

SISTEMAS WEB



Sistemas web

Sistemas web

Merris Mozer

Adriane Aparecida Loper

Danilo Augusto Bambini Silva

© 2014 by Editora e Distribuidora Educacional S.A.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Diretor editorial e de conteúdo: Roger Trimer

Gerente de produção editorial: Kelly Tavares

Supervisora de produção editorial: Silvana Afonso

Coordenador de produção editorial: Sérgio Nascimento

Editor: Casa de Ideias

Editor assistente: Marcos Guimarães

Revisão: Carla Camargo Martins

Diagramação: Casa de Ideias

Dados Internacionais de Catalogação na Publicação (CIP)

Mozer, Merris

M939s Sistemas WEB / Merris Mozer, Adriane Aparecida

Loper, Danilo Augusto Bambini Silva. – Londrina: Editora
e Distribuidora Educacional S.A., 2014.

192 p.

ISBN 978-85-68075-59-3

1. Conceitos. 2. História. I. Loper, Adriane Aparecida. II.
Silva, Danilo Augusto Bambini. III. Título.

CDD 5.276

Sumário

Unidade 1 — Conceitos básicos de html.....	1
 Seção 1 O que é uma página web.....	2
1.1 Você conhece e sabe o que é uma página web?	2
1.2 O que são tags.....	3
1.3 Estrutura básica de uma página html	4
1.4 O que é um browser	5
1.5 Como salvar o documento html	6
 Seção 2 Usando tags para melhorar um documento html.....	9
2.1 Cabeçalho	9
2.2 Parágrafo e quebra de linhas	11
2.3 Formatando documento html	14
2.4 Textos pré-formatados	16
2.5 Formatando fonte	18
2.6 Linha de separação	21
2.7 Alinhamentos.....	24
2.8 Títulos.....	28
2.9 Cor de fundo.....	29
Unidade 2 — Comandos em html.....	41
 Seção 1 Imagens em html e links	43
1.1 Imagens para web.....	43
1.2 Links	48
 Seção 2 Listas, tabelas e formulários em html.....	52
2.1 Listas.....	52
2.2 Tabelas.....	56
2.3 Formulários.....	62

Seção 3 Frames, animação simples (marquee) em html	70
3.1 Frames <frameset>.....	70
3.2 Marquee	74
Unidade 3 — HTML 5.....	79
 Seção 1 Multimedia	80
1.1 A declaração <!DOCTYPE>.....	80
1.2 Estrutura mínima	81
1.3 Multimídia	82
1.4 Vídeo.....	82
1.5 Legendas para áudio ou vídeo	87
1.6 Plugins.....	91
 Seção 2 Formulários.....	96
2.1 Campo e-mail	96
2.2 Required	99
2.3 Pattern.....	99
2.4 Novalidate	100
2.5 Entradas de dados para números, datas e hora	101
2.6 Outros tipos novos de entrada de dados.....	105
2.7 Listas e autocompletar	105
2.8 Autofocus	107
2.9 Placeholder.....	107
 Seção 3 Semântica	110
3.1 Compatibilidade	113
3.2 Figure	115
3.3 Outros novos elementos semânticos	116
3.4 Javascript	117
Unidade 4 — CSS (Cascading Style Sheets).....	123
 Seção 1 CSS	124
1.1 História.....	125
1.2 Vantagens de utilizar CSS.....	126
 Seção 2 Regras e sintaxe	127
2.1 A regra CSS e sua sintaxe	127
2.2 Relacionamento entre seletores.....	128

2.3	Tipos de seletores.....	129
2.4	Pseudoclasses	134
Seção 3	Tipos de vinculação de folhas de estilo	141
3.1	Folha de estilo externa	141
3.2	Folha de estilo interna.....	141
3.3	Estilo inline	142
Seção 4	Principais propriedades de CSS	143
4.1	Unidades de medidas absolutas e relativas do CSS.....	143
4.2	Cores no CSS	143
4.3	Cor hexadecimal	144
4.4	Cor RGB	144
4.5	Cor RGBA	144
4.6	Cor HSL	144
4.7	Cor HSLA	145
4.8	Cor por palavra chave	145
4.9	Nomes de cores predefinidas/cross-browser	145
4.10	Fontes	145
4.11	Folhas de estilo aural	146
4.12	Principais propriedades do CSS.....	147
4.13	Básico da propriedade box	154
4.14	Propriedades de layouts flexíveis.....	161
4.15	Propriedade para textos	164
4.16	Propriedades para tabelas	169
4.17	Propriedades para listas	170

Apresentação

O que será estudado neste livro são os conceitos básicos de uma página web. Esse trabalho pode ser executado de forma bem tranquila, pois, para escrever uma página web, não precisamos de muita tecnologia, somente de um editor de textos simples, embora existam editores próprios para edição e criação de Hypertext Markup Language (HTML). Um documento HTML é basicamente formado por textos, mas tem funcionalidades e comandos para a inserção de vídeos, sons, imagens e animações.

O HTML é uma linguagem de marcação de hipertexto e é usada para criar sites. Pode ser traduzida para o português como Linguagem de Marcação de Hiper Texto.

Os documentos de Hypertext Markup Language estão em todos os momentos da navegação pela Internet. Em cada página navegada existe um código fonte que, após a interpretação do navegador, se transforma em uma linda página da web. O significado de web é, de origem inglesa, tela. No contexto virtual, interpreta-se como sendo uma tela virtual, que possui diversos documentos HTML, denominados páginas web, que são vinculados ou ligados por hiperligações.

Pense em uma navegação pela Internet. Você, usuário, digita na barra de endereços do navegador o site e o que acontece?

Seja qual for o navegador escolhido, este irá buscar pelo site digitado na barra de endereços, buscando por meio dos **servidores de DNS**. Os servidores DNS possuem a informação de qual computador da rede tem a informação solicitada, ou seja, o endereço em que está a informação, ou, ainda, o DNS descobre qual computador tem a informação. A partir desse momento, solicita uma cópia para a sua máquina e possibilita a visualização do documento. Essa comunicação entre os computadores é feita por meio de textos corridos, que utilizam basicamente a linguagem HTML.

O mais interessante no desenvolvimento de uma página de Internet é não precisarmos de muita tecnologia — com um simples editor de textos o programador é capaz de escrever e compilar todo o código fonte, já que o navegador consegue executar a tarefa sem necessitar de uma conexão com a Internet. Sugere-se que se faça um esboço do documento desejado antes de iniciar toda a codificação.

Conceitos básicos de html

Merris Mozer

Objetivos de aprendizagem: Conhecer o HTML ou Hypertext Markup Language (Linguagem de formatação de textos), que pode ser escrito em qualquer editor de texto. É uma linguagem de marcação de hipertexto usada para criar sites.

↳ Seção 1: **O que é uma página web**

Neste primeiro momento será apresentado o que é uma página web, o que é um browser, o que é um navegador, como são chamados os comandos do documento html, qual a estrutura básica de um texto html e como salvar um texto qualquer para que este seja um documento html.

↳ Seção 2: **Usando tags para melhorar um documento html**

Nesta seção você aprenderá a criar cabeçalhos, parágrafos, inserir quebras de linhas e algumas formatações para deixar seu documento com uma aparência agradável para o usuário.

Introdução ao estudo

Nesta parte inicial do livro serão trabalhados conceitos básicos para o desenvolvimento de páginas web com a utilização da Linguagem de formatação de textos. Uma página .html é desenvolvida a partir de um editor de texto bem simples; pode ser utilizado o bloco de notas somente salvando o arquivo com a extensão correta. E como toda linguagem o documento html possui comandos específicos chamados de tags; estes na maioria devem ser abertos e fechados, ou seja, são escritos em pares, para que ocorra o funcionamento correto. O fechamento é o próprio comando repetido precedido de uma barra. Porém, toda regra tem sua exceção e, nesse caso, destaca-se, por exemplo, o tag
 responsável pela quebra de linha.

No desenrolar dessas linhas serão demonstrados muitos exemplos de códigos usando os comandos básicos para o desenvolvimento de um documento html. Vamos lá!

Seção 1

O que é uma página web

Nesta seção serão abordados os passos para o desenvolvimento da sua primeira página na Internet, como são denominados os comandos dessa linguagem de marcação de textos, como salvar seu primeiro documento e qual sua estrutura básica. E ainda o que é browse e qual a sua função nesse contexto.

1.1 Você conhece e sabe o que é uma página web?

Esta seção envolve os conceitos básicos de uma página web. Nota-se que os documentos são basicamente formados por textos, embora possam abranger vídeos, sons, imagens e animações, como serão estudados ao longo de todo o livro. É um formato padrão utilizado para o texto e layouts de páginas para web.

Todo e qualquer documento possui uma extensão que determina a que tipo de arquivo pertence, por exemplo, uma imagem, um arquivo de som etc. Os documentos HTML normalmente possuem a extensão “.html” ou “.htm”. Cada linguagem de programação possui uma sintaxe própria, com seus comandos específicos; esses comandos denominamos de TAGS. Os TAGS são os comandos utilizados para escrever uma página HTML, ou seja, o conteúdo dessa página. Através desses comandos, o código fonte é criado como em qualquer outra linguagem. Os **browsers** (navegadores) são capazes de interpretar e executar todos os comandos que serão visualizados pelo usuário.

O código fonte de HTML pode ser escrito em qualquer editor de texto, desde os mais simples, como o Bloco de Notas do Windows, até os editores mais sofisticados, porém a forma mais rápida de se escrever um documento HTML é usando um editor próprio para o HTML, pois esses editores agilizam e facilitam o trabalho do desenvolvedor.

Como mencionado anteriormente, a base de um documento HTML é um TAG. Mas você já ouviu falar em TAG?

1.2 O que são tags

Antes que a página seja exibida, o browser interpreta o código fonte. Nesse momento, ele busca por caracteres especiais, ou seja, procura pelos TAGs. Busca pelos comandos que determinam as formatações e/ou as ações, por exemplo: Se um texto será exibido em negrito ou exibido em itálico, ou ainda se esse texto irá funcionar como um link para um outro arquivo ou outra página da Internet.

O TAG é iniciado pelo sinal de menor "<" e finalizado pelo sinal de maior ">", ficando desta maneira "<>". Os TAGs são etiquetas ou marcas padrões para fazer indicações a determinado browser; normalmente são usadas em pares, por exemplo:

```
<nome do TAG> ...t e x t o   a s e r   e x i b i d o... </ nome do TAG>

<b> Este texto será apresentado em negrito </b>
<h1> ..... </h1>
<h2> ..... </h2>
<h3> ..... </h3>
<h4> ..... </h4>
<h5> ..... </h5>
<h6> ..... </h6>
<p>Continue lendo a reportagem:</p>
<p><i>Continue lendo a reportagem</i></p>
<p><i><b>Continue lendo a reportagem: </b></i></p>
<p><i><b><blink>Continue lendo a reportagem:</blink></b></i></p>
<li>Vale a pena entrar nessa?</li>
<li>Pré-requisitos para a matrícula</li>
<i> Este texto será apresentado em itálico</i>

<ol type="I">
    <li>Se Passou no teste de aptidão</li>
    <li>Se NÂO Passou no teste de aptidão</li>
</ol>
```

Na maioria dos comandos de HTML se abre o TAG com sinal de menor e fecha com o sinal de maior, mas no fechamento deve-se incluir a barra "/", desta forma:
<> </>.

Segue outro exemplo:

```
abre <HTML>
fecha </HTML>
```

Mas existem alguns tags que não se fecham, por exemplo, o tag
:

- 1-Vale a pena entrar nessa!!!!

- 2-Pré-requisitos para a matrícula

- 3-Qualidade pedagógica

- 4-Rotina do aluno

- 5-Estrutura e corpo docente

- 6-Perspectivas para o formado

1.3 Estrutura básica de uma página html

Existem muitos TAGs para se escrever um documento de Hypertext Markup Language. O TAG principal de um documento HTML é: <HTML>.

Lembrando que os TAGs são iniciados com sinal de menor e finalizados com o sinal de maior, incluindo o sinal que indica uma barra "", desta forma: <> </>.

Observe o exemplo:

```
abre <HTML>
fecha </HTML>
```

Existem mais dois TAGs importantes na estrutura principal do documento HTML, os TAGs <head> e <body>.

O TAG <head>...</head> é responsável pela criação do cabeçalho.

O TAG <body>...</body> é responsável pela criação do corpo do documento.

```
abre <head>    fecha </head>
abre <body>    fecha </body>
```

A seguir, constam os TAGs básicos de qualquer documento HTML, ou seja, todo documento de marcação de texto deverá conter, no mínimo, os tags abaixo.

```
<HTML>
<head>
</head>

<body>
</body>
</HTML>
```

- └ HTML → que inicia o documento HTML; nela pode ser acrescentado o atributo lang, cuja finalidade é identificar a língua na qual o documento será escrito;
- └ Head → TAG responsável pela criação do cabeçalho, com as informações de cabeçalho não visíveis da página, mas na aba da página;
- └ Title → TAG onde se encontra o título da página;
- └ Body → é o corpo do documento, onde será inserido o conteúdo da página.

Por padrão, no momento de salvar um documento HTML, deve-se utilizar nomes em letras minúsculas, sem nenhum acento e não usar caracteres especiais.

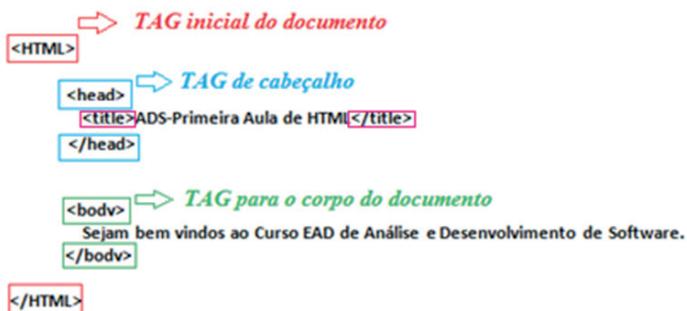
Utilizando essa estrutura é possível iniciar qualquer página HTML, desde uma bem simples, incluindo um título no TAG de <head> e um pequeno texto no corpo do documento no tag <body>, até as mais complexas.

Observe abaixo um pequeno exemplo de tudo o que foi mencionado sobre a estrutura básica de um documento de marcação de textos.

```
<HTML>
<head>
    <title> Primeiro exemplo de HTML </title>
</head>

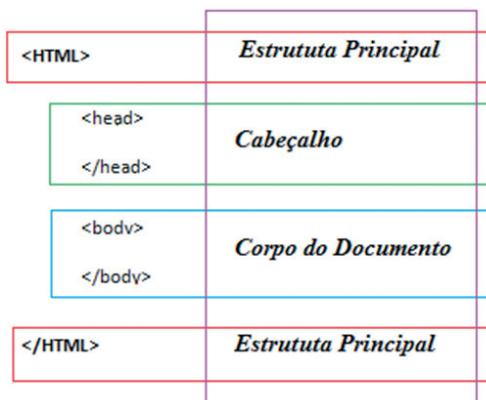
<body>
    Sejam bem-vindos ao Curso EAD de Análise e Desenvolvimento de Software.
</body>
</HTML>
```

Figura 1.1 Estrutura básica: 1



Fonte: Do autor (2014).

Figura 1.2 Estrutura básica: 2



Fonte: Do autor (2014).

1.4 O que é um browser

O primeiro **browser** surgiu em 1990, conhecido como World Wide Web. Não muito tempo depois foram surgindo outros navegadores, como o Internet Explorer, lançado em 1995, o Safari, lançado pela Apple em 2003, o navegador Mozilla Firefox, que está acompanhando os internautas desde 2004 e, mais recentemente, em 2008, lançado pelo Google no mundo tecnológico, o Google Chrome. Em suma, todos os browsers mencionados têm a mesma função, possibilitar às pessoas utilizarem a Internet, ou seja, a navegação na web. Embora a função seja basicamente a mesma, alguns desses navegadores são mais rápidos, ou são mais modernos ou possuem novas técnicas se comparados aos demais.

A palavra browser significa exatamente navegador, mas pode significar procurar, ou folheador, ou virar a página. A pronúncia correta é “bráuzer”.

A atividade de um browser é interpretar os códigos de várias linguagens, como HTML, ASP e PHP. Ele é o agente comunicador entre os servidores, onde desenvolve o processamento dos dados recebidos dos servidores da Internet e processa as respostas.

Atualmente, eles são capazes de interagir com interfaces bem desenvolvidas e bem coloridas para atender aos requisitos de interface entre o computador e o ser humano, mas, no passado, esse navegador só possuía textos e sem a velocidade permitida hoje.

Um navegador, também conhecido pelos termos em inglês web browser ou simplesmente browser, é um programa de computador que habilita seus usuários a interagirem com documentos virtuais da Internet, também conhecidos como páginas da web, que podem ser escritas em linguagens como HTML, ASP, PHP. (WIKIPEDIA, 2014).

1.5 Como salvar o documento html

Para que o browser consiga interpretar um documento e exibi-lo como uma página de Internet, ele deve ser salvo com a extensão “.html” ou “.htm”, por exemplo:

index.html

minha_primeira_pagina.html

arquivo_complementa.html

Quando existem várias páginas web, e todas elas estão linkadas, construiu-se um site. Onde a primeira página deve se chamar “index.html”.



