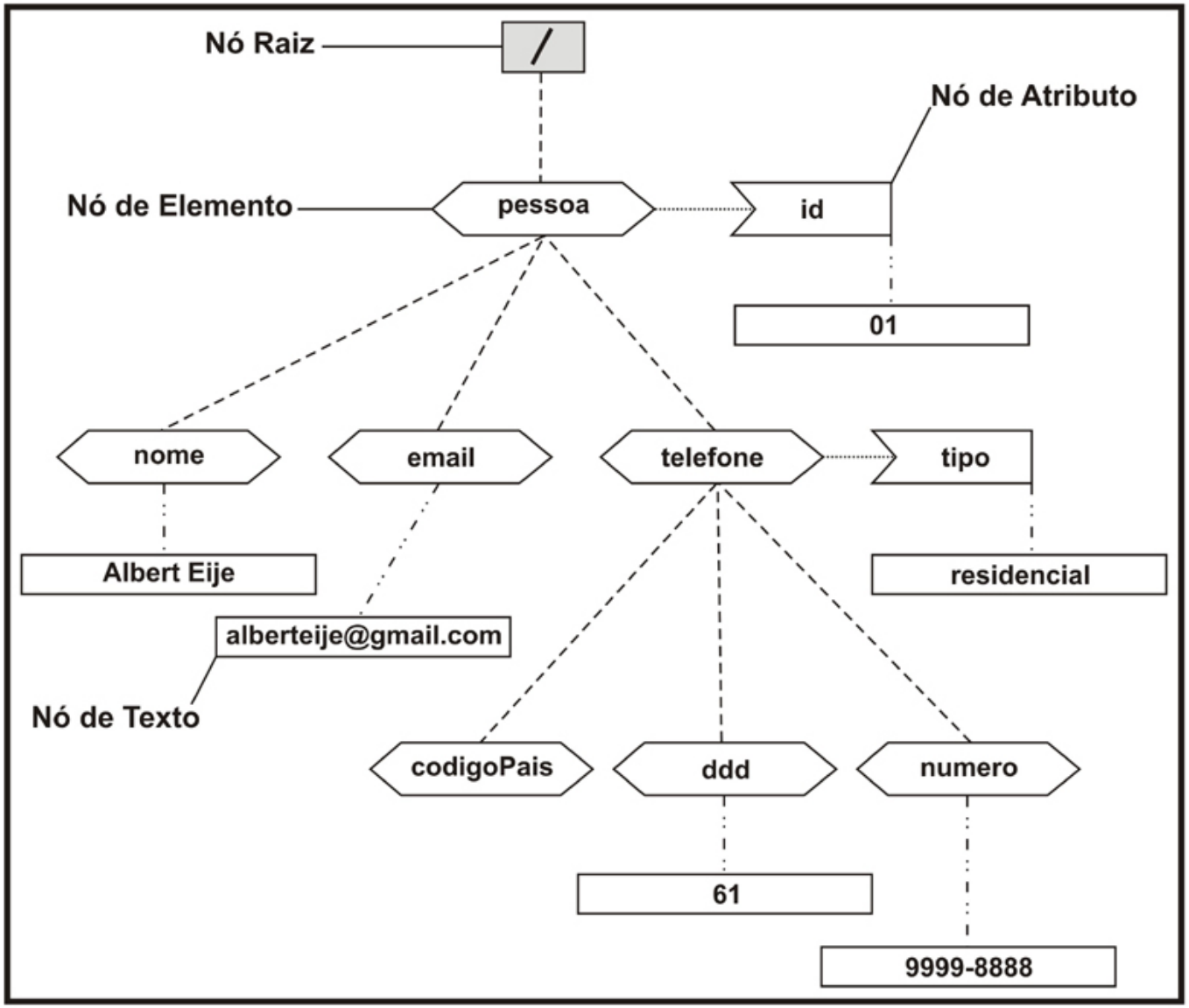
- Um nome ou;
- Um conjunto de atributos, cada qual consistindo de um nome e um valor.

Normalmente, comentários, declarações de entidades e informações DTD não são explicitamente representados na árvore.

Definição Conceitual do XML

Estrutura do Documento

Na imagem ao lado podemos ver a estrutura de um documento XML no formato de árvore.



Definição Conceitual do XML

Explicação das TAGs

Nós já vimos numa figura anterior o que são o elemento raiz, os demais elementos, os atributos e seus valores e o elemento vazio.

Os documentos XML são sensíveis a letras maiúsculas e minúsculas. Um documento XML é bem formatado quando segue algumas regras básicas. Tais regras são mais simples do que para documentos HTML e permitem que os dados sejam lidos e expostos sem nenhuma descrição externa ou conhecimento do sentido dos dados XML. Para que um documento XML esteja bem formatado deve seguir os seguintes princípios:

- Abrir e fechar corretamente todas as TAGs;
- As TAGs de elemento devem ser apropriadamente posicionadas;
- Os elementos não podem se sobrepor.

Definição Conceitual do XML

Explicação das TAGs

Veja abaixo uma sobreposição de elementos:

```
<nome>Albert Eije
 <sobrenome> Barreto Mouta </nome>
</sobrenome>
```

Observe que a TAG nome foi fechada antes da TAG sobrenome. Isso vai causar um erro no arquivo XML. No entanto, um documento XML apenas "bem formatado" tem pouca utilidade, pois:

- Não há especificação sobre quais TAGs podem ser usadas e para que servem;
- Não há verificação caso se digite uma TAG no local errado, com atributos errados ou mesmo com erro de digitação.

Definição Conceitual do XML

Documentos com DTDs

Para que um XML seja válido deve existir um modelo, uma gramática da estrutura do documento, que deve ser obedecida.

No XML as regras que definem um documento são ditadas por DTDs (Document Type Definition), as quais ajudam a validar os dados quando a aplicação que os recebe não possui internamente uma descrição do dado que está recebendo. Um analisador de documentos pode checar os dados que chegam analisando as regras contidas no DTD para ter certeza de que o dado foi estruturado corretamente. Os dados enviados sem DTD são conhecidos como dados bem formatados. Nesse caso, o documento pode ser usado para implicitamente se auto descrever, como o currículo vitae que vimos anteriormente.

Definição Conceitual do XML

Documentos com DTDs

Com os dados do XML bem formatados e válidos, o documento XML se torna auto descritivo, pois as TAGs dão ideia de conteúdo e estão misturadas com os dados. Como o documento é aberto e flexível, ele pode ser usado em qualquer lugar onde a troca ou transferência de informação é necessária: informações sobre páginas HTML, objetos ou regras de negócios, transações eletrônicas comerciais, etc. Os seguintes itens podem ser validados:

1. TAGs

- Que TAGs são permitidas;
- Que TAGs filhas são requeridas ou opcionais;
- Se a ordem ou a quantidade de elementos é importante;
- Que tipos de conteúdo são permitidos.

Definição Conceitual do XML

Documentos com DTDs

2. Atributos

- Que atributos são requeridos ou opcionais;
- Quais são os tipos de dados dos atributos;
- Se existem valores "padrão" para os atributos;

Para associar um DTD a um arquivo XML deve-se utilizar uma TAG específica indicando o arquivo DTD. Observe no exemplo na página seguinte.

Definição Conceitual do XML

Documentos com DTDs

```
<?xml version="1.0" ?>

<!DOCTYPE Pessoa PUBLIC "-//T2Ti//DTD Produtos//BR" "http://www.t2ti.com/dtd/Pessoa.dtd">

<pessoa>
 <nome>Albert Eije</nome>
 <email>alberteije@gmail.com</email>
 <telefone>
 <ddd>61</ddd>
 <numero>9999-8888</numero>
 </telefone>
</pessoa>
```

Observe que um arquivo DTD é seguido pela extensão dtd.

Definição Conceitual do XML

Documentos com Schemas

Utilizar Schema para criar uma gramática para o XML nos fornece mais recursos do que o DTD. O próprio Schema é um arquivo XML.

O Schema nos fornece uma abordagem OO para definir o documento XML. Nos fornece ainda um conjunto de tipos básicos. Esses tipos são bem mais abrangentes do que aqueles fornecidos pelo DTD (CDATA e PCDATA). Os tipos mais básicos estão disponíveis: byte, string, integer e float.

Dessa forma, o autor de um Schema pode criar esses tipos de dados, junto com modificadores e operadores, para criar outros tipos de dados mais complexos.