Para saber mais

O tag <head>.....</head>, responsável pelo cabeçalho do documento, pode ser considerado opcional na programação HTML, isso somente se a programação não utilizar comandos que façam parte das linguagens de extensão do HTML, por exemplo, utilizar o CSS. No entanto, é interessante que esse tag esteja sempre presente para que se mantenha um padrão de desenvolvimento.



Para saber mais

Servidores de DNS = (do inglês *Domain Name System*). É o sistema de endereçamento da Internet que ajuda a encaminhar e-mails e a localizar os sites na web. Ou seja, é o método que converte o nome numérico do “site” e torna mais fácil a sua navegação. Por exemplo: 200.17.50.36 = www.ufpa.br.

Fonte: <<http://www.cultura.ufpa.br/dicas/net1/int-glo2.htm#dns>>.

Browsers = Internet Explorer, Firefox, Opera, ou...



Atividades de aprendizagem

- A partir dessa estrutura básica de um documento HTML mencionado nesta seção, crie um documento no bloco de notas com a estrutura básica mencionada acima e o salve com o seguinte nome: primeira_pg.html.
- Analise o documento abaixo e identifique as linhas que pertencem à estrutura básica de um documento HTML.

```

1) <html>
2) <head>
3) <title>EAD-HTML Aula nro 1C </title>
4) </head>
5) <body>
6)     <h1 align="center">Educação a distância vale a pena?</h1>
7)     <h2 align="center">Quem procura EAD ?</h2>
8)     <p>Para quem mora longe de uma universidade ou não pode ir
à aula todos os dias, a Educação a distância (EAD) parece ideal.
Por isso, ela tem conquistado tanto espaço. Em 2000, 13 cursos
superiores reuniam 1.758 alunos. Em 2008, havia 1.752 cursos de
graduação e pós-graduação lato sensu com 786.718 matriculados,
segundo a Associação Brasileira de Educação a Distância (Abed).
A modalidade de ensino usa ambientes virtuais, chats, fóruns e
e-mails para unir professores e turmas. Assim, quem é de Ribeirão
Cascalheiras, a 900 quilômetros de Cuiabá, por exemplo, pode se
formar em Pedagogia pela Universidade Federal do Mato Grosso
(UFMT), que mantém um polo na cidade.</p>
9)     <p>As experiências no ensino a distância por aqui começaram
no início do século 20, com cursos profissionalizantes por carta,
rádio e, mais tarde, pela TV. Só com a Internet e a banda larga,
eles se tornaram viáveis na graduação e na pós.</p>
10)    <p>Apenas recentemente começamos a apostar na EAD como
uma saída para suprir a demanda por formação superior no país.
Criada em 2005, a Universidade Aberta do Brasil (UAB) tem como
prioridade a formação inicial de professores da Educação Básica
pública, além de formação continuada aos graduados. Por meio de
parcerias entre 38 universidades federais, a UAB oferece 92 opções
de extensão, graduação e pós-graduação.</p>
11)    <p><i><b>Fonte:</b></i><a href="http://revistaescola.abril.com.br/formacao/formacao-inicial/vale-pena-entrar-nessa-educacao-distancia-diploma-prova-emprego-rotina-aluno-teleconferencia-chat-510862.shtml?utm_source=redesabril_fvc&utm_medium=facebook&utm_campaign=redesabril_novaescola" style="color: blue; text-decoration: none;"></a></p>
12)    <p><i><b>Continue lendo a reportagem:</b></i></p>
13)    1-Vale a pena entrar nessa?<br>
14)    2-Pré-requisitos para a matrícula<br>
15)    3-Qualidade pedagógica <tt>pedagógica</tt><br>
16)    4-Rotina do <blink>aluno</blink><br>
17)    5-<big>Estrutura</big> e corpo docente<br>
18)    6-<small>Perspectivas</small> para o formado<br>
19)    <p align="center">Tele-aula Professora: <br> Merris Mozer</p>
20)    </body>
21)</html>
```

3. Sabe-se que a maioria dos comandos de HTML, os tags, estão inseridos no tag do corpo do documento (<body> ... </body>), pois é nesse espaço que ocorrem todas as formatações de um documento HTML. Assim, baseado nessa informação e de todo o conteúdo estudado até o momento, onde o comando <title>..</title> deverá ser inserido na estrutura do documento HTML?



Questões para reflexão

Você conseguiu compreender o que é uma página de Internet e como é composta sua estrutura?

Seção 2

Usando tags para melhorar um documento html

Nesta seção serão apresentados alguns comandos básicos que melhoraram a aparência do seu documento web. Para que um documento possa ser visualizado, tudo será desenvolvido, nesse momento, no bloco de notas, para que você consiga entender a essência de uma página de Internet.

Como já se sabe, um TAG sempre se abre e se fecha; para criar um TAG de título é necessário o TAG `<title> </title>`.

```
<HTML>
  <head>
    <title>ADS-Primeira Aula de HTML</title>
  </head>

  <body>
    Sejam bem-vindos ao Curso EAD de Análise e Desenvolvimento de
    Software.
  </body>
</HTML>
```

A partir da estrutura básica serão acrescentados vários textos para a demonstração das formatações que se deseja apresentar nesse contexto.

2.1 Cabeçalho

Cabeçalhos são textos em tamanho maior, ou diferenciado, que dão semblante de título. Sabe-se que para cada ação, formatação, é necessário utilizar um comando específico; nesse caso de cabeçalho, o tag responsável é: `<h>.....</h>`, porém o comando "h" necessita que seja especificado o tamanho de apresentação do texto que compõe o cabeçalho. Assim, sempre que se utilizar esse **tag**, deverá ser apresentando um número, veja:

```
<h1> .... texto do cabeçalho ... </h1>
<h2> .... texto do cabeçalho ... </h2>
<h3> .... texto do cabeçalho ... </h3>
<h4> .... texto do cabeçalho ... </h4>
<h5> .... texto do cabeçalho ... </h5>
<h6> .... texto do cabeçalho ... </h6>
```

Esse número apresentado com a letra "h" é o número que determina o tamanho da linha para que se destaque com relação ao texto do documento. Quanto maior o número apresentado, menor será o tamanho da linha apresentada, ou seja, se o texto estiver `<h1> texto do cabeçalho ... </h1>`, a linha será visualizada num tamanho maior do que se fosse escrito `<h6> texto do cabeçalho ... </h6>`.

O texto a seguir irá demonstrar o tamanho da linha do cabeçalho, conforme mencionado no parágrafo anterior.

Exemplo — Formatação de cabeçalho 01 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>

  <body>
    <h1> Estudando formatação de cabeçalho "h1" </h1>
    <h2> Estudando formatação de cabeçalho "h2" </h2>
    <h3> Estudando formatação de cabeçalho "h3" </h3>
    <h4> Estudando formatação de cabeçalho "h4" </h4>
    <h5> Estudando formatação de cabeçalho "h5" </h5>
    <h6> Estudando formatação de cabeçalho "h6" </h6>
  </body>
</html>
```

Fonte: Do autor (2014).

Figura 1.3 Formatação de cabeçalho 01

Estudando formatação de cabeçalho "h1"

Estudando formatação de cabeçalho "h2"

Estudando formatação de cabeçalho "h3"

Estudando formatação de cabeçalho "h4"

Estudando formatação de cabeçalho "h5"

Estudando formatação de cabeçalho "h6"

Fonte: Do autor (2014).

Para reforçar seus estudos, copie o texto apresentado no bloco de notas, salve-o como "tag_h.html" e observe seu trabalho. Parabéns! Conseguiu visualizar o tamanho!

Exemplo — Formatação de cabeçalho 02 →

```
<HTML>
  <HEADER>
    <TITLE> ... PARABÉNS ... </TITLE>
  </HEADER>

  <BODY>
    <H1> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H1>
    <H2> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H2>
    <H3> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H3>
    <H4> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H4>
    <H5> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H5>
    <H6> PARABÉNS ALUNO POR PARTICIPAR DESTA GRANDE EQUIPE !!! </H6>
  </BODY>
</HTML>
```

Figura 1.4 Formatação de cabeçalho 02

Parabéns aluno por participar desta grande equipe !!!

Fonte: Do autor (2014).

2.2 Parágrafo e quebra de linhas

O tag utilizado para gerar parágrafo é o tag <p> e o tag utilizado para quebra de linha é
. Ambos são escritos no final da linha, para indicar o final do parágrafo ou onde deseja-se que seja feita uma quebra de linha. Uma coisa importante a saber é que, na digitação de um documento “.html”, a quebra de linha ou o início de um novo parágrafo não ocorre simplesmente pressionando a tecla “enter”, por esse motivo há a necessidade dos tags <p> e
.

Outra coisa importante a saber é a diferença básica entre parágrafo e quebra de linha. Quando se refere a uma quebra de linha, o cursor é posicionado imediatamente na próxima linha. Quando se refere ao parágrafo, o cursor sofre uma quebra de linha, porém existe um espaço entre a linha anterior e a próxima, ou seja, gera uma linha em branco, que inicia o novo parágrafo.

Exemplo — Formatação de parágrafos e quebra de linhas 01→

```
<html>
<header>
    <title>Tags Básicas/Formatação - Parágrafo</title>
</header>

<h1> .. Como formatar parágrafo e quebra de linha? .. </h1>
```

Para que o documento tenha quebra de linha ao final do parágrafo, ou seja, que sejam separados, utiliza-se o TAG.

Quando necessitar somente de uma quebra de linha sem dar espaço entre linhas pode ser utilizado o TAG muito utilizado para exibição de tópicos.

O TAG não requer seu fechamento como as demais.

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

No exemplo anterior, foi pressionado “Enter” em alguns momentos; a ideia é exemplificar que o “Enter” não irá funcionar. Analise o texto anterior e a imagem abaixo.

Figura 1.5 Estudando formatação de parágrafos e quebra de linha: 1**.. Como formatar parágrafo e quebra de linha ? ..**

Para que o documento tenha quebra de linha ao final do parágrafo, ou seja, que sejam separados utiliza-se o TAG. Quando necessitar somente de uma quebra de linha sem dar espaço entre linhas pode ser utilizado o TAG muito utilizado para exibição de tópicos. O TAG não requer seu fechamento como as demais.

Fonte: Do autor (2014).

Agora, utilizando os tags responsáveis por parágrafo e quebra de linha tem-se uma aparência agradável.

Exemplo — Formatação de parágrafos e quebra de linhas 02 →

```
<html>
<header>
    <title>Tags Básicas/Formatação - Parágrafo</title>
</header>

<h1> .. Como formatar parágrafo e quebra de linha ? .. </h1>

<p>
Para que o documento tenha quebra de linha ao final do parágrafo, ou
seja, que sejam separados, utiliza-se o TAG.
</p>

<p>
Quando necessitar somente de uma quebra de linha sem <br>
dar espaço entre linhas, pode ser utilizado o TAG <br>
muito utilizado para exibição de tópicos. <br>
O TAG não requer seu fechamento como as demais.<br>
</p>
</body>
</html>
```

Com a utilização dos tags **<p>** e **
**, a nova imagem do documento fica assim:

Figura 1.6 Estudando formatação de parágrafos e quebra de linha: 2**.. Como formatar parágrafo e quebra de linha ? ..**

Para que o documento tenha quebra de linha ao final do parágrafo, ou seja, que sejam separados, utiliza-se o TAG.

Quando necessitar somente de uma quebra de linha sem
dar espaço entre linha, pode ser utilizado o TAG
muito utilizado para exibição de tópicos.
O TAG não requer seu fechamento como as demais.

Fonte: Do autor (2014).

Exemplo — Formatação de parágrafos e quebra de linhas 03 →

Quando necessitar somente de uma quebra de linha sem dar espaço entre linhas,
pode ser utilizado o TAG
, muito utilizado para exibição de tópicos. O TAG

não requer seu fechamento como os demais.

Sem o tag
 e <p> têm-se os seguintes linha e código:

"Alguns professores do Primeiro Semestre Merris Mozer Anderson Macedo Sérgio Goes"

Certamente que o documento foi digitado da seguinte forma:

```
<html>
  <header>
    <title>Tags Básicas/Formatação - Parágrafo</title>
  </header>

  <h1> .. Alguns professores do Semestre ? .. </h1>

  Merris Mozer
  Anderson Macedo
  Sérgio Goes
  </body>
</html>
```

Mas a visualização não se comportou da maneira esperada. Então vamos digitá-lo utilizando os comando corretos.

```
<html>
  <header>
    <title>Tags Básicas/Formatação - Parágrafo</title>
  </header>

  <h1> .. Alguns professores do Semestre ? .. </h1>

  <p>
    Merris Mozer<br>
    Anderson Macedo<br>
    Sérgio Goes<br>
  </p>

  </body>
</html>
```

Figura 1.7 Estudando formatação de parágrafos e quebra de linha: 3

.. Alguns professores do Semestre ? ..

Merris Mozer
 Anderson Macedo
 Sérgio Goes

Fonte: Do autor (2004).

2.3 Formatando documento html

Existem comandos para definir estilos de textos, criando um aspecto agradável para a página web. Quando se utiliza um editor de textos é possível formatar um texto para negrito, itálico, sublinhado, dentre outras formatações, apenas selecionando o texto e clicando na opção desejada. Em um documento “.html”, essas tarefas de melhorar a aparência dos documentos são possíveis através de comandos específicos.

Essa etapa do trabalho inicia-se com formatações para destacar algumas palavras importantes do documento em questão:

Comando	Sintaxe	- Função
	 ... 	- negrito
<i>	<i> ... </i>	- itálico
<u>	<u> ... </u>	- sublinhado
<sup>	^{...}	- sobreescrito
<sub>	_{...}	- subscrito
<tt>	<tt> ... </tt>	- letras com o mesmo espaço (largura)
<big>	<big> ... </big>	- letras grandes
<small>	<small> ... </small>	- letras pequenas
<blink>	<blink> ... </blink>	- piscar (firefox)

Exemplo — Formatando documento 01 →

Observação com relação ao exemplo abaixo, pois nesse exemplo foi incluído um comando para possibilitar a inserção de comentários no corpo do texto, sendo:

```
<!--texto-->

<html>
    <header>
        <title> ... Melhorando a aparência do documento ... </title>
    </header>

    <body>
        <!-- o tag <b></b> formata um determinado texto para aparecer
        em negrito-->
        NEGRITO -->
        <b> ... Introdução ao Desenvolvimento WEB ... </b>
        <p>

        <!-- o tag <i></i> formata um determinado texto para aparecer
        em itálico-->
        ITÁLICO -->
        <i> ... Introdução ao Desenvolvimento WEB ... </i>
        <p>

        <!-- o tag <u></u> formata um determinado texto para aparecer
        em sublinhado-->
        SUBLINHADO -->
        <u> ... Introdução ao Desenvolvimento WEB ... </u>
        <p>
```

```

<!-- o tag <sup></sup> formata um determinado texto para
aparecer sobreescrito-->
SOBRESCRITO -->
TEXTO <sup> ... Introdução ao Desenvolvimento WEB ... </sup>
<p>

<!-- o tag <sub></sub> formata um determinado texto para
aparecer subscrito-->
SUBSCRITO -->
TEXTO <sub> ... Introdução ao Desenvolvimento WEB ... </sub>
<p>

<!-- o tag <tt></tt> formata um determinado texto para aparecer
letras com o mesmo espaço(largura)-->
LETRAS DA MESMA LARGURA -->
<tt> ... Introdução ao Desenvolvimento WEB ... </tt>
<p>

<!-- o tag <big></big> formata um determinado texto para
aparecer letras grandes-->
LETRAS GRANDES -->
<big> ... Introdução ao Desenvolvimento WEB ... </big>
<p>

<!-- o tag <small></small> formata um determinado texto para
aparecer letras pequenas-->
LETRAS PEQUENAS -->
<small> ... Introdução ao Desenvolvimento WEB ... </small>
<p>

<!-- o tag <blink></blink> formata um determinado texto para
aparecer o texto piscando (firefox)-->
TEXTO PISCANDO -->
<blink> ... Introdução ao Desenvolvimento WEB ... </blink>
<p>
</body>
</html>

```

Para colocar a palavra em negrito (bold), usamos o tag ****; dessa forma, tudo o que estiver entre o TAG **** ficará em negrito.

Quando desejar destacar determinado texto e formatá-lo para itálico, utiliza-se o TAG *<i></i>*.

Para exemplificar pode-se alterar a linha do documento:

DE: <p>Professores do Primeiro Semestre</p>

Primeiramente formatar para negrito:

PARA: <p>Professores do Primeiro Semestre</p>

Para se ter letras com a mesma largura, como a fonte "courier new", utiliza-se o TAG **<tt></tt>**. Pode ser usado para diferenciar uma palavra das demais e/ou para demonstrar códigos de programas como no *Exemplo — Formatando documento 01*.

O código acima resulta na imagem a seguir, em que é possível verificar os efeitos dos comandos descritos.

Figura 1.8 Formatando documento html

NEGRITO --> ... **Introdução ao Desenvolvimento WEB** ...

ITÁLICO --> ... *Introdução ao Desenvolvimento WEB* ...

SUBLINHADO --> ... Introdução ao Desenvolvimento WEB ...

SOBRESCRITO --> TEXTO ... ^{Introdução ao Desenvolvimento WEB} ...

SUBSCRITO --> TEXTO ... _{Introdução ao Desenvolvimento WEB} ...

LETRAS DA MESMA LARGURA --> ... **Introdução ao Desenvolvimento WEB** ...

LETRAS GRANDES --> ... **Introdução ao Desenvolvimento WEB** ...

LETRAS PEQUENAS --> ... **Introdução ao Desenvolvimento WEB** ...

TEXTO PISCANDO --> ... **Introdução ao Desenvolvimento WEB** ...

Fonte: Do autor (2014).

2.4 Textos pré-formatados

Os textos pré-formatados são construídos a partir da utilização do comando <pre>. Esse comando facilita a digitação de qualquer documento, como é feito em editores de textos simples. Respeitando o final de linha originado pela tecla “enter”, respeitando inclusive as tabulações.

A sintaxe para usar esse tag é:

```
<pre>
    Colocar aqui o texto desejado
</pre>
```

Exemplo — Formatação de textos pré-formatados 01 →

```
<html>
    <header>
        <title> ... Texto pre-formatados ... </title>
    </header>
    <body>
        <pre>
            O tag PRE respeita o enter, (pressionando a tecla enter)
            neste momento vamos teclar enter novamente (pressionando
            a tecla enter)
            após o enter (pressionando a tecla enter)
            funciona.
            (pressionando a tecla enter)
            (pressionando a tecla enter)
            (pressionando a tecla enter)
            (pressionando a tecla enter)
        </pre>
    </body>
</html>
```

O código anterior resulta na visualização da tela abaixo:

Figura 1.9 Textos pré-formatados: 1

O tag PRE respeita o enter, (pressionando o enter) neste momento vamos teclar enter novamente (pressionando o enter) após o enter (pressionado o enter) funciona.

```
(pressionando o enter)
(pressionando o enter)
(pressionando o enter)
(pressionando o enter)
```

Fonte: Do autor (2014).

Exemplo — Formatação de textos pré-formatados 02 →

```
<html>
  <header> <title> ... Texto pre-formatados ... </title></header>
  <body>
    <pre>
      Para quem mora longe de uma universidade
      (pressionando a tecla enter)
      ou não pode ir à aula todos os dias, a Educação a distância (EAD)
      (pressionando a tecla enter)
      parece ideal. Por isso, ela tem conquistado tanto espaço.
      (pressionando a tecla enter)
      Em 2000, 13 cursos superiores reuniam 1.758 alunos.
      (pressionando o enter)
      Em 2008, havia 1.752 cursos de graduação e
      (pressionando a tecla enter)
      pós-graduação lato sensu com 786.718 matriculados,
      (pressionando a tecla enter)
      segundo a Associação Brasileira de Educação a Distância (Abed).
      (pressionando a tecla enter)
      A modalidade de ensino usa ambientes virtuais, chats,
      (pressionando a tecla enter)
      fóruns e e-mails para unir professores e turmas.
      (pressionando a tecla enter)
      Assim, quem é de Ribeirão Cascalheiras,
      (pressionando a tecla enter)
      a 900 quilômetros de Cuiabá, por exemplo,
      (pressionando a tecla enter)
      pode se formar em Pedagogia pela Universidade
      (pressionando a tecla enter)
      Federal do Mato Grosso (UFMT), que mantém um polo na cidade.
      (pressionando a tecla enter)
      Fonte:http://revistaescola.abril.com.br/formacao/formacao-inicial/vale-pena-entrar-nessa-educacao-distancia-diploma-prova-emprego-rotina-aluno-teleconferencia-chat-510862.shtml?utm\_source=redesabril\_fvc&utm\_medium=facebook&utm\_campaign=redesabril\_novaescola&
    </pre>
  </body>
</html>
```

O código anterior resulta na vivsualização da tela abaixo:

Figura 1.10 Textos pré-formatados: 1

Para quem mora longe de uma universidade (pressionando o enter) ou não pode ir à aula todos os dias, a Educação a distância (EAD) (pressionando o enter) parece ideal. Por isso, ela tem conquistado tanto espaço. (pressionando o enter) Em 2000, 13 cursos superiores reuniam 1.758 alunos. (pressionando o enter) Em 2008, havia 1.752 cursos de graduação e (pressionando o enter) pós-graduação lato sensu com 786.718 matriculados, (pressionando o enter) segundo a Associação Brasileira de Educação a Distância (Abed). (pressionando o enter)

A modalidade de ensino usa ambientes virtuais, chats, (pressionando o enter) fóruns e e-mails para unir professores e turmas. (pressionando o enter) Assim, quem é de Ribeirão Cascalheiras, (pressionando o enter) a 900 quilômetros de Cuiabá, por exemplo, (pressionando o enter) pode se formar em Pedagogia pela Universidade (pressionando o enter) Federal do Mato Grosso (UFMT), que mantém um polo na cidade. (pressionando o enter) Fonte: <http://revistaescola.abril.com.br/formacao/formacao-inicial/vale-pena-entrar-nessa>

Fonte: Do autor (2014).

2.5 Formatando fonte

Considera-se dos mais importantes comandos da linguagem de desenvolvimento da páginas web sendo o tag ` ... `, pois somente com a utilização dele é possível formatar o tamanho do texto, o tipo do texto (nome da fonte) e a cor desse texto.

A sintaxe desse tag é:

```
<font>
  Size = n
  Face = nome da fonte
  Color = cor
</font>
```

Para seu conhecimento, no parâmetro `size=n`, o “n” representa o tamanho da fonte que será atribuído. O tamanho padrão utilizado em documentos da web é de número 3, mas o desenvolvedor pode atribuir o que melhor lhe convier, desde que o size encontre-se entre 1 e 7.

O parâmetro “`face`” indica a fonte a ser atribuída ao texto, e poderá ser definido mais de um nome de fonte. O browser buscará a primeira fonte mencionada; caso esta não esteja instalada no computador, será atribuída a seguinte, e assim respectivamente. Embora a fonte padrão usada dos documentos da Internet seja Times New Romam.

Finalizando o tag `font`, faz-se necessário comentar sobre o parâmetro “`color`”; este é o responsável por definir a cor do texto. A cor pode ser definida pelo seu nome (“red”, “blue”, “White”, “Black”, ...) ou escrita em sua forma hexadecimal, por exemplo: `color="#008800"`.

Observe o exemplo a seguir, onde se encontra um cabeçalho sem formação de cor para comparação com o próximo exemplo.

Exemplo — Formatação de fonte 01 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>
  <body>
    <h1> Estudando formatação de cores "font" </h1>
  </body>
</html>
```

O código acima resulta na visualização da tela abaixo:

Figura 1.11 Formatando fonte: 1

Estudando formatação de cores "font"

Fonte: Do autor (2014).

Agora, o exemplo a seguir refere-se ao cabeçalho formatado, usando a cor azul, escrevendo o nome da cor.

Exemplo — Formatação de fonte 02 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>
  <body>
    <h1>
      <font color="blue">
        Estudando formatação de cores "font"
      </font>
    </h1>
  </body>
</html>
```

O código acima resulta na visualização da tela abaixo:

Figura 1.12 Textos pré-formatados: 1

Estudando formatação de cores "font"

Fonte: Do autor (2014).

Exemplo — Formatação de fonte 03 →

Alterando a linha onde se define a cor verde com a utilização de um código hexadecimal:

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>
  <body>
    <h1>
      <font color="#008800">
        Estudando formatação de cores "font"
      </font>
    </h1>
  </body>
</html>
```

Exemplo — Formatação de fonte 04 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>
  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
        Estudando formatação de cores "font"
      </font>
    </h1>
  </body>
</html>
```

É muito interessante a utilização do tag “font”, pois ele possibilita a formatação de somente parte do texto ou permite formatar somente uma palavra do texto.

Figura 1.13 Textos pré-formatados: 1

Estudando formatação de cores "font"

Fonte: Do autor (2014).

A Saber: No parâmetro “face” não diferencia se for utilizado o nome da cor ou seu código hexadecimal.

Segue uma pequena tabela com algumas cores e seus respectivos códigos hexadecimais.

Figura 1.14 Cores com nomes atribuídos oficialmente

Aqua (#00FFFF)	Black (#000000)	Blue (#0000FF)	Fuchsia (#FF00FF)
Green (#008000)	Gray (#808080)	Lime (#00FF00)	Maroon (#800000)
Navy (#000080)	Olive (#808000)	Purple (#800080)	Red (#FF0000)
Silver (#C0C0C0)	Teal (#008080)	White (#FFFFFF)	Yellow (#FFFF00)

Ainda formatando o documento, trabalhar com cores é uma tarefa muito fácil com HTML; para facilitar o processo de descobrir os códigos de cores existe a ferramenta livre denominada cpick.exe. Será tratado mais sobre esse programa no tópico Cor de Fundo.

No tratamento de cores no documento “.html” pode-se mudar o tom da cor de fundo, por exemplo, o tag `<body bgcolor="blue">` pode-se estabelecer um azul mais

claro usando: <body bgcolor="Lightblue"> e pode-se estabelecer um azul mais escuro usando: <body bgcolor="darkblue">.

2.6 Linha de separação

Como o próprio nome sugere, neste item trata-se de uma linha de separação, cuja função é melhorar a aparência do documento, para fazer a separação de objetos, fotos e textos dentro da página web.

O comando responsável por essa linha de separação é o tag <hr>, que pode ser incluído a qualquer momento no corpo do texto.

Exemplo — Formatando linha de separação 01 →

```
<html>
  <header>
    <title> ... Tags Básica - linha de separação ... </title>
  </header>

  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
        <hr>
        Estudando o tag LINHA DE SEPARAÇÃO
        <hr>
      </font>
    </h1>
  </body>
```

O código acima resulta na visualização da tela abaixo:

Figura 1.15 Linha de separação: 1

Estudando o tag LINHA DE SEPARAÇÃO

Fonte: Do autor (2014).

Essa linha de separação apresentada no exemplo anterior pode ser formatada, alterando a sua espessura, sua cor e seu tamanho. Para que essas alterações ocorram deve-se incluir dentro do tag “<hr>” alguns parâmetros, a seguir:

- └ <hr size=1>, o parâmetro size é responsável por definir a espessura da linha, permitindo que essa linha seja mais grossa ou mais fina, de acordo com o valor estipulado pelo número que acompanha o parâmetro. Lembrando que esse número refere-se a pixels.

Exemplo — Formatando linha de separação 02 →

```
<html>
  <header>
    <title> ... Tags Básica - linha de separação ... </title>
  </header>

  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
      <hr size=5>
      Estudando o tag LINHA DE SEPARAÇÃO
      <hr size=5>
      </font>
    </h1>
  </body>
```

O código acima resulta na visualização da tela abaixo:

Figura 1.16 Linha de separação: 2

Estudando o tag LINHA DE SEPARAÇÃO

Fonte: Do autor (2014).

- └ <hr color=blue>, o parâmetro color define a cor de apresentação da linha de separação.

Exemplo — Formatando linha de separação 02 →

```
<html>
  <header>
    <title> ... Tags Básica - linha de separação ... </title>
  </header>

  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
      <hr color=blue>
      Estudando o tag LINHA DE SEPARAÇÃO
      <hr color=blue >
      </font>
    </h1>
  </body>
```

O código acima resulta na visualização da tela abaixo:

Figura 1.17 Linha de separação: 3

Estudando o tag LINHA DE SEPARAÇÃO

Fonte: Do autor (2014).

- └ **<hr with=300> ou <hr with=30%>**, estipula o comprimento da linha, sendo representado em pixels ou em porcentagem. Quando utilizada a opção em pixels, o comprimento da linha será um tamanho fixo, porém quando especificado em porcentagem o comprimento da linha será sempre de acordo com a variação do tamanho da janela.

Exemplo — Formatando linha de separação 02 →

```

<html>
  <header>
    <title> ... Tags Básica - linha de separação ... </title>
  </header>

  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
        <hr width=600>
        Estudando o tag LINHA DE SEPARAÇÃO
        <hr width=600>
      </font>
    </h1>
  </body>

```

O código anterior resulta na visualização da tela abaixo:

Figura 1.18 Linha de separação: 4

Estudando o tag LINHA DE SEPARAÇÃO

Fonte: Do autor (2014).

Os tags utilizados para a criação de documentos “.html” podem receber vários parâmetros de uma única vez, sendo muito importante para que a estética e o design fiquem agradáveis aos olhos dos usuários que irão navegar pelo site.

```

<html>
  <header>
    <title> ... Tags Básicas/Formatação cabeçalho ... </title>
  </header>
  <body>
    <h1>
      <font size=5 color="red" face="Arial Black" >
        <hr size=6 color=red width=550 align="left">
        Estudando o tag LINHA DE SEPARAÇÃO
        <hr size=6 color=red width=550 align="left">
      </font>
    </h1>
  </body>
</html>

```

Figura 1.19 Linha de separação: 5

Estudando o tag LINHA DE SEPARAÇÃO

Fonte: Do autor (2014).

O parâmetro align será estudo ainda nesta seção.

Para utilizar a opção de porcentagem no parâmetro width basta fazer a seguinte mudança:

DE: **<hr size=6 color=red width=550>**

PARA: **<hr size=6 color=red width=40%>**

Dessa forma, a linha sempre será redimensionada de acordo com o tamanho que a janela do browser tiver.

2.7 Alinhamentos

Em qualquer editor de textos é possível escrever o documento com parágrafos alinhados para a esquerda, ou alinhados para a direita, e ainda alinhados de maneira centralizada entre as margens esquerda e direita. Na criação de um documento “.html” é possível fazer esses alinhamentos utilizando o parâmetro “align”.

O parâmetro align pode ser escrito em diversos momentos dentro do documento. Pode ser inserido no tag “hn” responsável pelo cabeçalho; pode ser inserido no corpo do texto; pode ser especificado dentro do tag “p” responsável pelo parágrafo.

Existem as opções de alinhar o texto de forma centralizada, alinhar o texto à direita e alinhar o texto à esquerda.

- └ align="center"
- └ align="right"
- └ align="left"
- └ align="justify"

Exemplo — Formatando alinhamento do documento 01 →

```

<html>
  <header>
    <title> ... Tags Básicas/Formatação alinhamento... </title>
  </header>

  <body>
    <h1 align="center">
      Estudando alinhamento CENTRALIZADO
    </body>
  </html>

```

Figura 1.20 Alinhamentos: centralizado

Estudando alinhamento CENTRALIZADO

Fonte: Do autor (2014).

Exemplo — Formatando alinhamento do documento 02 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação alinhamento... </title>
  </header>

  <body>
    <h1 align="right">
      Estudando alinhamento À DIREITA
    </body>
</html>
```

Figura 1.21 Alinhamentos: à direita

Estudando alinhamento À DIREITA

Fonte: Do autor (2014).

Exemplo — Formatando alinhamento do documento 03 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação alinhamento... </title>
  </header>

  <body>
    <h1 align="left">
      Estudando alinhamento À ESQUERDA
    </body>
</html>
```

Figura 1.22 Alinhamento à esquerda

Estudando alinhamento À ESQUERDA

Fonte: Do autor (2014).

Exemplo — Formatando alinhamento do documento 04 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação alinhamento... </title>
  </header>

  <body>
    <h1 align="justify">
      Para quem mora longe de uma universidade ou não pode ir à aula todos os dias, a Educação a distância (EAD) parece ideal. Por isso, ela tem conquistado tanto espaço. Em 2000, 13 cursos superiores reuniam 1.758 alunos. Em 2008, havia 1.752 cursos de graduação e pós-graduação lato sensu com 786.718 matriculados, segundo a Associação Brasileira de Educação a Distância (Abed).
    </h1>
  </body>
</html>
```

Figura 1.23 Formatando alinhamento de documento: 1

Para quem mora longe de uma universidade ou não pode ir à aula todos os dias, a Educação a distância (EAD) parece ideal. Por isso, ela tem conquistado tanto espaço. Em 2000, 13 cursos superiores reuniam 1.758 alunos. Em 2008, havia 1.752 cursos de graduação e pós-graduação lato sensu com 786.718 matriculados, segundo a Associação Brasileira de Educação a Distância (Abed).

Fonte: Do autor (2014).

No exemplo a seguir será mostrado um texto que se refere à Anatel, publicado em 19/03/2014. No texto, cada parágrafo foi formatado de maneira diferente, onde se nota a possibilidade de desenvolvimento dentro do documento “.html”.

O desenvolvedor usa sua criatividade e faz acontecer.

Exemplo — Formatando alinhamento do documento 05 →

```
<html>
  <header>
    <title> ... Tags Básicas/Formatação alinhamento... </title>
  </header>

  <body>
    <p align="justify">
      Após a repercussão e polêmica criadas com a divulgação da informação de que, a partir da última segunda-feira (17), a Anatel estaria monitorando e pretende bloquear aparelhos não homologados, a Agência Nacional de Telecomunicações emitiu uma nota oficial para esclarecer o projeto.
    </p>

    <p align="center">
      Brasil é o quarto país mais conectado da América Latina, revela pesquisa.
    </p>
  </body>
</html>
```

<p align="left">

O Sistema Integrado de Gestão de Aparelhos (Siga) é uma realidade, e visa mesmo fazer com que o número de aparelhos não homologados pela Anatel diminua no Brasil. A Anatel explica, entretanto, que ainda não definiu qual será a punição para quem utiliza esses gadgets e nem quando o programa vai entrar em funcionamento de fato.

</p>

<p align="right">

O comunicado confirma que a primeira fase do Siga, que é gerido pelas operadoras, está em andamento, mas em fase experimental. O projeto realmente começa realizando "um diagnóstico sobre a regularidade dos aparelhos conectados às redes das prestadoras", conforme divulgado na última segunda. O que vem depois disso ainda não foi decidido.