Definição Conceitual do XML

Documentos com Schemas

Para criar um Schema, deve-se criar primeiro um documento XSD:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <br>
<xsd:annotation> <br>
 <xsd:documentation xlm:lang="en"> <br>
 XML Schema – exemplo para livraria. <br>
 </xsd:documentation> <br>
</xsd:annotation>
```

Definição Conceitual do XML

Documentos com Schemas

Vamos criar o Schema com o nome e um tópico:

```
<xsd:element name="livraria" type="livrariaType"/> <br>
<xsd:complexType name="livrariaType"> <br>
 <xsd:sequence> <br>
 <xsd:element name="nome" type="xsd:string"/> <br>
 <xsd:element name="topico" type="topicoType" minOccurs="1"/> <br>
 </xsd:sequence> <br>
</xsd:complexType>
```

Definição Conceitual do XML

Documentos com Schemas

Primeiro definimos um elemento "livraria" do tipo "livrariaType". O "livrariaType" será então definido como um "complexType", que terá dois elementos: "nome" e "topico".

O elemento "nome" é do tipo String e o elemento "topico" é do tipo "topicoType", que deverá ser definido. Observe ainda que o elemento "topico" tem um atributo "minOccurs='1'", que indica que é obrigatório pelo menos um elemento.

Definição Conceitual do XML

DTD versus Schema

O DTD fornece uma gramática básica para definir um documento XML em termos dos metadados que compõem a forma do documento.

O Schema também fornece isso, mas vai além: fornece uma forma de definição do que os dados podem ou não podem conter. Fornece uma forma de trabalho Orientada a Objetos e dá muito mais poder ao desenvolvedor.

Padrões da Estrutura do XML

O XML é baseado em padrões de tecnologia comprovadamente otimizados para a Web.

Os padrões que compõem o XML são definidos pelo W3C (World Wide Web Consortium) e são os seguintes:

- Extensible Markup Language (XML) – é uma Recomendação, que é vista como o último estágio de aprovação do W3C. Isso significa que o padrão é estável e pode ser aplicado a Web e utilizado pelos desenvolvedores de ferramentas;

- XML Namespaces – é também uma Recomendação, a qual descreve a sintaxe de namespace, ou espaço de nomes, e que serve para criar prefixos para os nomes de TAGs, evitando confusões que possam surgir com nomes iguais para TAGs que definem dados diferentes;

Padrões da Estrutura do XML

- Document Object Model (DOM) Level 1 – é uma Recomendação que provê formas de acesso aos dados estruturados utilizando scripts, permitindo aos desenvolvedores interagir e computar tais dados consistentemente;

- Extensible Stylesheet Language (XSL) – O XSL apresenta duas seções: a linguagem de transformação e a formatação de objetos. A linguagem de transformação pode ser usada para transformar documentos XML em algo agradável para ser visto, assim como transformar para documentos HTML, e pode ser usada independentemente da segunda seção (formatação de objetos).

- XML Linking Language (XLL) e XML Pointer Language (XPointer) – O XLL é uma linguagem de construção de links que é similar aos links HTML, sendo que é mais poderosa, porque os links podem ser multidirecionais, e podem existir em nível de objetos, e não somente em nível de página.

Padrões da Estrutura do XML

Uma noção sobre DOM

DOM é uma API (Applications Programming Interface) independente de plataforma e linguagem que é utilizada para manipular as árvores do documento XML (e HTML também). DOMs são ideais para linguagem script, como exemplo JavaScript.

Essa API é definida em vários níveis:

- Nível 0: Funções existentes conhecidas das linguagens script dos browsers;
- Nível 1: Funcionalidade para navegação em documentos e manipulações;
- Nível 2: Adiciona modelos de style sheets (folhas de estilo), filtros, modelos de eventos, e suporte a namespaces;
- Nível 3: Possibilita as opções de carregar e salvar, DTDs, schemas, visualização de documentos e status de formatação.

Principais Benefícios da Linguagem XML

O XML tem por objetivo trazer flexibilidade e poder às aplicações Web. Dentre os benefícios para desenvolvedores e usuários temos:

- Buscas mais eficientes;
- Desenvolvimento de aplicações Web mais flexíveis. Isso inclui integração de dados de fontes completamente diferentes, de múltiplas aplicações;
- Computação e manipulação local dos dados;
- Múltiplas formas de visualização e atualização granulares do conteúdo;
- Distribuição dos dados via rede de forma mais comprimida e escalável;
- Padrões abertos.

Principais Benefícios da Linguagem XML

Buscas mais eficientes

Os dados em XML podem ser unicamente "etiquetados", o que permite que, por exemplo, uma busca por livros seja feita em função do nome do autor. Sem o XML é necessário para a aplicação de procura saber como é esquematizado e construído cada banco de dados que armazena os dados de interesse, o que é impossível. O XML permite definir livros por autor, título, assunto, etc. Isso facilita enormemente a busca.

Desenvolvimento de aplicações flexíveis para a Web

O desenvolvimento de aplicações Web em três camadas é altamente factível com o XML. Os dados XML podem ser distribuídos para as aplicações, objetos ou servidores intermediários para processamento. Esses mesmos dados também podem ser distribuídos para a camada cliente para serem visualizados em um navegador ou qualquer outro cliente que possa ler os arquivos XML.

Principais Benefícios da Linguagem XML

Integração de dados de fontes diferentes

Atualmente existem dados espalhados em diversos bancos de dados diferentes. O XML permite que tais dados possam ser facilmente combinados. Essa combinação seria feita via software em um servidor intermediário, estando os bancos de dados na extremidade da rede. Os dados poderiam ser distribuídos para outros servidores ou clientes para que fizessem o processamento, a agregação e a distribuição.

Múltiplas formas de visualizar os dados

Os dados recebidos por um usuário podem ser visualizados de diferentes formas uma vez que o XML define somente os dados e não o visual. A interpretação visual poderia ser dada de várias maneiras diferentes, de acordo com as aplicações. Os recursos de CSS e XSL permitem essas formas particulares de visualização.

Principais Benefícios da Linguagem XML

Computação e manipulação locais

Os dados XML recebidos por um cliente são analisados e podem ser editados e manipulados de acordo com o interesse do usuário. Ao contrário de somente visualizar os dados, os usuários podem manipulá-los de várias formas. Os recursos disponíveis do Document Object Model (DOM) permitem que os dados sejam manipulados via scripts ou outra linguagem de programação. A separação da interface visual dos dados propriamente ditos permite a criação de aplicações mais poderosas, simples e flexíveis.

Fácil distribuição na Web

Assim como o HTML, o XML, por ser um formato baseado em texto aberto, pode ser distribuído via HTTP sem necessidade de modificações nas redes existentes.

Principais Benefícios da Linguagem XML

Atualizações granulares dos documentos

Os dados podem ser atualizados de forma granular, evitando que uma pequena modificação no conjunto de dados implique na busca do documento inteiro novamente. Dessa forma, somente os elementos modificados seriam enviados pelo servidor para o cliente. O XML também permite que novos dados sejam adicionados aos já existentes, sem a necessidade de reconstrução da página.

Compressão

A compressão de documentos XML é fácil devido à natureza repetitiva das TAGs usadas para definir a estrutura dos dados. A necessidade de compressão é dependente da aplicação e da quantidade de dados a serem movidos entre clientes e servidores. Os padrões de compressão do HTTP 1.1 podem ser usados para o XML.

XML

Testando o Código – w3schools

Para aumentar o seu aprendizado em XML, que é importantíssimo para a construção de aplicações web, web services, etc, você vai precisar praticar. Você vai testar o código no site w3schools.com.

Você vai observar que o site fornece várias formas de visualização do XML:

- Display the XML File
- Display the XML File as a Note
- Display with XSLT

Experimente as opções, explore os exemplos e crie seus próprios arquivos.

XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Display the XML File » Display the XML File as a Note »

JavaScript


JAVASCRIPT
<div class="buttonb
<a href="Albums

<asp:ID="D
<ItemTemplate>
<table bord
<tr>
<td

JavaScript

Introdução

JavaScript é uma linguagem de programação interpretada, ou seja, não é preciso compilar o programa feito, o navegador interpreta cada linha de programação e age de acordo.

Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados no lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido.

É atualmente a principal linguagem para programação no lado do cliente (client-side) em navegadores web.

Note que se trata de uma linguagem de programação e, como tal, exige muito estudo. O que vamos abordar aqui será apenas a ponta do iceberg. O leitor deverá estudar profundamente a linguagem JavaScript. A trinca JavaScript + CSS + HTML é a base para construir aplicações web. É preciso ter conhecimento das três para avançar.

JavaScript

Introdução

A linguagem JavaScript foi criada pela Netscape Communications Corporation e foi desenvolvida com o nome de Mocha, depois passou a se chamar LiveScript e foi finalmente lançada como JavaScript em 1995 integrando a versão 2.0B3 do navegador Netscape e visava implementar uma tecnologia de processamento no lado do cliente.

A denominação da linguagem, JavaScript, se deve a similaridades com a sintaxe do Java. Fora isso, as duas linguagens não têm nenhuma outra relação.

JavaScript permite criar pequenos programas embutidos no próprio código de uma página HTML capazes de gerar números, processar alguns dados, verificar formulários, alterar valores de elementos HTML e criar elementos HTML.

Tudo isso diretamente no lado do cliente (no navegador), evitando a troca de informações com o servidor. Sendo assim, passa a depender somente do processamento local do cliente, não mais da latência da rede.

JavaScript

Introdução

JavaScript é uma linguagem completa e poderosa que possui muitas das qualidades de diversas outras linguagens, como: listas associativas, tipagem dinâmica, expressões regulares e sintaxe similar a C/C++.

Além disso, JavaScript é multi paradigma, ou seja, é possível programar de forma estruturada e orientada a objetos.

Lembrando que o JavaScript é uma linguagem interpretada, ou seja, não é compilada.

Para executar um programa compilado é necessário escrever o seu código fonte correspondente e compilá-lo. No caso do SO Windows, costumamos dizer "criar um executável". Durante a compilação, o código fonte é lido pelo compilador que gera então um arquivo de saída com uma tradução daquele código fonte para linguagem de máquina (o código executável). Esse arquivo em linguagem de máquina pode ser então executado no computador e não pode ser facilmente editado, pois não é compreensível por nós seres humanos.

JavaScript

Introdução

Para editar o programa compilado, será necessário acessar o seu código fonte original e compilar o programa novamente para criar um novo executável.

Diferente disso, para executar um programa em uma linguagem interpretada (script) precisamos apenas digitar o código fonte e o interpretador irá ler esse código e executar as instruções, comando por comando, a partir do próprio texto do código fonte, cada vez que o script for rodado. Para alterar o programa basta alterar o código e ele já estará pronto para rodar novamente.

Podemos observar que existe uma vantagem óbvia na utilização do script: a agilidade para se alterar o programa, eliminando a sequência editar-compilar-linkar-rodar comum em softwares compilados.

Como preço dessa flexibilidade, perde-se um pouco em desempenho e será sempre necessário possuir um interpretador no computador onde será rodado o script. Além disso, seu código estará sempre visível, podendo ser facilmente copiado. JavaScript é uma linguagem de script.

Conceitos

Em relação aos tipos de dados, podemos dizer que o JavaScript é uma linguagem fracamente tipada. Linguagens fracamente tipadas são aquelas que não se importam com o tipo de dados contido em uma variável. Permitem que o programador não tenha de fazer conversões de tipos (cast).

Alguns preferem chamar de linguagem dinamicamente tipada. O nome fica menos depreciativo dessa forma. Mas existe algum problema nisso? Não diria "problema", mas, como tudo nessa vida, tem lá suas vantagens e desvantagens.

Uma linguagem que obriga a adotar um tipo e não alterá-lo costuma ter o seu código mais legível. Ou seja, se um tipo é definido como Inteiro, ele será Inteiro e pronto. Não dá pra alterar para String. Se tentar atribuir uma String para um Inteiro, ocorre um erro. Já numa linguagem fracamente tipada, você pode atribuir qualquer tipo de dado à variável e não ocorrerão erros. O código fica bem flexível, mas, dependendo de como você o escreve, pode ficar muito confuso, de modo que somente quem escreveu consiga dar manutenção. Pense nisso!

JavaScript

Conceitos

JavaScript é uma linguagem de programação que possibilita a definição de funções de ordem superior.

São funções que recebem uma ou mais funções como argumentos ou que têm uma função como saída.

Com isso, é possível criar o que são chamadas *function factories* que são funções que a partir de outras funções simples são capazes de realizar ações mais complexas.

JavaScript é uma linguagem que nasceu para rodar no lado do cliente – c*lient-side* (que roda no computador cliente). Mas, existe a possibilidade de ser executada no servidor também.

No entanto, é mais utilizado no lado do cliente. Quando o programa é criado com esta característica ele é enviado para o computador cliente ainda na forma de código fonte, que só então é interpretado e executado, dependendo assim unicamente da capacidade de processamento do cliente.

Conceitos

Por ser uma linguagem que é executada no computador do cliente, o JavaScript precisa ter severas restrições para evitar que se façam códigos maliciosos que possam causar danos ao usuário. Tais restrições existem para garantir a segurança do usuário.

As principais limitações do JavaScript para garantia de segurança são a proibição de:

- Abrir e ler arquivos diretamente da máquina do usuário.
- Criar arquivos no computador do usuário (exceto cookies).
- Ler configurações do sistema do usuário.
- Acessar o hardware do cliente.
- Iniciar outros programas.
- Modificar o valor de um campo de formulário do tipo <input>.

Essas limitações interferem muito pouco no desenvolvimento de aplicações web.

Veremos em livros posteriores que existem Apps híbridas que tem a capacidade de acessar os recursos do hardware através de plugins.

Tipos de Dados

O JavaScript possui tipos de dados parecidos com os de outras linguagens.

Tipos Numéricos

Em JavaScript os números são representados pelo padrão IEEE 754. Todos os valores numéricos são "declarados" pela simples atribuição dos valores a uma variável. Temos o inteiro e o ponto flutuante, conforme exemplos a seguir.