</p>

<p align="right">

Fonte:<http://www.techtudo.com.br/noticias/noticia/2014/03/anatel-esclarece-seu-projeto-de-bloqueio-de-celulares-piratas.html>

</p>

</body>

</html>

Figura 1.24 Formatando alinhamento de documento: 2

Após a repercussão e polêmica criadas com a divulgação da informação de que, a partir da última segunda-feira (17), a Anatel estaria monitorando e pretendendo bloquear aparelhos não homologados, a Agência Nacional de Telecomunicações emitiu uma nota oficial para esclarecer o projeto.

Brasil é o quarto país mais conectado da América Latina, revela pesquisa.

O Sistema Integrado de Gestão de Aparelhos (Siga) é uma realidade, e visa mesmo fazer com que o número de aparelhos não homologados pela Anatel diminua no Brasil. A Anatel explica, entretanto, que ainda não definiu qual será a punição para quem utiliza esses gadgets e nem quando o programa vai entrar em funcionamento de fato.

O comunicado confirma que a primeira fase do Siga, que é gerido pelas operadoras, está em andamento, mas em fase experimental. O projeto realmente começa realizando "um diagnóstico sobre a regularidade dos aparelhos conectados às redes das prestadoras", conforme divulgado na última segunda. O que vem depois disso ainda não foi decidido.

Fonte:<http://www.techtudo.com.br/noticias/noticia/2014/03/anatel-esclarece-seu-projeto-de-bloqueio-de-celulares-piratas.html>

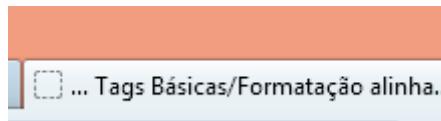
Fonte: Do autor (2014).

2.8 Títulos

Primeiro ponto a ser abordado é diferenciar título de cabeçalho.

Título é um texto exibido na aba do browser.

Figura 1.25 Título



Fonte: Do autor (2014).

Cabeçalho é um texto exibido no corpo do documento, usando o tag <hn> contido visto no início desta seção.

Figura 1.26 Cabeçalho



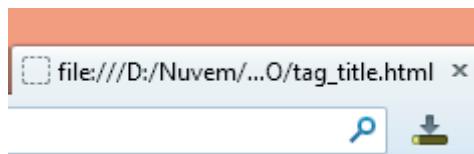
Fonte: Do autor (2014).

Para a inclusão do título em um documento é necessário a utilização do tag <title>.....</title>. Esse comando deve estar sempre declarado entre os tags <head>.....</head>, que fazem parte do escopo principal do texto.

Exemplo — Formatando título do documento 01 →

Este exemplo não terá o tag <title>.....</title> definido.

```
<html>
  <header>
  </header>
  <body>
    <h1> Parabéns aluno por participar desta grande equipe !!! </h1>
    <h2> Parabéns aluno por participar desta grande equipe !!! </h2>
    <h3> Parabéns aluno por participar desta grande equipe !!! </h3>
    <h4> Parabéns aluno por participar desta grande equipe !!! </h4>
    <h5> Parabéns aluno por participar desta grande equipe !!! </h5>
    <h6> Parabéns aluno por participar desta grande equipe !!! </h6>
  </body>
</html>
```

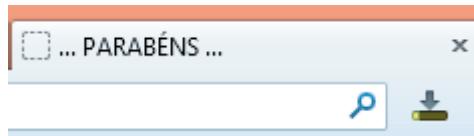
Figura 1.27 Ilustrativa da tela do Windows: 1

Fonte: Do autor (2014).

Exemplo — Formatando título do documento 02 →

Este exemplo terá o tag <title>.....</title> definido.

```
<html>
  <header>
    <title> ... PARABÉNS ... </title>
  </header>
  <body>
    <h1> Parabéns aluno por participar desta grande equipe !!! </h1>
    <h2> Parabéns aluno por participar desta grande equipe !!! </h2>
    <h3> Parabéns aluno por participar desta grande equipe !!! </h3>
    <h4> Parabéns aluno por participar desta grande equipe !!! </h4>
    <h5> Parabéns aluno por participar desta grande equipe !!! </h5>
    <h6> Parabéns aluno por participar desta grande equipe !!! </h6>
  </body>
</html>
```

Figura 1.28 Ilustrativa da tela do Windows: 2

Fonte: Do autor (2014).

2.9 Cor de fundo

Todo documento web pode ter uma cor especificada pelo desenvolvedor para o fundo da tela. Para definir uma cor de fundo ou uma imagem de fundo à página web deve-se configurar um parâmetro no tag <body>...</body>. Para relembrar, os parâmetros são características que alteram o comportamento de um tag; como mencionado anteriormente no tag de parágrafo <p>, pode-se incluir o parâmetro align="center" para centralizar o conteúdo do tag, ou incluir o parâmetro align="right" para alinhar à direita o texto.

No tag <body>....</body> inclui-se o parâmetro background para especificar a cor desejada de fundo do texto.

`bgcolor="cor desejada"`

Exemplo — Formatando cor de fundo 01 →

```
<html>
  <header>
    <title> ... Cor de fundo é amarela... </title>
  </header>

  <body bgcolor="yellow">
    <h1> Cor de fundo é amarela !!! </h1>
    <h2> Cor de fundo é amarela !!! </h2>
    <h3> Cor de fundo é amarela !!! </h3>
    <h4> Cor de fundo é amarela !!! </h4>
    <h5> Cor de fundo é amarela !!! </h5>
    <h6> Cor de fundo é amarela !!! </h6>
  </body>
</html>
```

Figura 1.29 Formatando a cor de fundo: 1



Cor de fundo é amarela !!!

Fonte: Do autor (2014).

Na linha de código <body bgcolor="yellow">, onde se define a cor de fundo do documento, pode ser utilizado o código hexadecimal da cor desejada, por exemplo: <body bgcolor="#008800"> que se refere à cor verde.

Exemplo — Formatando cor de fundo 02 →

O exemplo abaixo mostra a cor sendo atribuída utilizando o código hexadecimal num tom mais forte que o amarelo apresentado no “Exemplo — Formatando cor de fundo 02”.

```

<html>
<head>
    <title>Locadora Alunos UNOPAR</title>
</head>

<body bgcolor="#F0C828" text="#ffffff">
    <h1 align="center"><font color="green">Para a realização das
    atividades no decorrer do curso, leiam com muita atenção o estudo
    de caso descrito no cenário “Locadora Alunos UNOPAR”.</font></h1>

</body>
</html>

```

Figura 1.30 Formatando a cor de fundo: 2

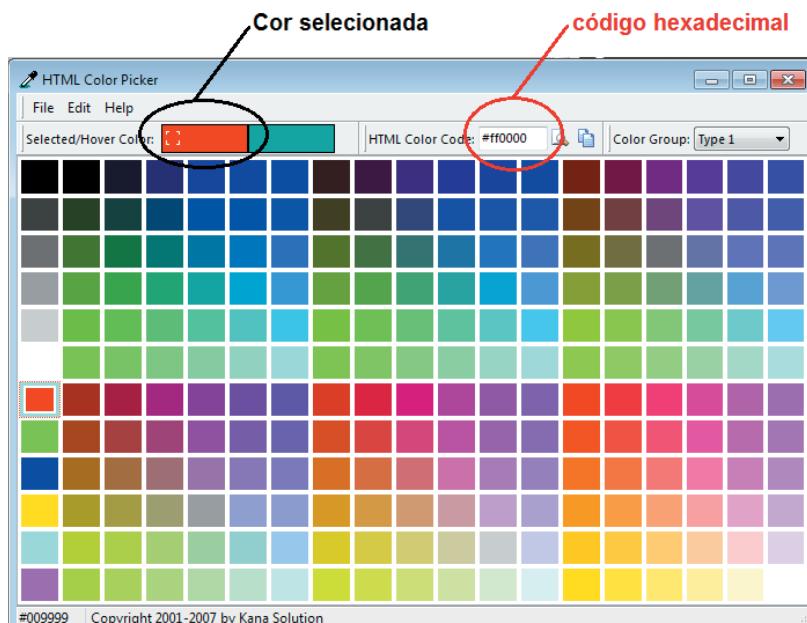
**Para a realização das atividades no decorrer do
curso, leiam com muita atenção o estudo de caso
descrito no cenário “Locadora Alunos UNOPAR”.**

Fonte: Do autor (2014).

Como descobrir o código hexadecimal de uma cor utilizando o cpick.exe?

O programa cpick.exe é um programa livre, de fácil utilização, para descobrir os códigos hexadecimais. Por exemplo, a cor branca = white tem o código #fffff; a cor amarela = yellow: #ffff00.

Figura 1.31 Ilustrativa da tela do próprio cpick.exe



Fonte: Do autor (2014).

Mas se ocorrer uma situação onde seja necessário descobrir o código hexadecimal e uma cor qualquer, e nesse momento o computador a ser usado não tenha o programa cpick.exe instalado? Como resolver essa situação?

Muito fácil!

Com o auxílio do programa paint e da calculadora disponível no próprio Windows, a tarefa está quase concluída. No programa Paint, na opção editar cores:

Figura 1.32 Ilustrativa da tela do próprio Paint



Fonte: Do autor (2014).

O usuário escolhe a cor desejada; após a seleção, observa os campos que mostram a quantidade de vermelho, a quantidade de verde e a quantidade de cor azul, conforme a figura a seguir:

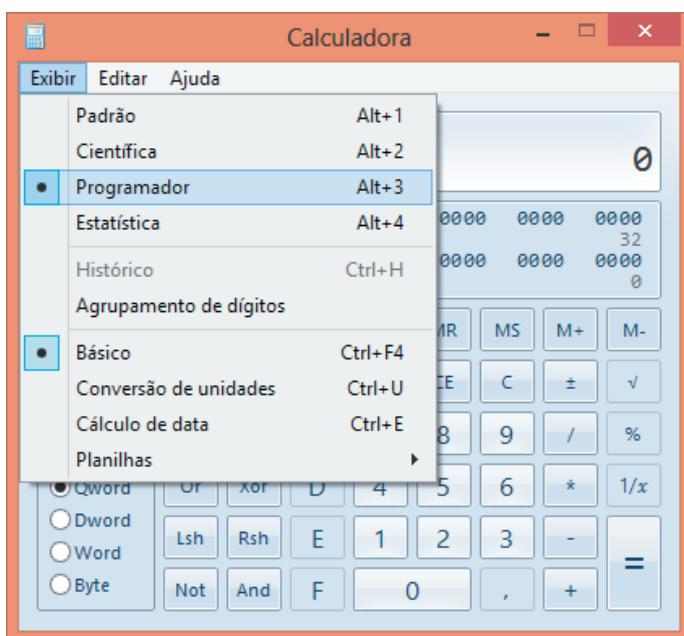
Figura 1.33 Ilustrativa da tela do próprio Paint



Fonte: Do autor (2014).

Nesse momento deve-se abrir a calculadora do próprio Windows e selecionar a opção de programador:

Figura 1.34 Ilustrativa da tela da própria calculadora do Windows



Fonte: Do autor (2014).

Digitar cada um dos códigos encontrados nos campos de vermelho, azul e verde com a opção DECIMAL selecionada, conforme imagem ilustrativa da própria calculadora, abaixo:

Figura 1.35 Ilustrativa da tela da própria calculadora do Windows



Fonte: Do autor (2014).

Na sequência, deve-se selecionar a opção HEXADECIMAL; quando se faz essa seleção automaticamente já aparece o código 234 na sua forma hexadecimal, conforme imagem ilustrativa da própria calculadora, abaixo:

Figura 1.36 Ilustrativa da tela da própria calculadora do Windows



Fonte: Do autor (2014).

A seguir, uma breve explicação de como é composto o código de cores na sua forma hexadecimal ("#XXXXXX"). Este é composto de:

- └ uma tralha ou jogo da velha;
- └ dois dígitos que serão a quantidade de vermelho;
- └ os próximos dois dígitos serão a quantidade de verde;
- └ e os últimos dois dígitos serão a quantidade de azul.

Esse código encontra-se no padrão RGB, que é o mesmo utilizado em: monitores, televisões, qualquer aparelho que exiba imagens etc.

Exemplo — Formatando cor 03 →

#0000FF, no exemplo apresentado, diz que não possui nada de cor vermelho e nada de cor verde e possui o máximo de azul.



Para saber mais

Os comandos em ".html" normalmente necessitam do fechamento do tag; diferentemente dos demais; os tags
 e o tag <p> não precisam ser fechados para funcionar.

Destaques, negritos, diálogos

- └ tag refere-se aos comando específicos do documento HTML;
- └ cabeçalho é um texto exibido no corpo do documento, usando o tag <hn> — conteúdo visto no início desta seção;

- ↳ título é um texto exibido na aba do browser;
- ↳ size, tamanho da fonte;
- ↳ face, indica o nome da fonte.



Atividades de aprendizagem

1. Efetuar uma pesquisa para descobrir qual é o outro comando que pode ser utilizado para colocar em negrito determinado texto do documento.
2. Utilizando a tabela abaixo, transforme as cores vermelho, verde e azul para o código hexadecimal.

Vermelho	Verde	Azul	Hexadecimal
255	100	20	
240	200	40	
120	70	120	
40	210	240	
20	150	255	



Para saber mais

Lembre-se de que com o comando `
` o texto somente passa para a linha de baixo ou para a próxima linha, sem gerar nenhuma linha em branco, como ocorre quando se usa o comando `<p>`. Fica muito fácil lembrar relacionando-o a seu significado:

- ↳ `<p>` = parágrafo = gera linha em branco.
- ↳ `
` = break = quebra = quebra linha.



Questões para reflexão

Você concorda que existe a possibilidade de depois da formatação de um texto para negrito também formatá-lo para itálico?



Fique ligado!

No decorrer desta unidade foram abordados os seguintes temas:

- conceito de página web;
- conceito de tags;
- estrutura básica de uma página HTML;
- conceito de browser;
- formas de salvamento de documento HTML;
- uso das tags em documentos HTML;
- formatação de documento HTML.



Para concluir o estudo da unidade

Chegamos ao final desta unidade. Parabéns a você que concluiu os estudos referentes à Unidade 1 deste livro, mas para que você possa aprofundar seus conhecimentos ainda é importante muito treino. Recomendo que consulte outros livros que abordam o tema HTML. Enfim, “a felicidade não está no fim da jornada, e sim em cada curva do caminho que percorremos para encontrá-la!” (FRAZZ, 2014).



Atividades de aprendizagem da unidade

1. Faça uma pesquisa na Internet sobre o que significa HTML e para que serve essa linguagem?
2. Analise o documento abaixo e faça uma relação dos tag encontrados com suas respectivas funções.

```
<html>
<head>
    <title>Locadora Alunos UNOPAR</title>
</head>

<body bgcolor="#F0C828" text="#ffffff">
    <h1 align="center">
        <font color="green">
            Para a realização das atividades no decorrer do curso,
            leiam com muita atenção o estudo de caso descrito no
            cenário "Locadora Alunos UNOPAR".
```

```
</font>
</h1>

<h3 align="center"><font color="#ffff00">Cenário Proposto:  
"Locadora Alunos UNOPAR"</font></h3>

<p>
A "Locadora Alunos UNOPAR" é uma empresa privada com fins  
lucrativo, situada na  
região central da cidade de São Paulo, fundada em 10 de maio  
de 1992, com o objetivo  
de <b>locar veículos leves</b>.
</p>

<p>
O <b>diretor</b> Sr. Antônio Carlos, fundador, conta com  
um quadro de <b><font color="red" size=5 face="courier  
new">7 funcionários</font></b> que atuam em diversas áreas,  
sendo: 00:59 07/01/2013 <br>Paulo, Ana Paula e José Roberto  
(<i>departamento de locação</i>), <br>Márcia Regina (<i><font  
color="blue" size=7 face="courier new">departamento de  
compras</font></i>),  
<br>André Augusto (<i>departamento de controle de frota</i>),  
<br>Marcos (<i><font size=1>departamento de manutenção</font>  
</i>) e <br>Maria Helena (<i><font color="green">departamento  
financeiro</font></i>).
</p>

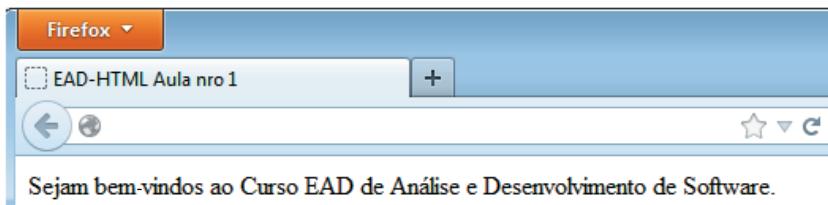
<p>
Seu estoque atual possui 200 veículos, classificados em:  
básico, básico com opcionais, intermediário, intermediário  
com opcionais, premium, premium com opcionais e utilitários.  
Sendo os valores da locação distribuídos, respectivamente:  
R$ 60,00; R$ 80,00; R$ 100,00, R$ 120,00, R$ 150,00, R$ 200,00  
e R$ 130,00. Essas classificações em faixas proporcionais,  
entre o menor e maior preço, segundo os valores de compra.  
Sua estrutura física possui uma área total de 3.500m2,  
contando com a área de garagem, vistoria, pequenos reparos  
e lavagem. A figura 1 apresenta o organograma da "Locadora  
Alunos UNOPAR".
</p>

<p><b><i><blink>Alguns professores do primeiro semestre:</blink></i></b></p>

Merris <u>Mozer</u><br>
Everson <sup>Moraes</sup><br>
Sérgio <sub>Goes</sub><br>
Marco <tt>Hisatomi</tt><br>
<big>Anderson</big> <small>Corrrea</small><br>
</body>
</html>
```

3. Crie um novo documento denominado exercício_01.html que tenha as seguintes especificações:
 - └ Uma palavra formatada em negrito.
 - └ Uma palavra formatada em itálico.
 - └ Uma palavra formatada em negrito e itálico.
 - └ Uma palavra formatada em sobreescrito.
 - └ Uma palavra formatada em subscrito.
 - └ O plano de fundo do documento será da cor roxa.
 - └ O cabeçalho principal deverá ser formatado da cor azul.
4. Agora escreva um pequeno texto que deseja colocar na web com as formatações desejadas.