```
var x = 35; //inteiro
var x = 12,3; //ponto flutuante
```

Booleano

Uma variável do tipo booleano pode assumir apenas dois valores: true e false. Os valores deste tipo são em geral usados pela linguagem como resultado de comparações e podem ser usados pelo usuário para valores de teste ou para atributos que possuam apenas dois estados. Equivale ao uso de um inteiro com valores 0 ou 1 na linguagem C.

O JavaScript converte automaticamente true para 1 e false para 0 quando isso for necessário.

Tipos de Dados

Indefinido

Uma variável é indefinida quando ela foi declarada de alguma forma mas não possui nenhum valor concreto armazenado. Quando tentamos acessar uma variável que não teve nenhum valor associado a ela teremos como retorno "undefined" (indefinido).

```
var eije;
window.alert(eije);
// ao tentar exibir o conteúdo de eije
// teremos o retorno "undefined"
// pois não há valor associado a ela
```

Null

O null é a ausência de valor. Quando atribuímos null a um objeto ou variável significa que essa variável ou objeto não possui valor válido. Para efeito de comparação, se usarmos o operador de igualdade "==", os valores null e undefined são considerados iguais. No entanto, se for necessário diferenciar os dois valores, é recomendável o uso do operador "===" de identidade. Assim, para efeito de comparação, undefined e null são iguais, mas não são idênticos.

Tipos de Dados

Strings

Strings são sequencias de caracteres. Em JavaScript, a string pode ser tanto um tipo primitivo de dado como um objeto. No entanto, ao manipulá-la temos a impressão de que sejam objetos, pois as strings em JavaScript possuem métodos que podemos invocar para realizar determinadas operações sobre elas. Essa confusão ocorre porque quando criamos uma string primitiva, o JavaScript cria também um objeto string e converte automaticamente entre esses tipos quando necessário.

Arrays

Os Arrays são pares do tipo "inteiro-valor" para se mapear valores a partir de um índice numérico. Em JavaScript os Arrays são objetos com métodos próprios. Um objeto do tipo Array serve para guardar uma coleção de itens em uma única variável.

```
var array = new Array(ele1, ele2);
//Cria um array com elementos
var array = [1, 2, 3, 4];
//Cria um array com elementos
```

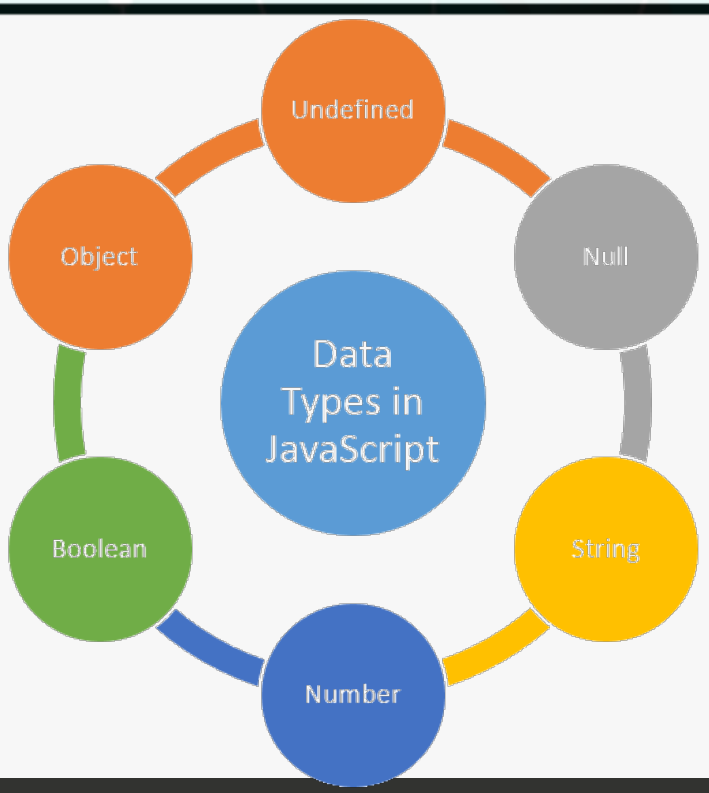
Tipos de Dados

Arrays

Para acessar as variáveis dentro de um array basta usar o nome do array e o índice da variável que se deseja acessar.

```
array[1] = 42;
document.write(array[1]);
//imprime o conteúdo de array[1]
```

Os arrays podem conter valores de tipos diferentes. É possível colocar num mesmo array inteiros, strings, booleanos e qualquer objeto que quiser.



JS

JavaScript

Operadores - Aritméticos

Operador	Operação	Exemplo
+	Adição	x+y
-	Subtração	x-y
*	Multiplicação	x*y
/	Divisão	x/y
%	Módulo (resto da divisão inteira)	x%y
-	Inversão de sinal	-x
++	Incremento	x++ ou ++x
--	Decremento	x-- ou --x

Operadores - Comparação

Operador	Função	Exemplo
==	Igual a	(x == y)
!=	Diferente de	(x != y)
===	Idêntico a (igual e do mesmo tipo)	(x === y)
!==	Não Idêntico a	(x !== y)
>	Maior que	(x > y)
>=	Maior ou igual a	(x >= y)
<	Menor que	(x < y)
<=	Menor ou igual a	(x <= y)

Operadores - Bit a Bit

Operador	Operação	Exemplo
&	E (AND)	(x & y)
|	OU (OR)	(x | y)
^	Ou Exclusivo (XOR)	(x ^ y)
~	Negação (NOT)	~x
>>	Deslocamento à direita (com propagação de sinal)	(x >> 2)
<<	Deslocamento à esquerda (preenchimento com zero)	(x << 1)
>>>	Deslocamento à direita (preenchimento com zero)	(x >>> 3)

Operadores - Atribuição

Operador	Exemplo	Equivalente
=	x = 2	Não possui
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
&=	x &= y	x = x & y
|=	x |= y	x = x | y
^=	x ^= y	x = x ^ y
>>=	x >>= y	x = x >>= y
<<=	x <<= y	x = x <<= y
>>>=	x >>>= y	x = x >>>= y

Operadores - Lógicos

Operador	Função	Exemplo
&&	E Lógico	(x && y)
||	OU Lógico	(x || y)
!	Negação Lógica	!x

Estruturas de Controle

If . Else

A estrutura if é usada quando se deseja verificar se determinada expressão é verdadeira ou não, e executar comandos específicos para cada caso. Assim, se a expressão for avaliada como verdadeira, o primeiro bloco de comandos é executado. Caso seja avaliada como falsa, o bloco de comandos que segue o else será executado. Também é possível aglomerar mais testes, utilizando-se o comando else if.

```
var teste = 50;
if (teste < 50) {
   window.alert("teste menor que 50");
} else if (teste > 50) {
   window.alert("teste maior que 50");
} else { window.alert("teste igual a 50"); }
```

Estruturas de Controle

Switch . Case

As estruturas do tipo switch são usadas quando queremos selecionar uma opção dentre várias disponíveis. Ao contrário de outras linguagens, os valores de comparação podem ser strings além de valores numéricos.

```
var letra = "a";
switch (letra) {
  case "a":
     document.write("é a letra .a.");
     break;
  case "b":
     document.write("é a letra .b.");
     break; }
```

Estruturas de Controle

While

Os laços do tipo while são usados quando se deseja que uma sequencia de ações seja executada apenas no caso da expressão de condição ser válida.

```
var array = [5, 2];

while ((array[0]+array[1]) < 15) {
  array[0]+=1;
  array[1]+=2;
  document.write("array0 = " + array[0] + "array1 = " + array[1]);
}
// O laço será feito enquanto a soma dos dois itens do array seja menor que 15
```

Estruturas de Controle

Do . While

Diferentemente do while, o "do . while" primeiro executa o conteúdo do laço uma vez e, depois disso, realiza o teste da expressão para decidir se continuará executando o laço.

```
var array = [5, 2];

do {
  array[0]+=1;
  array[1]+=2;
  document.write("array0 = " + array[0] + "array1 = " + array[1]);
} while ((array[0]+array[1]) < 15)

// os elementos serão somados uma vez antes de a condição ser testada
```

Estruturas de Controle

For

Na maioria das vezes, quando usamos um laço do tipo *while* também construímos uma estrutura com um contador que é incrementado a cada passo para controle do laço e manipulação interna de objetos. Os laços *for* oferecem a vantagem de já possuírem em sua estrutura essa variável de contador e incrementá-la de maneira implícita.

```
var array = [5, 2, 3];

for(var i=0; i<3; i++) {
   array[i]++;
}
// Ao final do laço cada elemento do array será incrementado em 1
```

JavaScript

Funções

Funções possuem um papel muito importante na programação estrutural pelo fato de ajudar muito na modularização no programa, ou seja, viabiliza a divisão do programa em partes menores e logicamente relacionadas. Em JavaScript, existem diversas maneiras de se declarar uma função.

Um ponto importante é que em JavaScript as funções são consideradas como dados, ou seja, podemos atribuir uma função a uma variável ou propriedade de um objeto e a partir desde momento usar a variável ou a propriedade da mesma for ma que se usaria a função.

Elas também podem ser passadas como argumentos para outras funções e por isso funções de JavaScript são chamadas funções de alta ordem, elas podem tanto receber funções como argumento quanto retornar uma função.

Funções – Expressão *function*

A primeira maneira de se declarar uma função é através do uso da palavra chave *function* de maneira similar a como elas são declaradas na linguagem C, com as diferenças de que em JavaScript não definimos o tipo de retorno e nem mesmo o tipo dos argumentos. Uma função complexa pode ser capaz de tratar argumentos diferentes e retornar argumentos diferentes dependendo das circunstâncias nas quais foi invocada.

```
function incrementaArray(array, valor) {
   for(item in array) {
      array[item] += valor;
   }
   return array;
}
// Para invocar essa função depois basta usar incrementaArray(argumento1, argumento2)
```

Funções – Construtor *function*

A segunda forma de se declarar uma função é utilizando o construtor *Function()* e o operador *new*, pois em JavaScript funções e objetos são interligados. Esse construtor aceita um número qualquer de strings como argumentos. O ultimo argumento será sempre o corpo da função contendo comandos separados por ponto-e-virgula normalmente e todos os outros argumentos do construtor serão considerados argumentos da função que se está criando. Devido a sua estrutura, essa forma de se declarar funções costuma ser mais usada quando precisamos declarar um função pequena, ocupando apenas uma linha. Preze sempre pela legibilidade. Deixe seu código legível.

```
var areaTri = new Function("b","h","return (b*h)/2;");
// a função acima calcula a área de um triângulo dadas sua base
// altura. Para invocá-la basta usar areaTri(arg1,arg2)
```

Funções – Literais

Uma terceira forma de declarar uma função em JavaScript é através de literais. Essa forma é basicamente a mesma que declarar através do construtor Function(). No entanto, ela é melhor porque os comandos podem ser declarados com a sintaxe normal do JavaScript ao invés de ser uma string como é o caso do construtor. Com literais não há necessidade de manter a função em uma linha, dentro das chaves podemos construir a função usando um comando por linha normalmente.

```
var areaTri = function(b, h) { return (b*h)/2; };
// a função acima calcula a área de um triângulo dadas sua base
// altura. Para invocá-la basta usar areaTri(arg1,arg2)
```

Objetos

Ao contrário de uma variável, um objeto pode conter diversos valores e tipos diferentes armazenados nele (atributos) e também possuir funções que operem sobre esses valores (métodos). Tanto os atributos, quanto os métodos, são chamados de propriedades do objeto. Para criar um objeto é muito simples, basta invocar seu construtor através do operador new.

```
var objeto = new Object();
// Quando usamos o construtor Object() criamos um objeto genérico
```

É possível criar objetos através de literais:

```
var homem = {
nome: "joão paulo",
idade: 25,
}
```

Objetos

Para acessar uma propriedade do objeto, basta usar "objeto.propriedade" e no caso de métodos adicionar o parêntesis "objeto.execute()". Os métodos são simplesmente funções que são invocadas por meio de um objeto.

Também podemos definir os métodos dentro do próprio construtor de uma função, tanto definindo a função fora e atribuindo no construtor, como definindo a própria função dentro do próprio construtor uma vez que JavaScript suporta o aninhamento de funções.

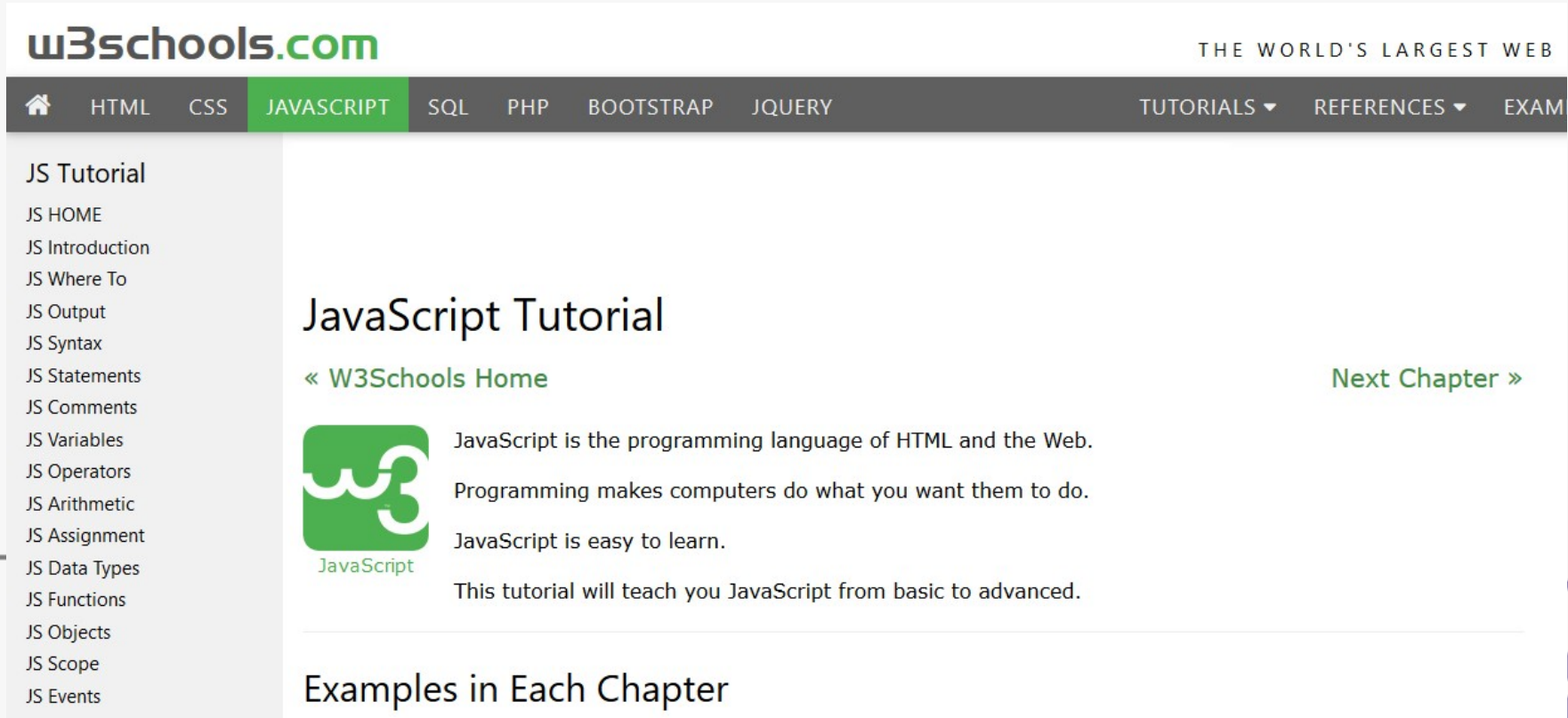
```
// Uma função fictícia para cálculo de um consumo de combustível
function calcula_peso() {
   return 50;
}