Exemplo:



Fonte: Do autor (2014).

5. Escreva o código necessário para:
 - └ inserir uma imagem;
 - └ inserir um vídeo que se encontra salvo no seu computador;
 - └ inserir um vídeo do YouTube em um documento .html.

Referências

- CRIARWEB. Tutoriais de webdesign. Disponível em: <http://www.criarweb.com/desenho_web/>. Acesso em: 6 jun. 2014.
- Dib, C. Z. **Tecnologia da educação e sua aplicação à aprendizagem de física**. São Paulo: Pioneira, 1974.
- Dills, C. R.; Romiszowski, A. J. **Instructional development paradigms**. Englewood Cliffs: Educational Technology Publications, 1997.
- FRAZZ. A felicidade... Disponível em: <http://www.frazz.com.br/citacao/www_praiandguia_com_br/a_felicidade_nao_esta_no_fim_da_jornada/67974>. Acesso em: 6 jun. 2014.
- PAULINO, Daniel. Curso básico de Webdesign: introdução. **Oficina da Net**. Disponível em: <http://www.oficinadanet.com.br/artigo/2234/curso_basico_de_webdesign_-_introducao>. Acesso em: 6 jun. 2014.
- Romiszowski, A. J. **Developing autoinstructional materials**. London: Kogan Page, 1983.
- SOUZA, Carlos H. P. Webdesign. Sobre Sites. Disponível em: <<http://www.sobresites.com/webdesign/tutoriais.htm>>. Acesso em: 6 jun. 2014.
- WEBTUTORIAIS. Disponível em: <<http://webtutoriais.com/>>. Acesso em: 6 jun. 2014.
- WIKIPEDIA. Navegador. Disponível em: <<Http://Pt.Wikipedia.Org/Wiki/Browser>>. Acesso em: 6 jun. 2014.
- Will, B. **Distance education: strategies and tools**. Englewood Cliffs: Educational Technology Publications, 1994.

Comandos em html

Adriane Aparecida Loper

Objetivos de aprendizagem: Nesta unidade aprenderemos a trabalhar com imagens em páginas de web, utilizando seus recursos. Criar vínculos entre páginas. Organizar as informações através das tabelas, criar interatividade através dos formulários e construiremos quadros para melhor estruturar nossas páginas, com um visual agradável e de acesso rápido para o usuário.

↳ Seção 1: **Imagens em html e links**

As imagens trazem para qualquer site um enriquecimento intenso. Quando navegamos necessitamos de imagens coloridas, belas, para que as páginas não se tornem monótonas e consigam transmitir a mensagem desejada ao usuário. Há uma frase que nos demonstra esse pensamento: "uma imagem nos diz mais que as palavras".

↳ Seção 2: **Listas, tabelas e formulários em html**

Nesta seção utilizaremos listas para trabalhar com itens e subitens com marcadores e numeração que assumem um padrão ou que podemos criar. Tabelas em HTML assumem não apenas a tabulação como também a parte estética de sua página e os formulários permitem a comunicação ou a interatividade entre os usuários e o site.

↳ Seção 3: **Frames, animação simples (marquee) em html**

Nesta seção trataremos dos frames, que são quadros que dividem a janela do browser fazendo com que sejam mostradas mais de uma página ao mesmo tempo. Assim, faremos um planejamento prévio da estrutura para dimensioná-la e deixá-la de fácil leitura para nossos usuários.

Introdução

Nesta unidade trabalharemos com estruturas de páginas web através da linguagem de marcação HTML. Partindo-se do princípio de que na web temos um usuário de comportamento ativo, que quer ser participativo, que publica suas informações pessoais e expressa suas opiniões cada vez mais rápido.

Temos que acompanhar essas evoluções para podermos elaborar páginas criativas e que chamem a atenção de nosso usuário, afinal essa mídia é nova, muito ativa e diferente das opções de diversão que tínhamos até agora, como o rádio e a televisão. Assim sendo, nossa obrigação é criarmos páginas muito bem elaboradas que atinjam uma grande parte dos usuários da web.

Iniciaremos a unidade com a inserção de imagens e links em nossa página. Imagens devem ser representativas e com processamento rápido, afinal os usuários não têm paciência para ficar esperando uma imagem ser formada. Depois vamos trabalhar com listas, tabelas e formulários, frames e marquees que permitem que a parte estética seja contemplada.

Fazendo um bom estudo e planejamento de nosso público-alvo, conseguiremos desenvolver páginas adequadas. Vamos utilizar essas ferramentas?

"A Web tinha menos de 10 milhões de sites em 2000. E, em 2007, a Web tinha 80 milhões de sites. Hoje a situação é esta: as pessoas simplesmente pressupõem que a Web tem o que elas querem" (NIELSEN; LORANGER, 2007, p. XX). Para conseguirmos colocar todas as informações pertinentes ao nosso negócio na web, precisamos construir páginas claras, leves e rápidas para sermos competitivos. A linguagem HTML é uma ferramenta que nos auxilia muito nessa construção. Vamos descobrir como?

Seção 1 **Imagens em html e links**

Nesta seção iniciaremos nossa aprendizagem ao inserir imagens e utilizar seus recursos, como modificar sua altura e largura, inserir bordas e dimensionar as imagens para o centro, direita ou esquerda, ou figuras como fundo para melhor trabalharmos nossas páginas web.

1.1 **Imagens para web**

1.1.1 **Sintaxe do tag imagem**

Certamente, os sites na Internet seriam muito difíceis de trabalhar se não fosse possível adicionar imagens ao conteúdo das páginas. Inserir imagens em páginas Web é simples; necessita-se ter a imagem desejada e fazer uma referência a ela com o tag ``, onde o `src` é o seu atributo. O valor do atributo `src` (`source` = origem) aponta para o endereço da imagem. O valor do atributo pode ser o caminho abso-

luto ou relativo de uma imagem. Temos que estar atentos também ao tamanho das imagens; se ocupam muito espaço, isso implicará tempo para aparecer as imagens na tela, o que faz o usuário perder a paciência, o que jamais seria nosso objetivo.

A sintaxe do tag Imagem é:

```
<img src = " nome da figura. extensão. >
```

1.1.2 Formato dos arquivos de imagem

Em geral, a página de HTML aceita arquivos de imagem em formatos JPEG e GIF, pois esses formatos comprimem as imagens, deixando o arquivo pequeno. Cada formato tem seus objetivos, então deve-se escolher a forma mais adequada.

1.1.2.1 **GIF**

A extensão GIF, do inglês "Graphics Interchange Format", é o formato usado para todos os tipos de grafismos e logotipos, animados ou não. O processamento de leitura é rápido, pode ser gravado em disco e recarregado diversas vezes sem perder a qualidade, e utiliza até 256 cores.

1.1.2.2 **JPEG**

A extensão JPEG, do inglês "Joint Photographic Experts Group", é o formato usado para fotos ou desenhos detalhados. O processamento de leitura é rápido; cada vez que esse arquivo for gravado e regravado ele perde aos poucos a qualidade se tornando, após repetidas utilizações, inapropriado para uso. Utiliza os tons de cinza ou ainda sistema de cores RGB, 200 dpi.

1.1.2.3 **Comparando GIF e JPEG**

A taxa de compressão da extensão JPEG é melhor que a da extensão GIF.

A taxa de descompressão da extensão JPEG é mais demorada que a da extensão GIF, fazendo com que os GIF apareçam mais rápido na tela.

A extensão GIF é mais pobre em cores, porém podem-se fazer imagens transparentes.

Devemos transformar a extensão BMP em GIF ou JPEG para melhor trabalharmos em nossa página HTML.



Questões para reflexão

Você sabe como escolher uma imagem que represente a mensagem exata que desejamos transmitir a nossos usuários?

1.1.3 Atributos do tag imagem

- └ **Src:** este atributo nos indica onde está a imagem;
- └ **Alt:** este atributo nos indica um texto alternativo ou legenda — texto que normalmente os navegadores mostram quando paramos o mouse sobre a imagem;
- └ **Border:** este atributo nos configura uma borda ou moldura na imagem. Seus valores indicam 0 se não houver borda e infinito para borda;
- └ **Width:** este atributo nos configura a largura da imagem. Seu valor é medido em pixels;
- └ **Height:** este atributo nos configura a altura da imagem. Seu valor é medido em pixels;
- └ **Align:** atribui valores para o alinhamento horizontal, que podem ser à esquerda, ao centro, à direita (left, center, right) ou alinhamento vertical (top middle, baseline e bottom);
- └ **Align = Left:** a imagem fica à esquerda;
- └ **Align = Right:** a imagem fica à direita;
- └ **Align = Center:** a imagem fica ao centro;
- └ **Align = Top:** alinha o topo da imagem com o topo do elemento mais alto da linha;
- └ **Align = Middle:** alinha a base do texto com o centro da imagem;
- └ **Align = Bottom:** alinha a imagem com a base dos outros elementos da lista;
- └ **Align = Texttop:** alinha o topo da imagem com o topo do texto;
- └ **Align = Absmiddle:** alinha o centro do texto com o centro da imagem;
- └ **Align = Baseline:** alinha a imagem com a base dos outros elementos da linha;
- └ **Align = Absbottom:** alinha a imagem com a base dos outros elementos da linha;
- └ **Vspace:** especifica o espaço acima e abaixo da imagem. Seu valor é dado em pixels;
- └ **Hspace:** especifica o espaço à direita e à esquerda da imagem. Seu valor é dado em pixels;

Veja um exemplo de formatação: Inserindo imagem

```
<html>
<head>
<title> Inserir Imagens na WEB </title>
</head>
<body>
<h3><center>Trabalhando imagens</center></h3>
<p>
<center>Colocaremos "banana.jpg".</center>
<p>
<img src ="banana.jpg">
</body>
</html>
```

Figura 2.1 Exemplo 1: Inserindo uma imagem jpg

Fonte: Do autor (2014).

1.1.4 Alinhamento de imagens com textos

Quando uma imagem é colocada perto de um texto, ela, por definição padrão, alinha a parte inferior da figura com a linha do texto.

Exemplo 2: Imagem alinhada padrão

```
<html>
<head>
<title> Inserir Imagens na WEB </title>
</head>
<body>
<h3><center>Trabalhando imagens</center></h3>
<p>
```

O desenho de bananas que você está vendo `` será alinhado à parte inferior da figura, com a linha de texto.

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

Figura 2.2 Exemplo 2: Inserindo um texto e uma imagem jpg

Fonte: Do autor (2014).

1.1.4.1 Alinhamento do centro das imagens com textos

```
<html>
<head>
<title> Inserir Imagens na WEB </title>
</head>
<body>
<h3><center>Trabalhando imagens</center></h3>
<p>
```

O desenho de bananas que você está vendo será alinhado à parte inferior da figura, com a linha de texto.

Exemplo 3: Imagem centralizada

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

Figura 2.3 Exemplo 3: Inserindo um texto e uma imagem centralizada



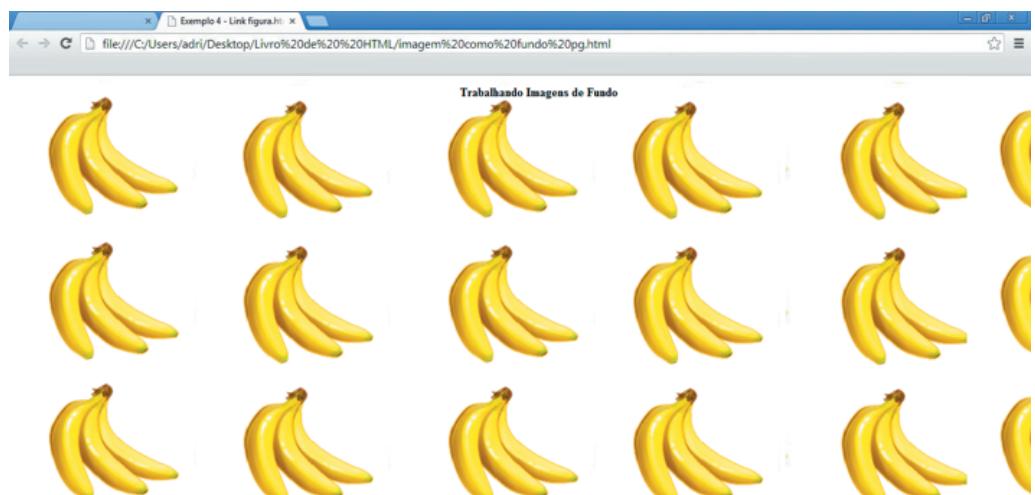
Fonte: Do autor (2014).

1.1.5 Usando imagens como fundo de página

Exemplo 4: Imagem como fundo

```
<html>
<head>
<title> Inserir Imagens na WEB </title>
</head>
<body background = " banan.jpg">
<h3><center>Trabalhando imagens de fundo</center></h3>
<p>
</body>
</html>
```

Figura 2.4 Exemplo 4: Inserindo uma imagem como fundo



Fonte: Do autor (2014).

1.1.6 Transparência

Alguns programas permitem criar imagens com o fundo *transparente*, ou seja, permitem definir que determinada área da imagem não tenha cor e dê a impressão de “vazada” (como a figura da direita).

Figura 2.5 Figura com transparência



Fonte: Do autor (2014).

Nesse caso, certa cor é selecionada para aparecer como transparente. Ao carregar a imagem no navegador, essa cor não será mostrada, gerando o efeito de transparência. Então, é necessário que a área que se pretende deixar transparente tenha somente uma cor, que não pode se repetir em qualquer outra parte da imagem. Se essa cor existir em outro local, tal pedaço também ficará transparente. Veja a imagem acima à direita. É a mesma imagem mostrada no lado esquerdo, mas em que a cor branca do fundo foi selecionada para aparecer como transparente.

Além do tamanho do arquivo, outros pontos devem ser considerados. Assim, quando se usa uma imagem como fundo `<body background>` de uma página, deve-se cuidar para não carregar demais a página.

Pode-se usar a mesma imagem em várias hp's do site, para identificá-lo. Portanto, essa imagem deve ser muito bem escolhida, pois será a mais vista do site. Se for possível, optar pelos fundos muito claros ou muito escuros, pois não se consegue uma boa relação de contraste de cores com o texto quando se usa imagens com cores intermediárias.

É importante notar que *nada é pior* que uma página ilegível. Na Internet, em que há tantas páginas lindas, úteis e fáceis de ler, é querer demais que o usuário se esforce para, simplesmente, ler um texto.

1.2 Links

Um site é composto de várias páginas web e para vincular uma página a outra é preciso criar links. A palavra link vem do inglês e significa ligação ou caminho. Através dos links é possível navegar entre páginas, a partir de palavras ou imagens.

Links são ligações dinâmicas entre os documentos. Quando o usuário clica sobre um link, ele é levado para uma outra parte ou para um outro documento dentro da rede.

“Âncora” é uma referência dentro da página. Quando criamos um link para uma âncora, é como se criássemos um link, interno ao próprio documento. Assim, na página HTML, quando clicamos no link, o navegador nos leva até a “âncora”.

Para se definir a âncora, vamos até o local onde queremos colocá-la e indicamos. Com HTML é possível fazer ligações de alguma parte do texto ou imagem para outro documento. Os links podem ser agrupados da seguinte forma:

- └ Links internos: aqueles que indicam outras partes da mesma página;
- └ Links locais : aqueles que indicam outras páginas no mesmo site;
- └ Links remotos: aqueles que indicam outras páginas de outros sites;
- └ Links com endereços de correio: para criar uma mensagem de correio.

A Internet é totalmente linkada, ou seja, quando o usuário clicar em determinado texto, deverá redirecionar para uma página.

Opções principais disponíveis:

- └ Target
- └ “_blank” abre nova janela/browser;
- └ “_top” na mesma janela, sobrepondo todas as definições do frame;
- └ “nome” indica destino na estrutura de frame;
- └ “#âncora” indica âncora dentro da página.

“Âncora” é uma referência dentro da página. Quando criamos um link para uma âncora, é como se criássemos um link interno ao próprio documento. Assim, na página HTML, quando clicamos no link o navegador nos leva até a “âncora”.

Para se definir a âncora, vamos até o local onde queremos colocá-la e indicamos
``

O comando é semelhante à criação de um hiperlink. Pode-se colocar uma imagem ou mesmo um texto entre os comandos `<a>`, mas não é imprescindível.