homem.peso = calcula_peso();
```

Testando o Código – w3schools

Chegou o momento de exercitar. O leitor deve acessar o site w3schools para isso.

O site contém um tutorial com diversos exemplos. Vá clicando no menu do lado esquerdo para explorar cada um dos exemplos.

Divirta-se!



Ajax

Ajax

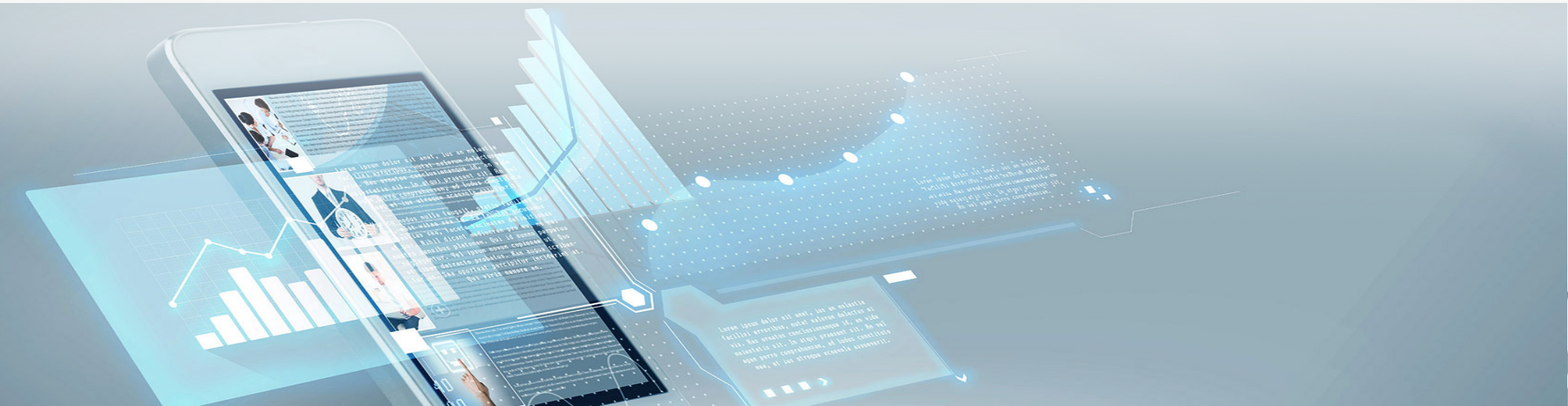
Introdução

O nome Ajax foi cunhado por Jesse James Garrett em 2005 para se referir a um conjunto de tecnologias que já existiam, mas que passaram a ser usadas de forma inovadora, enriquecendo as possibilidades de interação na web para torná-las o mais próximo de aplicações desktop quanto possível.

As páginas web sempre sofreram com falta de interação devido a própria natureza caótica da internet, de possuir uma latência alta e ser pouco confiável.

Assim, é muito comum clicarmos em um link e termos que esperar as vezes até alguns segundos, dependendo da conexão, até que a próxima página seja carregada do servidor para sua máquina.

É claro que as conexões têm melhorado dia a dia, mas ainda assim o simples fato de que a cada mudança de página precisamos exibir um documento totalmente novo que contém, além dos dados que requisitamos, todas as informações de layout novamente, representa um gasto de banda considerável.

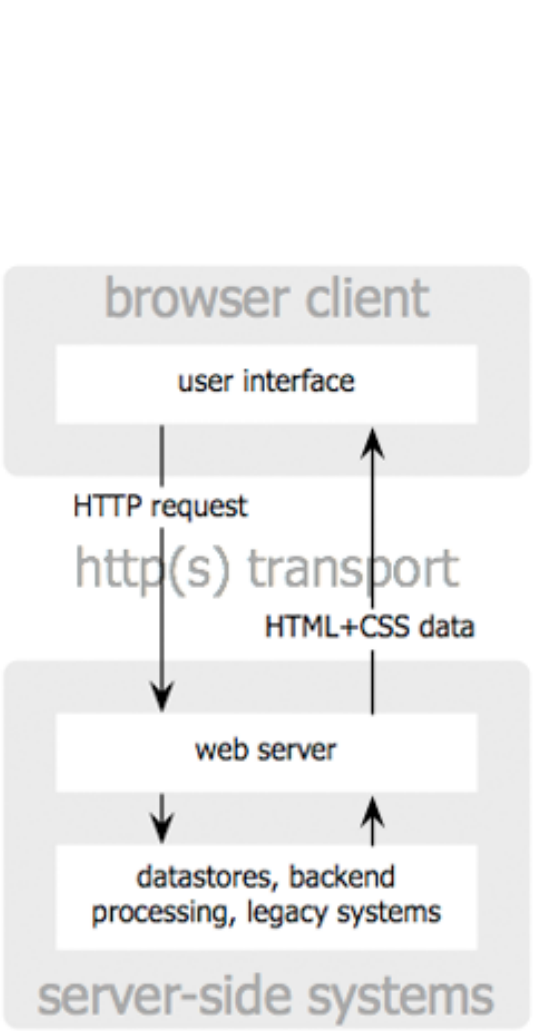


Ajax

Introdução

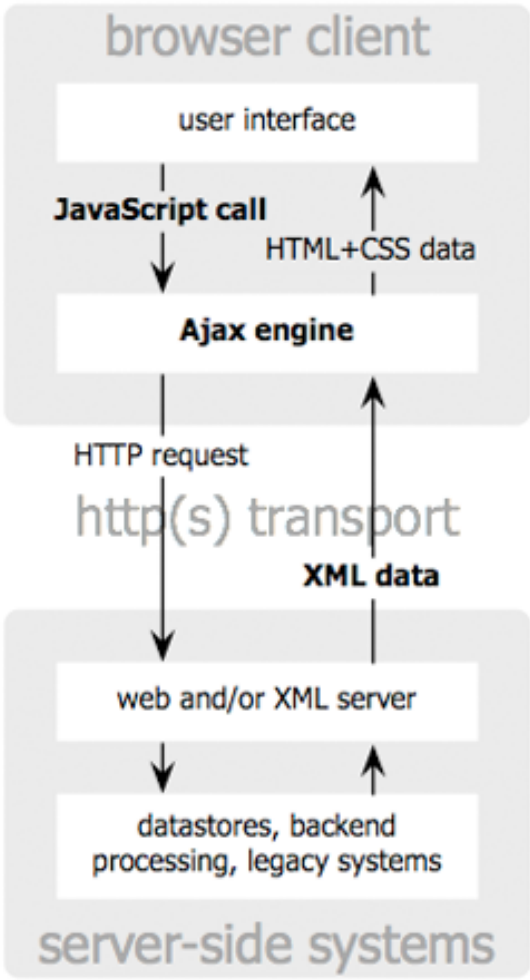
Uma grande diferença do Ajax é que as páginas apresentadas no browser passam a ser aplicações, ou seja, a primeira vez que entramos em uma página, a aplicação é carregada para nossa máquina e depois essa aplicação fica responsável por requisitar ou enviar os dados para o servidor de forma assíncrona.