Para isto é necessário criar uma âncora utilizando o TAG `<a>` , assim:

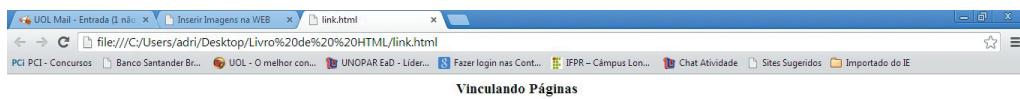
A sintaxe do tag link é :[JR1]

` Descrição do link`

Ex. Utilização: ``

1.Exemplo de âncora

```
<body>
<h3><center> Vinculando Páginas</center></h3>
<p>
Um link em geral se transforma em “dedo”
<p>
<a href="cadastro.htm">Clique para cadastro</a>
</body>
```

Figura 2.6 Exemplo 5: Inserindo um link

Fonte: Do autor (2014).

1.2.1 Usando uma imagem como link

Exemplo 6: Insere uma figura como link

```
<body>
<h3><center> Imagens como link</center></h3>
<p>
<center> Clique nas bananas para informações </center>
<p>
<a href="banana.htm"></a><p>
</body>
```

Figura 2.7 Exemplo 6: O link posicionado na imagem

Fonte: Do autor (2014).

1.2.2 Abrindo o link em uma nova aba

Para abrir um link em uma nova aba, vamos utilizar o comando target com o valor igual a _blank (abre o link em uma nova aba do navegador).

Exemplo:

```
<a href = "http://www.unopar.com.br" target ="_blank"> Abre o site em uma nova aba.</a>
```

1.2.3 Criando um link para e-mail

Para se criar um link que acesse um endereço de e-mail e que quando acionado abra o seu serviço de gerenciamento de e-mails padrão, utilizamos:

```
<a href="mailto:usuario@dominio.com.br">
```

Observe a necessidade de indicar "mailto:" antes do endereço de e-mail destino da mensagem.

Exemplo 7: Criar um link para e-mail

```
<html>
<head>
<title> Link de e-mail</title>
</head>
<body>
<h3><center> Utilizando um endereço de e-mail como link </center></h3>
<p>
<center> Clique no link abaixo e envie e-mail </center>
<p>
<a href= "mailto:nome@provedor.com.br"> seunome@provedor.com.br</a><p>
</body>
</html>
```

Figura 2.8 Exemplo 7: Link de e-mail



Fonte: Do autor (2014).



Atividades de aprendizagem

1. Faça uma página que deverá ter uma imagem que abra a página da Unopar em uma aba nova e inclua uma âncora para navegação interna do documento.
2. Monte uma página com 3 imagens de frutas (1 banana, 1 maçã e 1 abacaxi) e faça uma breve descrição delas com seus links.



Para saber mais

RODRIGUES, Andrea. **Desenvolvimento para Internet**. Curitiba: LT, 2010. Nos capítulos 2 e 4 a autora trata de links e imagens.

Seção 2 Listas, tabelas e formulários em html

Quando navegamos pela Internet, por diversas vezes procuramos informações, preços, orçamentos e para isto o modo de comunicação entre o site e o usuário são informações colhidas através de formulários. É através dos formulários que os visitantes inserem seus dados para uma futura consulta ou resposta. Listas e tabelas nos ajudam na organização de nossas páginas.

2.1 Listas

Listas são formas de ordenação e estruturação do conteúdo de texto das páginas web. As listas são criadas para dar aparência de índices ou resumos do conteúdo dessas páginas. As listas em HTML podem ser:

- Não ordenadas;
- Ordenadas;
- Encadeadas.

Figura 2.9 Índice de um livro: tópicos

└ Unidade 1 – Introdução	1
O conceito de lógica	1
Lógica no dia-a-dia.....	2
Aplicabilidade da lógica no auxílio do desenvolvimento de programas.....	2
O que é algoritmo?.....	2
Sobre linguagens de programação	4
└ Unidade 2 – Tipos de representação de algoritmos	7
Descrição narrativa.....	7
Pseudolinguagem ou pseudocódigo (Portugol).....	8
Fluxogramas	8
Diagrama de Nassi-Shneiderman ou diagrama de Chapin	10
Que representação iremos utilizar?.....	10
Aspecto estático e aspecto dinâmico dos algoritmos.....	10
Etapas a serem seguidas para criação de algoritmos	11
Um ambiente para escrever e executar algoritmos	11
Funcionamento do nosso computador	11
Resolvendo um problema	12

Fonte: Do autor (2014).

2.1.1 Listas não ordenadas

As listas não ordenadas são geradas pela estrutura dos tags ``.

Estrutura básica:

```
<ul>
<li>
</li>
</ul>
```

Exemplo 1: Lista simples não ordenada padrão

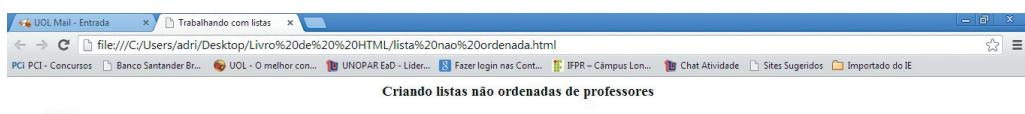
```
<html>
<head>
```

```

<title> Trabalhando com listas</title>
</head>
<body>
    <h3><center> Criando listas não ordenadas de professores </center></h3>
    <p>
        <ul>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ul>
    </body>
</html>

```

Figura 2.10 Exemplo 1: Lista simples padrão



Fonte: Do autor (2014).

As listas não ordenadas assumem os marcadores em forma de círculo como padrão de marcadores da lista. Podemos modificá-los com o parâmetro type do tag — temos as opções square, circle e disk.

Vamos fazer o Exemplo 1 com essas três opções:

Exemplo 1.1: Opções de marcadores nas listas não ordenadas

```

<html>
<head>
<title> Trabalhando com listas</title>
</head>
<body>
    <h3><center> Criando listas não ordenadas de professores </center></h3>
    <p>
        <ul type = square>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ul>
        <ul type = circle>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ul>
        <ul type = disc>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ul>
    </body>
</html>

```

Figura 2.11 Exemplo 1.1: Variações das listas não ordenadas

Fonte: Do autor (2014).

2.1.2 Listas ordenadas

As listas ordenadas são geradas pela estrutura dos tags

Estrutura básica:

```
<ol>
<li>
</li>
</ol>
```

Exemplo 2: Lista ordenada

```
<html>
<head>
<title> Trabalhando com listas</title>
</head>
<body>
    <h3><center> Criando lista ordenada de professores </center></h3>
    <p>
        <ol>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ol>
    </body>
</html>
```

Figura 2.12 Exemplo 2 — Lista ordenada

Fonte: Do autor (2014).

2.1.2.1 Listas ordenadas alterando os seus numeradores

As listas ordenadas podem ser modificadas através do parâmetro type. As opções desse parâmetro são (Quadro 2.1):

Quadro 2.1 Parâmetro para listas ordenadas

Parâmetro	Resolução
Type =1	Cria uma lista numérica padrão
Type =A	Cria uma lista alfabetica com letras maiúsculas iniciando com A
Type =a	Cria uma lista alfabetica com letras minúsculas iniciando com a
Type =I	Cria uma lista numérica com algarismos romanos maiúsculos
Type = i	Cria uma lista numérica com algarismos romanos minúsculas

Exemplo 2.1: Lista ordenada com numeradores alterados

```
<html>
<head>
<title> Trabalhando com listas</title>
</head>
<body>
    <h3><center> Criando lista ordenada de professores </center></h3>
    <p>
        <ol type = A>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ol>
        <ol type = I >
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ol>
        <ol type = 1>
            <li> Adriane<br>
            <li> Everson<br>
            <li> Merris<br>
            <li> Veronice <br>
        </ol>
    </body>
</html>
```

Figura 2.13 Exemplo 2.1: Listas ordenadas com diferentes marcadores

Fonte: Do autor (2014).

2.1.3 Listas encadeadas

É importante mencionar no tópico sobre listas que se pode criar encadeamento de listas ordenadas x listas não ordenadas misturando as duas categorias, ou simplesmente criando encadeamentos.

Sintaxe de lista encadeada:

```
<ol>
<li></li>
<ul>
<li></li>
</ul>
</ol>
```

2.2 Tabelas

Tabelas são muito usadas na organização de dados e informações, que são compostas por linhas e colunas onde a intersecção desses elementos formam as células. As tabelas são muito úteis na parte estética da pagina web.

2.2.1 Estrutura de uma tabela

Para criar tabelas, vamos utilizar o tag `<table> </table>`, que define uma tabela em HTML. Dentro do tag `<table>`, precisamos definir a estrutura de linhas e colunas que formarão cada célula de nossa tabela. Para isso, vamos usar outros dois tags, um que cria linhas e outro que cria as colunas de uma tabela. Esses tags são:

- └ `<tr></tr>` → tag que define uma linha, abreviação do inglês "Table Row";
- └ `<td></td>` → tag que define uma coluna, abreviação do inglês "Table Data".
Esses tags devem ser informados em tags para cada célula da linha e devem estar sempre entre os tags `<tr></tr>`;
- └ `<th></th>` → define a tabela cabeçalho, abreviação do inglês "Table Header";
são utilizados com o mesmo propósito dos tags `<td></td>`, porém o conteúdo é exibido em negrito e centralizado;

- └ **<Caption></Caption>** → Caption significa título; essa estrutura de tags é utilizada para criar legenda para a tabela e deve ser indicada após o tag `<table>` e antes do tag `<tr>`. Essa legenda é um texto que pode ser exibido no cabeçalho (topo) ou no rodapé da tabela, depende apenas de um valor atribuído ao parâmetro Align que pode ser top (topo) ou bottom (rodapé);
- └ **Border = valor** → É a borda da tabela, já que as tabelas não apresentam bordas quando são criadas. A indicação desse parâmetro sem nenhum valor, faz aparecer uma linha fina como borda; a espessura da borda dependerá do valor informado, quanto maior o valor numérico, mais espessa será a borda;
- └ **Align** → Align significa alinhamento; esse parâmetro acompanha vários tags da linguagem HTML. Dentro do tag `<Caption>` Align especifica os valores que podem ser top (topo) ou bottom (rodapé). Dentro dos tags `<tr>`, `<td>` e `<th>`, Align especifica os valores left (esquerda), center (centro) ou right (direita).
- └ **Valign** → Valign significa alinhamento vertical, este parâmetro tem a mesma finalidade do Align, porém é utilizado para alinhar o texto verticalmente na célula. Suas opções são top, middle, bottom e baseline.

Exemplo 3: Uma tabela de três linhas e três colunas:

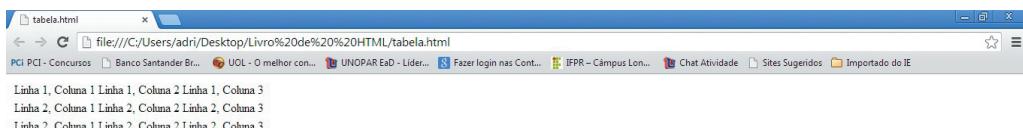
```
<table>
  <tr>
    <td> Linha 1, Coluna 1 </td>
    <td> Linha 1, Coluna 2 </td>
    <td> Linha 1, Coluna 3 </td>

  </tr>
  <tr>
    <td> Linha 2, Coluna 1 </td>
    <td> Linha 2, Coluna 2 </td>
    <td> Linha 2, Coluna 3 </td>

  </tr>
  <tr>
    <td> Linha 3, Coluna 1 </td>
    <td> Linha 3, Coluna 2 </td>
    <td> Linha 3, Coluna 3 </td>

  </tr>
</table>
```

Figura 2.14 Exemplo 3: Uma tabela de três linhas e três colunas



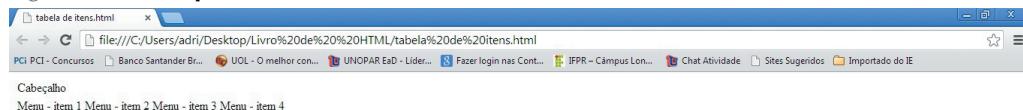
Fonte: Do autor (2014).

Além de criar tabelas em formato padrão, você pode intercalar novas linhas e/ou colunas dentro de qualquer parte de uma tabela, possibilitando, assim, a construção de uma tabela que irá atender a suas necessidades.

Exemplo 4: Tabela de itens

```
<table>
  <tr>
    <td> Cabeçalho </td>
  </tr>
  <tr>
    <td> Menu - item 1 </td>
    <td> Menu - item 2 </td>
    <td> Menu - item 3 </td>
    <td> Menu - item 4 </td>
  </tr>
</table>
```

Figura 2.15 Exemplo 4: Tabela de itens



Fonte: Do autor (2014).

Observe o exemplo acima sem qualquer recurso, no topo de um site. Essa tabela tem duas linhas; a primeira possui uma coluna, onde será inserido o cabeçalho do site, e a segunda que possui quatro colunas, sendo cada coluna um item do menu de navegação de um site.

Podemos melhorar muito nossas tabelas.

2.2.2 Comandos

Para personalizar a nossa tabela, temos à disposição diversos comandos HTML. Observe abaixo uma pequena lista de comandos, suas funções e seus possíveis valores:

Quadro 2.2 Comandos de tabela

Comando	Função	Valor
width	Controla a largura da tabela.	Valores em píxel, porcentagem e outras unidades numéricas
height	Controla a altura da tabela.	Valores em píxel, porcentagem e outras unidades numéricas
align	Alinha a tabela.	center (centro), left (esquerda) right (direita)
border	Controla a largura da borda da tabela.	Valores em píxel, porcentagem e outras unidades numéricas
cellpadding	Define uma margem interna para as células de uma tabela.	Valores em píxel, porcentagem e outras unidades numéricas
cellspacing	Define um espaço entre as células de uma tabela.	Valores em píxel, porcentagem e outras unidades numéricas
background	Define uma imagem para o fundo da tabela.	URL de uma imagem
bgcolor	Define uma cor para o fundo da tabela.	Números de cores em hexadecimal rgb etc.

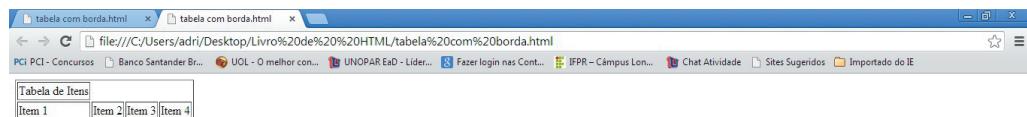
2.2.3 Criando tabelas com bordas

Analisamos nos exemplos anteriores que quando construímos a tabela, ela vem sem bordas criadas. Vamos agora acrescentar o tag Border se não acrescentarmos nenhum valor, aparecerá uma linha fina como borda.

Exemplo 5: Tabela de itens com bordas

```
<table border>
  <tr>
    <td> Tabela de Itens </td>
  </tr>
  <tr>
    <td> Item 1 </td>
    <td> Item 2 </td>
    <td> Item 3 </td>
    <td> Item 4 </td>
  </tr>
</table>
```

Figura 2.16 Exemplo 5 — Tabela de itens com bordas



Fonte: Do autor (2014).

Colocamos bordas, porém ainda podemos melhorar muito.

2.2.4 Criando tabelas com títulos para suas colunas

Exemplo 6: Tabela com bordas e título para as colunas

```
<table border = 2>
  <tr>
    <th> Título A </th><th> Título B </th><th> Título C </th>
  </tr>
  <tr>
    <td> Linha 1, Coluna 1 </td>
    <td> Linha 1, Coluna 2 </td>
    <td> Linha 1, Coluna 3 </td>
  </tr>
  <tr>
    <td> Linha 2, Coluna 1 </td>
    <td> Linha 2, Coluna 2 </td>
    <td> Linha 2, Coluna 3 </td>
  </tr>
  <tr>
    <td> Linha 3, Coluna 1 </td>
    <td> Linha 3, Coluna 2 </td>
    <td> Linha 3, Coluna 3 </td>
  </tr>
</table>
```

Figura 2.17 Exemplo 6 — Tabela com bordas e título para as colunas

The screenshot shows a Microsoft Internet Explorer window with the URL <file:///C:/Users/adri/Desktop/Livro%20de%20HTML/tabelas%20com%20titulos%20para%20colunas.html>. The page displays a table with three columns titled 'Título A', 'Título B', and 'Título C'. The table has 3 rows, each containing three cells. The data is as follows:

Título A	Título B	Título C
Linha 1, Coluna 1	Linha 1, Coluna 2	Linha 1, Coluna 3
Linha 2, Coluna 1	Linha 2, Coluna 2	Linha 2, Coluna 3
Linha 3, Coluna 1	Linha 3, Coluna 2	Linha 3, Coluna 3

Fonte: Do autor (2014).