Como a aplicação está o tempo todo no browser do cliente, este não perde a interação e pode realizar ações mesmo enquanto espera a requisição de algum dado do servidor. As aplicações Ajax possuem uma camada a mais entre a interface com o usuário e o servidor, essa camada é a aplicação propriamente dita.



classic
web application model

Jesse James Garrett / adaptivepath.com



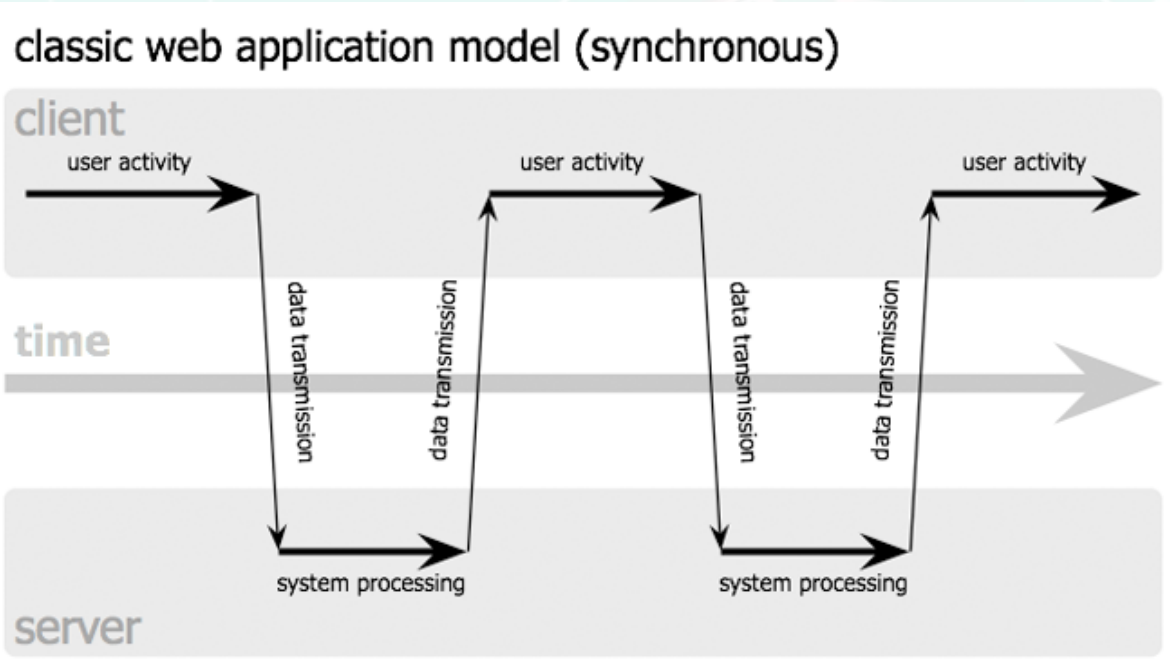
Ajax
web application model

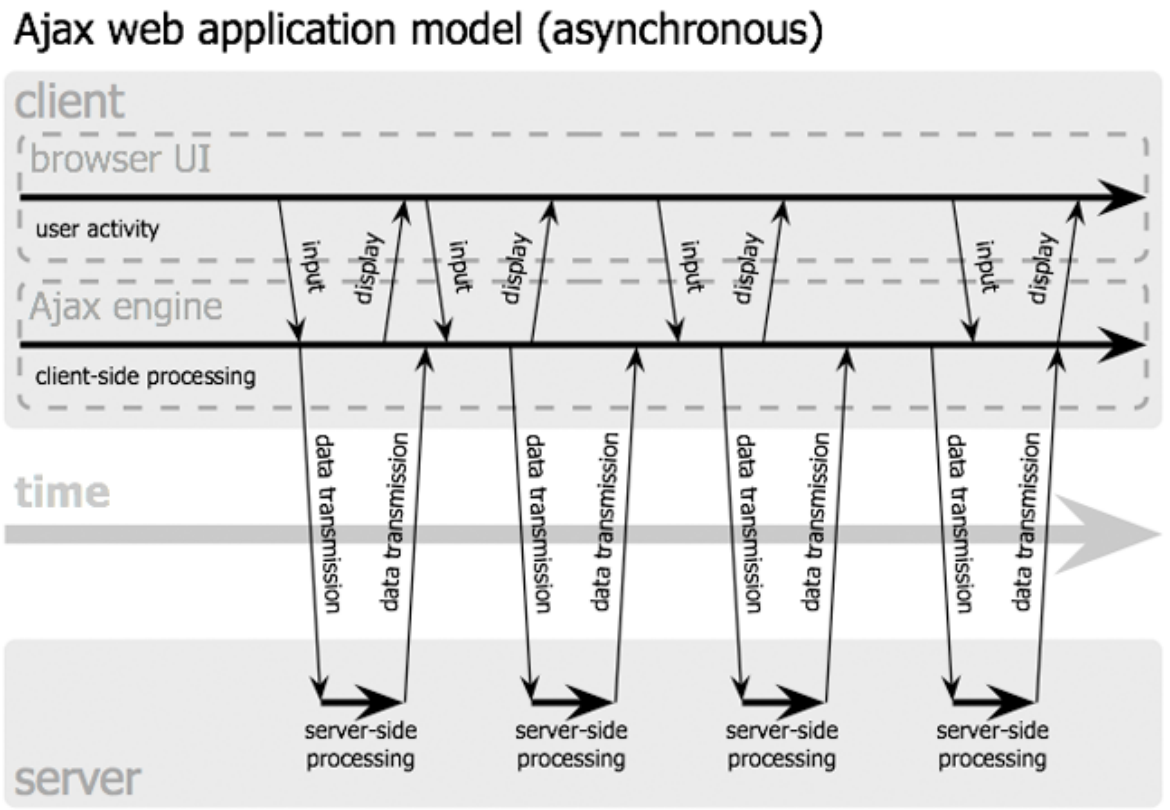
Ajax

Introdução

Uma grande diferença do Ajax é que as páginas apresentadas no browser passam a ser aplicações, ou seja, a primeira vez que entramos em uma página, a aplicação é carregada para nossa máquina e depois essa aplicação fica responsável por requisitar ou enviar os dados para o servidor de forma assíncrona.

Como a aplicação está o tempo todo no browser do cliente, este não perde a interação e pode realizar ações mesmo enquanto espera a requisição de algum dado do servidor. As aplicações Ajax possuem uma camada a mais entre a interface com o usuário e o servidor, essa camada é a aplicação propriamente dita.

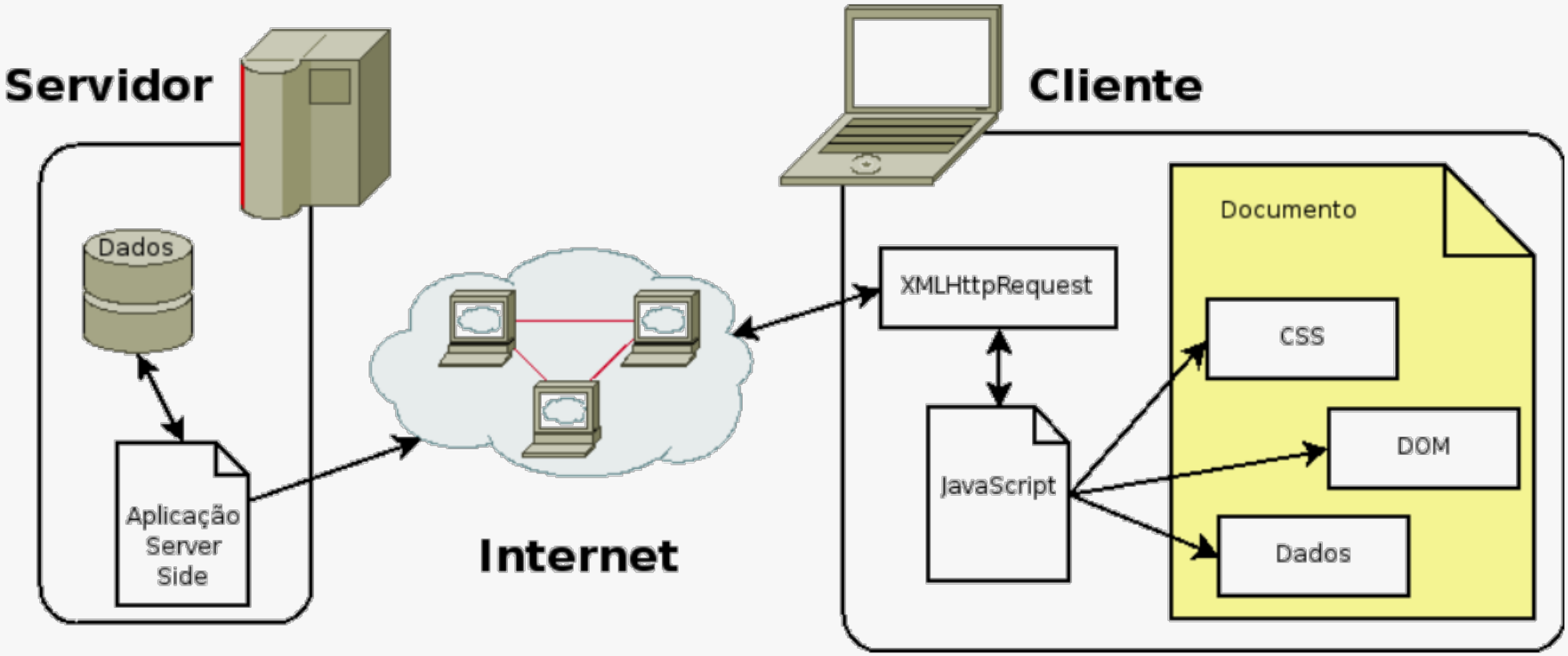




Jesse James Garrett / adaptivepath.com

Tecnologias

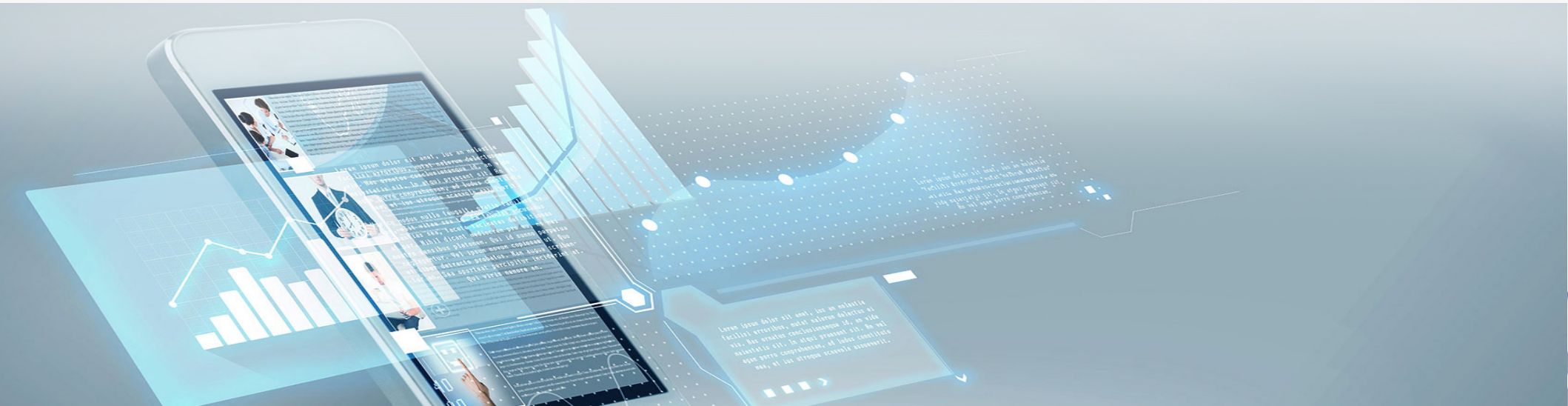
As quatro principais tecnologias utilizadas para o desenvolvimento de aplicações Ajax são esquematizadas na imagem abaixo.



Tecnologias

Elas desempenham as seguintes funções:

- JavaScript é responsável por interligar todas as outras tecnologias, é a linguagem de programação e portanto com ela é desenvolvida a aplicação que irá ser executada na máquina do cliente.
- CSS (Cascade Style Sheets) é um padrão da W3C para estilizar elementos em uma página web. Ele é utilizado para dar uma boa aparência às páginas, podendo ser acessado e editado pelo JavaScript.
- DOM (Document Object Model) permite que uma linguagem como o JavaScript possa manipular e alterar a estrutura de documentos, por exemplo, uma página durante seu tempo de vida no browser do cliente.
- XMLHttpRequest é um objeto existente em JavaScript que permite a troca de dados com o servidor de forma assíncrona.

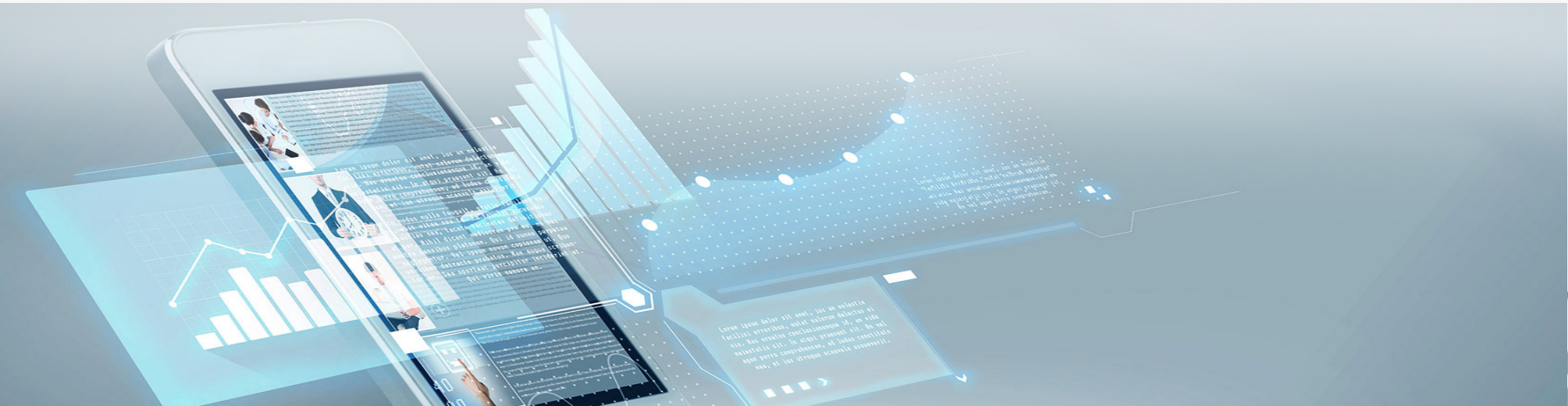


Tecnologias

Com o código JavaScript, podemos acessar todos os elementos da árvore DOM de um documento e podemos alterá-los, removê-los ou mesmo inserir um novo elemento. Isso nos ajuda a exibir os dados novos que foram requisitados do servidor pelo XMLHttpRequest.

Vamos começar a utilizar alguns códigos para que o leitor pratique e veja como o Ajax funciona.

Para realizar a prática, vamos acessar o site que sugerimos nos últimos livros, o w3schools.

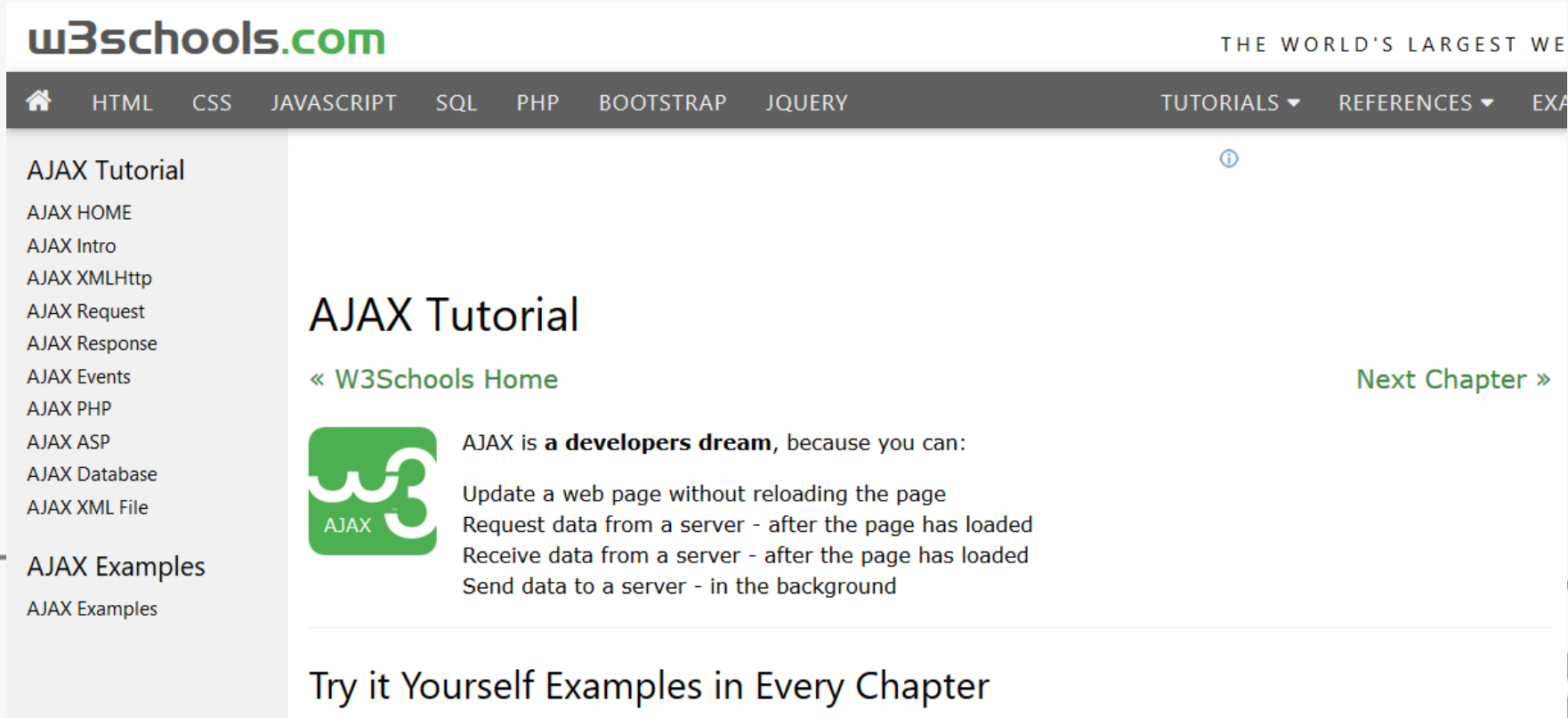


Testando o Código – w3schools

Chegou o momento de exercitar. O leitor deve acessar o site w3schools para isso.

O site contém um tutorial com diversos exemplos. Vá clicando no menu do lado esquerdo para explorar cada um dos exemplos.

Para testar os exemplos do livro, basta copiar o código e colar no editor do w3schools.



Explorando

```
<!DOCTYPE html>
<html><head>
<script type="text/javascript" src="dom.js"></script>
</head>
<body>
<div id="header">
Exemplos de manipulação do DOM por JavaScript.
</div>
<hr />
<br />
<input type="button" value="Altera árvore DOM" onclick="divEdit()" />


<script>
function divEdit() {
var header = document.getElementById("header");
var conteudo = header.innerHTML;
header.innerHTML = "<strong>"+conteudo+"</strong>";
var paragrafo = document.createElement('p');
paragrafo.setAttribute('title','Novo parágrafo');
var txt = document.createTextNode('Parágrafo adicionado a árvore DOM');
paragrafo.appendChild(txt);
header.appendChild(paragrafo);
}
</script>

</body></html>
```

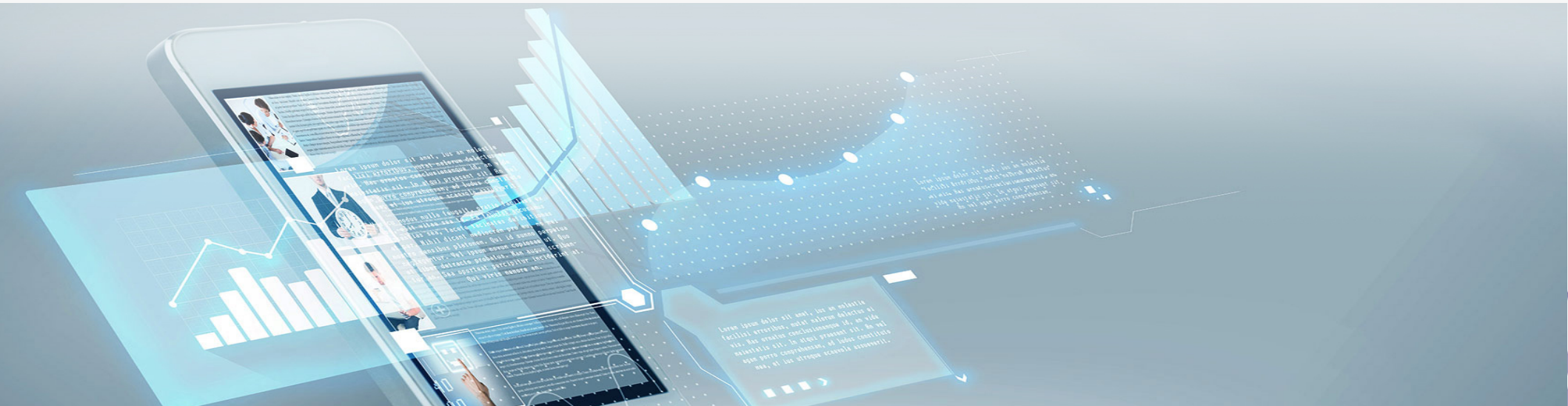
Explorando

Quando executar o exemplo da página anterior em seu browser, a página irá conter apenas uma frase no início e um botão com o nome "Altera árvore DOM".

Quando for clicado esse botão, a função divEdit() é chamada e insere um novo parágrafo na página sem necessidade de recarregá-la.

Este exemplo simples mostra apenas como usar JavaScript para manipular a árvore DOM, não entramos na área do Ajax ainda, mas essa é a base para o tratamento dos dados obtidos através da técnica.

O próximo exemplo é bem simples, mas já implementado em Ajax.



Explorando

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

<div id="demo"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("demo").innerHTML = xhttp.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

Ajax

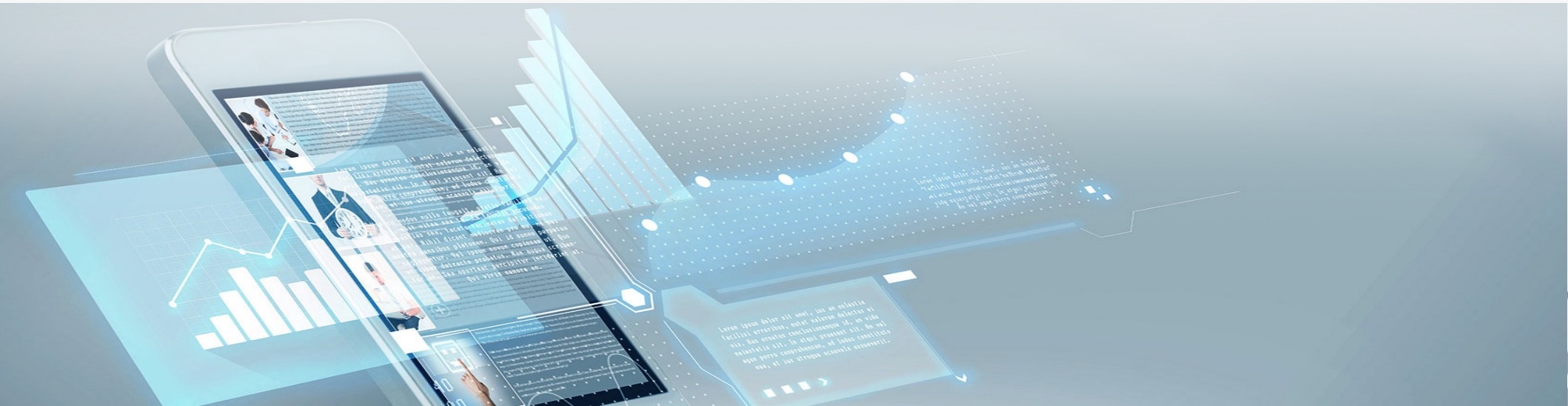
Explorando

O exemplo anterior, é o primeiro que você verá no tutorial do w3schools. A página HTML contém uma seção e um botão. Essa , que tem um "id" chamado de "demo" será utilizada para exibir dados que virão do servidor.

O botão possui um evento "OnClick" que será executado quando o usuário clicar no botão. Tal evento fará com que o método "loadDoc()" seja executado.

O método "loadDoc()" requisita dados do servidor que estão no arquivo "ajax_info.txt". O conteúdo desse arquivo será exibido na cujo "id" é igual a "demo".

E esse arquivo "ajax_info.txt" está mesmo no servidor? Para testar, basta inserir os seguinte caminho: "http://www.w3schools.com/ajax/ajax_info.txt". Observe o conteúdo do arquivo e veja que é exatamente o que aparece na quando clicamos no botão.

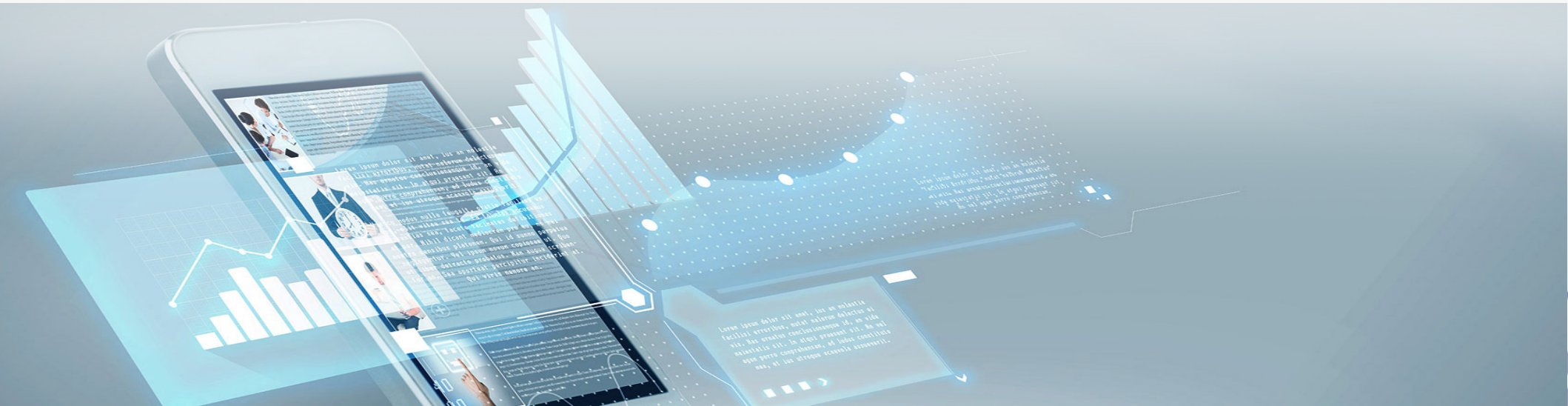


Considerações

Bom, se você tinha calafrios quando ouvia o termo Ajax, pode ficar tranquilo. Ele é tão simples quanto o que vimos anteriormente. Agora é só praticar. Evidentemente, existirão rotinas mais complexas, mas tais rotinas surgirão naturalmente de acordo com os requisitos da aplicação.

Muitas vezes você vai utilizar um framework ou um componente que vai facilitar sua vida, todo implementado em Ajax. Você não vai precisar alterar o framework, apenas utilizá-lo. Mas, saber como as coisas funcionam é sempre uma boa ideia.

O que você deve fazer agora é concluir o tutorial do w3schools. Se quiser se aprofundar no assunto, basta pesquisar um pouco mais sobre Ajax no Google ou mesmo no Youtube.



Referências

The World Wide Web Consortium (W3C)
http://www.w3.org/

Wikipedia, a enciclopédia livre
http://en.wikipedia.org/ | http://pt.wikipedia.org/

Website W3Schools
https://www.w3schools.com/