2.2.5 Criando tabelas com legendas

Exemplo 7: Tabela com bordas, títulos das coluna e legendas

```
<table border = 10 >
<caption align=top> Legenda no topo da tabela de três linha e três colunas</caption>
<tr>
    <th> Título A </th><th> Título B </th><th> Título C </th>
</tr>
<tr>
    <td> Linha 1, Coluna 1 </td>
    <td> Linha 1, Coluna 2 </td>
    <td> Linha 1, Coluna 3 </td>
</tr>
<tr>
    <td> Linha 2, Coluna 1 </td>
    <td> Linha 2, Coluna 2 </td>
    <td> Linha 2, Coluna 3 </td>
</tr>
<tr>
    <td> Linha 3, Coluna 1 </td>
    <td> Linha 3, Coluna 2 </td>
    <td> Linha 3, Coluna 3 </td>
</tr>
</table>
```

Figura 2.18 Exemplo 7 — Tabela com legendas

The screenshot shows a Microsoft Internet Explorer window with the URL <file:///C:/Users/adri/Desktop/Livro%20de%20HTML/legenda.html>. The page displays a table with three columns titled 'Título A', 'Título B', and 'Título C'. The table has 3 rows, each containing three cells. The data is as follows:

Título A	Título B	Título C
Linha 1, Coluna 1	Linha 1, Coluna 2	Linha 1, Coluna 3
Linha 2, Coluna 1	Linha 2, Coluna 2	Linha 2, Coluna 3
Linha 3, Coluna 1	Linha 3, Coluna 2	Linha 3, Coluna 3

Fonte: Do autor (2014).

2.2.6 Criando cores na tabela <bcolor>

Exemplo 8: Tabela com cores

```
<table border = 10 bcolor = "yellow">
<caption align=top> Cor (amarela) na tabela de três linha e três colunas</caption>
<tr>
    <th> Título A </th><th> Título B </th><th> Título C </th>
```

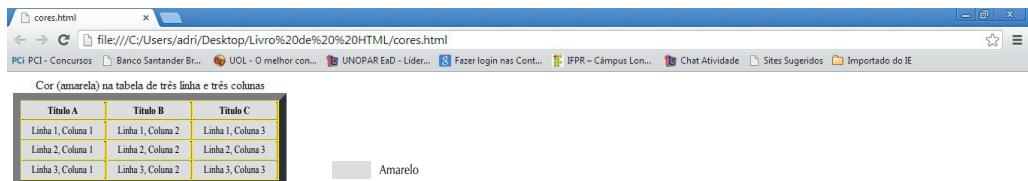
```

</tr>
<tr>
    <td> Linha 1, Coluna 1 </td>
    <td> Linha 1, Coluna 2 </td>
    <td> Linha 1, Coluna 3 </td>

</tr>
<tr>
    <td> Linha 2, Coluna 1 </td>
    <td> Linha 2, Coluna 2 </td>
    <td> Linha 2, Coluna 3 </td>
</tr>
<tr>
    <td> Linha 3, Coluna 1 </td>
    <td> Linha 3, Coluna 2 </td>
    <td> Linha 3, Coluna 3 </td>
</tr>
</table>

```

Figura 2.19 Exemplo 8 — Tabela com cores



Fonte: Do autor (2014).

2.2.7 Criando cores nas células

Exemplo 9 — Tabela com cores nas células

```

<table border = 10 bgcolor = "yellow">
<caption align=top> Legenda no topo da tabela de três linha e três colunas</caption>
<tr>
    <th> Título A </th><th> Título B </th><th> Título C </th>
</tr>
<tr>
    <td bgcolor "blue"> Linha 1, Coluna 1 </td>
    <td bgcolor "red"> Linha 1, Coluna 2 </td>
    <td> Linha 1, Coluna 3 </td>

</tr>
<tr>
    <td> Linha 2, Coluna 1 </td>
    <td> Linha 2, Coluna 2 </td>
    <td bgcolor "green"> Linha 2, Coluna 3 </td>
</tr>
<tr>
    <td> Linha 3, Coluna 1 </td>
    <td> Linha 3, Coluna 2 </td>
    <td> Linha 3, Coluna 3 </td>
</tr>
</table>

```

Figura 2.20 Exemplo 9 — Tabela com cores nas células

Título A	Título B	Título C
Linha 1, Coluna 1	Linha 1, Coluna 2	Linha 1, Coluna 3
Linha 2, Coluna 1	Linha 2, Coluna 2	Linha 2, Coluna 3
Linha 3, Coluna 1	Linha 3, Coluna 2	Linha 3, Coluna 3

Legenda no topo da tabela de três linhas e três colunas

Amarelo	Verde
Vermelho	Azul

Fonte: Do autor (2014).

2.3 Formulários

Os formulários em HTML são utilizados para a coleta de dados do usuário, permitindo a interatividade, e após isso podem ser manipulados com a utilização de outras linguagens como PHP. Um exemplo muito comum de um formulário em HTML é o formulário de cadastro de usuários de e-mail, no qual estão inseridos campos como nome, sobrenome, data de nascimento, entre outros.

Os formulários são indispensáveis em nosso dia a dia, além de um dos recursos mais importantes na linguagem HTML. São usados em inúmeras situações, por exemplo, para enviar um simples e-mail. O elemento `<form>` sozinho não permite que o navegador desenhe nada na página, nem permite inserir informações. Ele delimita os elementos que coletam os dados e possui informações que orientam o navegador sobre como e para onde devem ser enviadas essas informações.

Uma das aplicações comuns de formulários é o comércio eletrônico.



Questões para reflexão

Muitas vezes, temos receio de preencher formulários com dados pessoais, pois há o receio desses dados serem utilizados de forma inadequada. O que poderemos construir em nossas páginas para maior segurança dos usuários?

2.3.1 Sintaxe dos tags de um formulário.

- └ `<form> </form>` → É o tag responsável pela criação de formulários. É nesse tag que precisamos decidir como os dados serão trabalhados em nosso formulário; esse tag indica o início e o fim de um formulário;
- └ `<input type>` → Define os campos de informação de dados do formulário;
- └ `<select>...</select>` → Define uma lista de itens onde o usuário pode optar por uma delas;
- └ `<textarea>...</textarea>` → Define uma caixa onde se pode digitar textos.

2.3.2 Criando os campos do formulário

Existem vários tipos de campos e elementos que podem ser utilizados na construção de um formulário; vamos ver alguns comandos muito utilizados.

Observe a tabela de funções de um formulário que serão aplicadas aos seus devidos campos e parâmetros (Quadro 2.3):

Quadro 2.3 Comandos para formulários

Comandos	Funções	Valores
name	Atribui um nome ao campo. É muito importante a utilização deste comando, pois é através desse nome atribuído a cada campo que poderemos capturar os dados e manipulá-los.	Qualquer nome que não inicie com símbolos ou números, e que não contenha caracteres especiais (ç), nem acentos ou espaços.
value	Atribui um valor predefinido ao campo.	Qualquer tipo de valor. Por exemplo, frases, números entre outros.
maxlength	Determina um valor máximo de caracteres que um campo irá suportar.	Números inteiros maiores que zero.
size	Determina o tamanho do campo em caracteres.	Números inteiros maiores que zero.
rows	Determina a quantidade de linhas em uma área de texto.	Números inteiros maiores que zero.
cols	Determina a quantidade de colunas em uma área de texto.	Números inteiros maiores que zero.
disabled	Desabilita um campo ou área de texto, impossibilitando ao usuário inserir dados.	Valor disabled, em português desativado.
checked	Este comando é aplicável a caixas de seleção e a botões do tipo rádio, e quando for aplicado, a opção já aparece selecionada.	Valor checked, em português verificado/marcado/selecionado.
type	Define o tipo do campo ou elemento do formulário.	button (botão), checkbox (caixa de seleção), file (campo para a seleção de arquivos no PC), password (campo de senha), radio (botão rádio), reset (botão que limpa o formulário), submit (botão que envia o formulário), text (campo de texto).

Fonte: Do autor (2014).

2.3.3 O tag <Input type>

Este tag define os tipos de dados com os quais o usuário irá trabalhar. Há vários campos editáveis dentro de um formulário. Cada campo de um formulário deve atribuir seu conteúdo a uma variável. Uma variável pode ser definida como um elemento lógico que armazena informações para serem tratadas. Os tipos mais comuns de campos são: text, checkbox, radio, submit, reset e password.

2.3.4 Campo de texto

Um campo de texto é criado com o tag <input> e definido seu tipo para texto com o comando type; também não podemos esquecer de nomeá-lo com o comando name. Ele aceita dados do tipo character, que podem ser letras, números ou símbolos. Em geral, muito usados para cadastro. O padrão é uma caixa de texto com 20 caracteres.

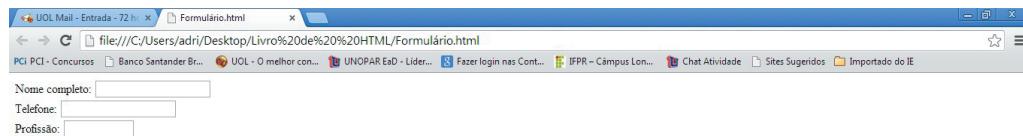
Os campos text possuem alguns padrões de configuração que são:

- ❑ **Size** → Limita o tamanho do campo para digitação, mas não limita o texto da digitação;
- ❑ **maxlength** → Limita a quantidade de caracteres que podem ser digitados no campo;
- ❑ **Value** → Especifica um valor padrão para o campo, porém esse valor pode ser alterado;
- ❑ **Submit** → Envia os dados do formulário para um servidor de dados.

Exemplo 10: Formulário simples

```
<form>
Nome completo: <input type=text name = nome completo> <br>
Telefone: <input type=text maxlength=9 name = telefone> <br>
Profissão: <input type=text size=10 name = profissão> <br>
</form>
```

Figura 2.21 Exemplo 10 — Formulário simples



Fonte: Do autor (2014).

2.3.4.1 Senha

A única diferença entre um campo de senha e um de texto é que no campo de senha os caracteres são protegidos, ou seja, o navegador exibe asteriscos ou bolinhas em vez dos caracteres. Para criar um campo de senha, defina o valor do comando type para password.

Exemplo 11: Formulário com senha

```
<html>
<head>
<title> Formulários</title>
</head>
<body>
<h3><center>CRIANDO FORMULÁRIOS COM SENHA</centre></h3><p>
<form>
Nome completo: <input type=text name = nome completo> <br>
Senha:<input type=password name=senha><br>
</form>
</body>
</html>
```

Figura 2.22 Exemplo 11 — Formulário com senha

A screenshot of a web browser window titled "UOL Mail - Entrada (1 não)" with the URL "file:///C/Users/adri/Desktop/Livro%20de%20HTML/Formulário%20com%20senha.html". The page title is "CRIANDO FORMULÁRIOS COM SENHA". It contains two input fields: "Nome completo:" with the value "aaaaaaaaaaaaaaaaaaaaaa" and "Senha:" with the value "*****".

Fonte: Do autor (2014).

2.3.5 Botões de seleção radio

Chama-se botão de rádio para nos indicar que poderemos escolher apenas uma estação de rádio por vez. Um conjunto desses botões permite ao usuário selecionar apenas uma opção. Em geral, utilizamos para resposta única de uma pergunta.

Observe que nesse tipo de botão de seleção o comando name de todas as opções deve ter o mesmo valor. No caso da caixa de seleção e dos botões radio, o valor a ser enviado é o valor do comando value.

Figura 2.23 Exemplo 12 — Formulário com botões de rádio

A screenshot of a web browser window titled "UOL Mail - Entrada (1 não)" with the URL "file:///C/Users/adri/Desktop/Livro%20de%20HTML/Formularios%20com%20radio.html". The page title is "CRIANDO FORMULÁRIOS". It contains an input field "Nome completo:" with the value "Adriane Aparecida Loper" and a group of radio buttons labeled "Estado Civil". The "Solteiro" option is selected, while "Casado", "Divorciado", and "Outros" are unselected.

Fonte: Do autor (2014).

2.3.5.1 Caixa de seleção

São caixas que possibilitam ao usuário selecionar uma ou mais opções. Para criar uma caixa de seleção, atribua o valor checkbox ao comando type. Também responderá a uma pergunta que pode ter várias respostas.

Exemplo 13: Formulário com checkbox

```
<form>
  De que cores você gosta? <br />
  <input type="checkbox" name="azul" value="azul"/>Azul <br />
  <input type="checkbox" name="amarelo" value="amarelo"/>Amarelo <br />
  <input type="checkbox" name="verde" value="verde"/>Verde <br />
</form>
```

Figura 2.24 Exemplo 13 — Formulário com checkbox

A screenshot of a web browser window titled "UOL Mail - Entrada (1 não)" with the URL "file:///C/Users/adri/Desktop/Livro%20de%20HTML/formulario%20com%20checkbox.html". The page title is "CRIANDO FORMULÁRIOS". It contains the question "De que cores você gosta?" followed by three checkboxes: "Azul", "Amarelo", and "Verde". The "Verde" checkbox is checked, while "Azul" and "Amarelo" are unselected.

Fonte: Do autor (2014).

2.3.5.2 Menu drop-down

Tem a função semelhante aos botões rádio, pois permite apenas a escolha de uma única opção. Sua utilização é recomendada quando relacionamos uma grande lista de opções e não há muito espaço no formulário. Para visualizar as opções da lista, o usuário se utilizará de uma seta onde está a lista de opções. Para criar um menu drop-down utilize o tag <select>, e para criar cada uma das opções utilize o tag <option>.

Exemplo 14: Formulários drop down

```
<html>
<head>
<title> Formulários</title>
</head>
<body>
<h3><center>CRIANDO SELEÇÃO</centre></h3><p>
<form>
Nome completo: <input type=text name = nome completo> <br>
Estado:<select name=estado>
<option>Bahia
<option>Ceará
<option>Paraná
<option> São Paulo
</select>
</form>
</body>
</html>
</form>
```

Figura 2.25 Exemplo 14: Formulários drop down



Fonte: Do autor (2014).

2.3.5.2.1 Áreas de texto

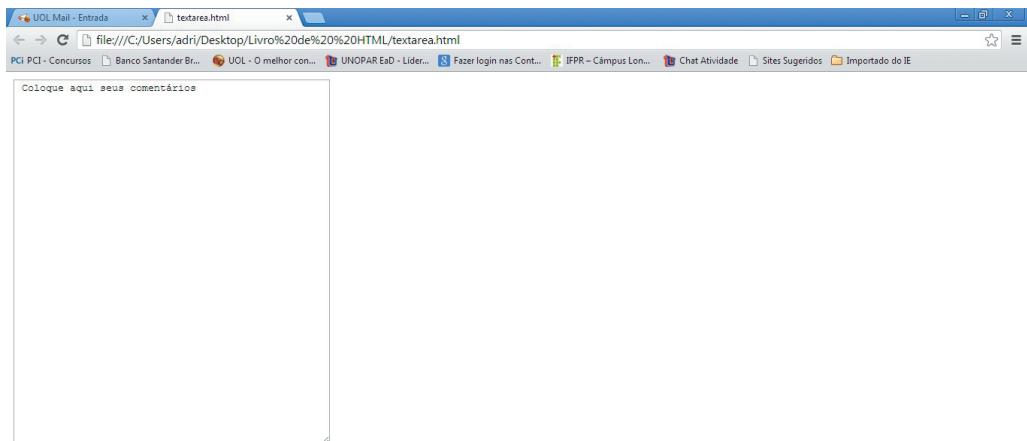
Este tipo de campo é uma caixa de digitação onde o usuário pode digitar nessa área de texto, delimitando o número de linhas "rows=n" e o número de colunas "cols=n", onde n é um número que você determina.

Áreas de texto são ideais para formulários de e-mail e comentários por comporem uma quantidade de texto maior que os campos de texto. Para criar uma área de texto utilize o tag <textarea>.

Exemplo 15: Área de texto

```
<form>
<textarea name="Comentários" rows="30" cols="50"> Coloque aqui seus comentários
</textarea>
</form>
```

Figura 2.26 Exemplo 15: Área de texto



Fonte: Do autor (2014).

2.3.5.3 Limpando dados do formulário

É possível limpar todos os dados preenchidos do formulário através de um único botão. Em geral, esse botão está localizado no final do formulário e é conhecido como reset.

Sua sintaxe é <input type = "reset" value = "nome do botão">.

Observando que quando o parâmetro value não for especificado, o botão será representado pelo nome reset.

2.3.5.4 Enviando os dados do formulário

Para enviar os dados do formulário para o servidor, a linguagem HTML utiliza o botão "submit" que acessa o url do programa script especificado no comando form e envia os dados do formulário.

A sintaxe é:

<input type="submit" value="nome do botão">

Exemplo 16: Botões Reset e Submit

```

<html>
<body>
<h1> Cadastro </h1>
<form>
  Primeiro nome: < input type = "text" name = "primeironome" ><br><br>
  Sobrenome: < input type = "text" name = "sobrenome" ><br><br>
  < input type = "reset" value = "Limpar" >
  < input type = "submit" value = "Enviar" >
</form>
</body>
</html>
  
```

Figura 2.27 Exemplo 16: Botões Reset e Submit

Cadastro

Primeiro nome:

Sobrenome:

Fonte: Do autor (2014).

2.3.5.4.1 Ações e métodos de envio dos formulários

Quando o botão submit é pressionado, as informações do formulário são enviadas. Devemos informar a ele o método de envio das informações e também a página que receberá as informações para a manipulação.

Então, para informar ao navegador o método de envio dos dados do formulário utilize o comando method; esse comando pode assumir os seguintes valores:

- └ POST — envia os dados do formulário de forma invisível ao usuário; as informações são enviadas em um bloco de dados separado da url;
- └ GET — envia os dados de forma visível ao usuário, onde as informações são enviadas como se fizessem parte da url.

2.3.5.5 Alinhando campos do formulário

Alinhar os campos de um formulário na tela é algo que não se consegue fazer naturalmente, já que cada campo ocupa um espaço diferente da tela. Para resolver esse problema de estética podemos criar os campos de formulário dentro de uma tabela.

Exemplo 17: Formulário dentro de uma tabela

```
<head>
<title> Formulários</title>
</head>
<body>
<h3><center>Cadastro de pessoa física</center></h3><p>
<form>
<table>
<tr>
<td>Nome:</td>
<td><input type=text name=nomecompleto size=25>
<td>Profissão:</td>
<td><input type=text name=profis size=20>
<td>Endereço:</td>
<td><input type=text name=ender size=20>
</tr>
<tr>
<td>Bairro:</td>
<td><input type=text name=bairro></td>
<td>CEP:</td>
<td><input type=text name=cep maxlength=9 size=9></td>
<td>Cidade:</td>
```

```

<td><input type=text name=cidade size=20></td>
</tr>
</table>
<table>
<tr>
<td> Estado Civil:</td>
<td> Solteiro <input type=radio name=estcivil>
Casado<input type=radio name=estcivil>
Divorciado<input type=radio name=estcivil>
Outros<input type=radio name=estcivil></td>
</tr>
</table>
</form>
</body>
</html>

```

Figura 2.28 Exemplo 17 — Formulário dentro de uma tabela

The screenshot shows a web browser window with the title 'Formulários'. The address bar contains the URL 'file:///C:/Users/adri/Desktop/Livro%20de%20HTML/formularios%20com%20tabelas.html'. The main content is a form titled 'Cadastro de pessoa física'. It includes fields for Name (Nome: Adriane Loper), Profession (Profissão: Engenheira), Address (Endereço: Rua XYZ, 1234), Neighborhood (Bairro: San Remo), Zip Code (CEP: 86056-222), City (Cidade: Londrina), and State Civil (Estado Civil: Solteiro). Below these fields are radio buttons for marital status: Casado (checked), Divorciado, and Outros.

Fonte: Do autor (2014).

📝

Atividades de aprendizagem

Utilize o enunciado abaixo para responder às próximas questões:

“Vamos trabalhar com uma dieta balanceada e assim aplicar nossos conhecimentos.”

1. Construa um campo de texto com nome, sobrenome e telefone.
2. Construa um Radio Button com 8 frutas.

Seção 3 **Frames, animação simples (marquee) em html**

Nesta seção trataremos dos frames, que são quadros que dividem a janela do browser, fazendo com que seja mostrada mais de uma página ao mesmo tempo. Necessitamos de planejamento para criação desses quadros, tais como quantos quadros terá a janela, qual será a disposição deles e quais os seus conteúdos.

3.1 Frames <frameset>

Os frames devem ser criados em um documento html específico que não possui os tags `<body></body>`; em vez desses, usamos os tags `<frameset></frameset>`. Dentro dessa estrutura são definidos todos os atributos dos quadros que serão criados.

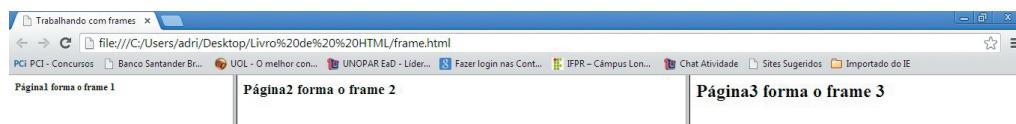
Basicamente, uma página em frames é constituída de dois elementos:

- uma página chamada frameset, que contém em seu código tags que especificam a divisão das frames dentro da janela do browser;
- as páginas internas em si, chamadas de frames, carregadas de acordo com as instruções contidas no código da frameset.

Exemplo 1: Frames

```
<html>
<head>
<title> Trabalhando com frames </title>
</head>
<frameset cols=200,400,300>
    <frame src=pagina1.htm>
    <frame src=pagina2.htm>
    <frame src=pagina3.htm>
</frameset>
<html>
```

Figura 2.29 Exemplo 1: Frames



Fonte: Do autor (2014).

No código anterior, a linha `<frameset cols=200,400,300>` define que serão criados três quadros na janela, um com 200 pixels, o segundo com 400 pixels e o último com 300 pixels.

A linha `<frame src=pagina1.htm>` especifica que uma página chamada `página1.htm` será mostrada no primeiro frame criado, o frame de 200 pixels.

A página1 tem a seguinte estrutura:

```
<html>
<body>
<h5>Página1 forma o frame 1</h5>
<p>
</body>
</html>
```

A linha `<frame src=pagina2.htm>` especifica que uma página chamada página2.htm será mostrada no segundo frame criado, o frame de 400 pixels.

A página2 tem a seguinte estrutura:

```
<html>
<body>
<h3>Página2 forma o frame 2</h3>
<p>
</body>
</html>
```

A linha `<frame src=pagina3.htm>` especifica que uma página chamada página3.htm será mostrada no segundo frame criado, o frame de 300 pixels.

A página3 tem a seguinte estrutura:

```
<html>
<body>
<h2>Página3 forma o frame 3</h2>
<p>
</body>
</html>
```

O tag frameset é bastante completo. Possui atributos internos e também possui tags inteiros também internos. Esses atributos são:

- └ **atributo COLS** → determina divisões em colunas. Use-o da seguinte maneira: separe por vírgula os comprimentos de cada coluna da página, em pixels ou em valores porcentuais ou ainda use * para que o browser determine o tamanho de acordo com o tamanho da janela;
- └ **atributo ROWS** → determina divisões em linhas. Use-o da seguinte maneira: separe por vírgula as alturas de cada linha da página, em pixels ou em valores porcentuais ou ainda use * para que o browser determine o tamanho de acordo com o tamanho da janela;
- └ **atributo FRAMESPACING** → determina o espaçamento entre cada frame (em pixels);
- └ **atributo FRAMEBORDER** → determina se haverá ou não bordas entre as frames (os valores deste atributo são fixos, ou seja, digite 1 para inserir bordas ou 0 para retirá-las);
- └ **atributo FRAMESPACING** → determina o espaçamento entre cada frame (em pixels);
- └ **atributo FRAMEBORDER** → determina se haverá ou não bordas entre as frames (os valores desse atributo são fixos, ou seja, digite 1 para inserir bordas

ou 0 para retirá-las). Vale lembrar que este atributo sobrepõe-se aos valores de FRAMEBORDER que sejam inseridos na frameset;

- └ **atributo MARGINHEIGHT** → especifica a altura das margens superior e inferior do frame em pixels;
- └ **atributo MARGINWIDTH** → especifica a altura das margens direita e esquerda do frame em pixels;
- └ **atributo NAME** → atribui um nome para o frame, de maneira que possa ser identificado e localizado para carregar documentos;
- └ **atributo NORESIZE**: este atributo dentro do tag impede o usuário de modificações;
- └ **atributo SCROLLING**: atribui valores YES, caso você queira que o frame possua barras de rolagem ou NO, em caso contrário (atribuindo YES, a frame só apresentará barra de rolagem caso seja realmente necessário em função do conteúdo. do frame);
- └ **atributo SRC**: define o caminho da página HTML que será exibida no frame.



Para saber mais

Recomendo como leitura o Capítulo 2 (HTML: Linguagem de marcação) do livro:

SILVA, Maurício Samy Silva. **Criando sites com HTML**: sites de alta qualidade com HTML e CSS. São Paulo: Novatec, 2008.

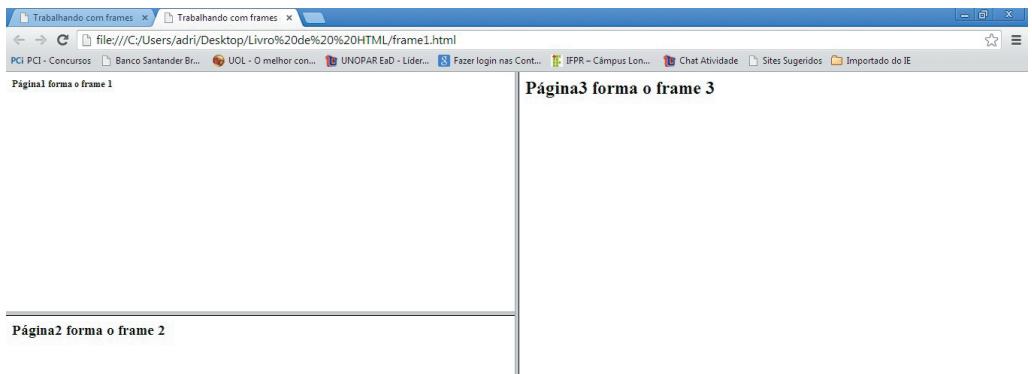
3.1.1 Utilizando rows e cols simultaneamente

Podemos criar frames combinando os dois parâmetros de linha e colunas. Para isso, basta criar uma estrutura de frames dentro de outra.

Exemplo 2: Frames usando cols e rows ao mesmo tempo

```
<html>
<head>
<title> Trabalhando com frames </title>
</head>
<frameset cols=50%,50%>
    <frameset rows=50%,50%>
        <frame src=pagina1.htm>
        <frame src=pagina2.htm>
    </frameset>
    <frame src=pagina3.htm>
</frameset>
<html>
```

Figura 2.30 Exemplo 2 — Frames usando cols e rows ao mesmo tempo



Fonte: Do autor (2014).

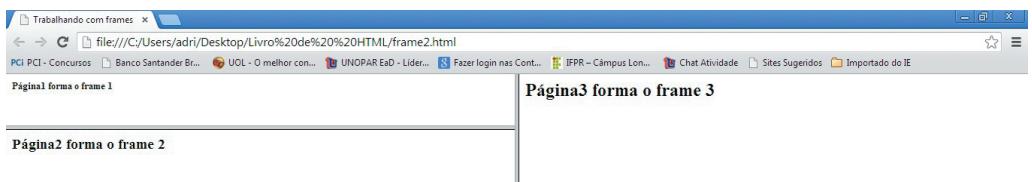
3.1.2 Utilizando barras de rolagem

As barras de rolagem nos frames são apresentadas sempre que o conteúdo da página não cabe no frame. Também podemos alterar o tamanho dos quadros arrastando as bordas. Para isso, usamos dois comandos, scrolling para desativar as barras de rolagem e noresize para impedir que o frame tenha seu tamanho alterado pelo usuário através da movimentação das bordas.

Exemplo 2: Frames usando cols e rows ao mesmo tempo

```
<html>
<head>
<title> Trabalhando com frames </title>
</head>
<frameset cols=50%,50%>
    <frameset rows=50%,50%>
        <frame src=pagina1.htm scrolling=no>
        <frame src=pagina2.htm>
    </frameset>
    <frame src=pagina3.htm noresize>
</frameset>
<html>
```

Figura 2.31 Exemplo 3 — Frames usando barras de rolagem



Fonte: Do autor (2014).

E assim podemos trabalhar para distribuir melhor os conteúdos mostrados nas páginas web.

3.2 Marquee

Um efeito muito interessante no HTML é a possibilidade de colocar elementos, imagens e textos em movimento, criando uma animação e utilizando um simples tag e alguns comandos.

O tag que tem a função de colocar movimento no HTML é o tag marquee.

Exemplo:

```
<marquee>
    Aqui vai o conteúdo que ficará em movimento!
</marquee>
```

3.2.1 Parâmetros

Alguns parâmetros de controle podem ser aplicados o tag marquee para melhorar o efeito.

3.2.1.1 Align (posição)

Especifica como o texto marquee será alinhado. Suas opções são:

- _| **Top** — Alinha o texto pela parte superior do marquee;
- _| **Middle** — Alinha o texto pela parte central do marquee;
- _| **Bottom** — Alinha o texto pela parte inferior do marquee.

3.2.1.2 Behavior (tipo)

Este comando possibilita o controle do efeito de movimento. Seus possíveis valores são:

- _| **alternate**: O conteúdo vai de um lado para outro, e assim sucessivamente.

Exemplo:

```
<marquee behavior="alternate">
Isto é um marquee....
</marquee>
```

- _| **slide**: O conteúdo sai de um lado, predefinido ou não, e para outro.

Exemplo:

```
<marquee behavior="slide">
Isto é um marquee....
</marquee>
```

- _| **scroll**: O conteúdo fica deslizando sempre no mesmo sentido.

Exemplo:

```
<marquee behavior="slide">
Isto é um marquee....
</marquee>
```

↳ **scrollamount:** Comando que permite o controle da velocidade do conteúdo em movimento. Aceita como valor números maiores que zero.

Exemplo:

```
<marquee scrollamount="10">
Isto é um marquee.....
</marquee>
```

3.2.1.3 Direction (direção)

Indica para qual direção de deslocamento do texto. Seus possíveis valores são:

↳ **down:** O conteúdo desliza de cima para baixo.

Exemplo:

```
<marquee direction="down">
Isto é um marquee.....
</marquee>
```

↳ **left:** O conteúdo desliza da direita para a esquerda.

Exemplo:

```
<marquee direction="left">
Isto é um marquee.....
</marquee>
```

↳ **right:** O conteúdo desliza da esquerda para a direita.

Exemplo:

```
<marquee direction="right">
Isto é um marquee.....
</marquee>
```

↳ **up:** O conteúdo desliza de baixo para cima.

Exemplo:

```
<marquee direction="up">
Isto é um marquee.....
</marquee>
```

3.2.1.4 Bgcolor (cor)

Especifica uma cor de fundo para o marquee. O padrão é branco.

3.2.1.5 Height (n)

Especifica a altura do marquee em pixels ou porcentagem.

3.2.1.6 Width (n)

Especifica a largura do marquee em pixels ou porcentagem.

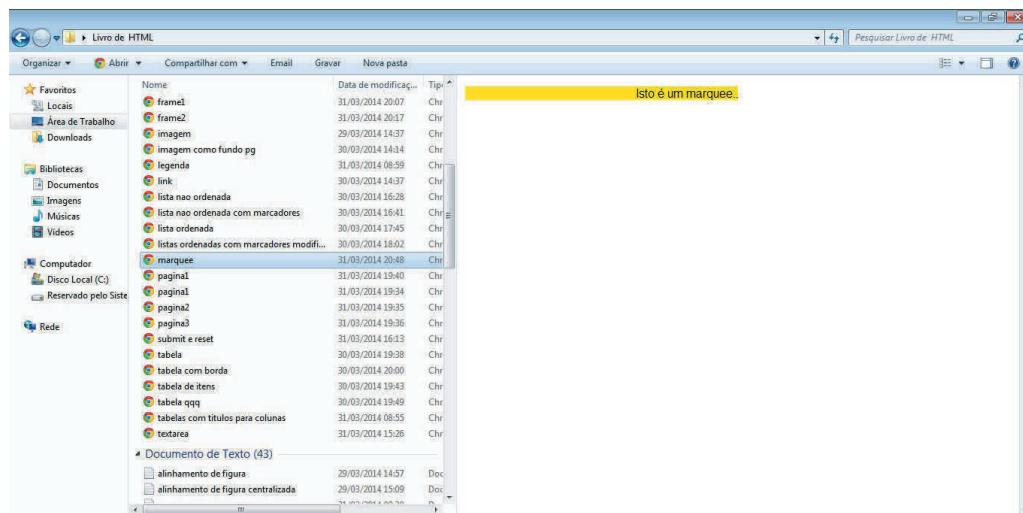
3.2.1.7 Loop (n)

Especifica quantas vezes o marquee será executado. Se for indicado “infinite”, ele será executado continuamente.

Exemplo 4: Marquee

```
<marquee height=10 width=50% behavior=scroll direction=right bgcolor=yellow loop=infinite>
Isto é um marquee....
</marquee>
```

Figura 2.32 Exemplo 4: Marquee



Fonte: Do autor (2014).

📝
Atividades de aprendizagem

1. Crie um documento HTML em FRAME, de tal modo que, ao acessar o documento, mostre a seguinte página:

```

graph TD
    Frame["<!DOCTYPE html><html><head></head><body><div style='border: 1px solid black; min-height: 100px; min-width: 100px;'>Pagina1.html</div><div style='border: 1px solid black; min-height: 100px; min-width: 100px;'>Pagina2.html</div><div style='border: 1px solid black; min-height: 100px; min-width: 100px;'>Pagina3.html</div></body></html>"]
    
```

2. Faça um marquee com a UNOPAR.



Questões para reflexão

Como pudemos observar, os comandos de HTML que utilizamos neste capítulo estão todos configurados na língua inglesa, por exemplo, as cores, as direções. Assim sendo, questiono: será que não conseguimos uma linguagem para web totalmente nacional?



Fique ligado!

A linguagem de marcação HTML está presente na maioria das páginas construídas. Mas, como em qualquer programação, temos outras versões com maiores recursos e facilidade para nossa estruturação. Assim sendo, atualize-se sempre.



Para concluir o estudo da unidade

Nas seções desta unidade foram apresentadas diversas formas de trabalho com HTML, para construirmos nossas páginas web de maneira que fiquem claras, concisas, rápidas e que transmitam de maneira correta a informação que estamos tentando passar ao usuário.



Atividades de aprendizagem da unidade

1. Crie uma página e insira a bandeira de seu estado como link — por exemplo, eu inserirei a do Paraná.
2. Crie um link para e-mail com seu nome e um provedor.
3. Monte um checkbox com 8 verduras e legumes, em que você escolherá 4 deste grupo de alimentos para nossa futura dieta balanceada.
4. Vamos trabalhar com uma tabela dentro de nosso formulário de 3 frutas e 3 legumes ou verduras.
5. Insira os botões de reset e submit para podermos enviar nossa dieta.

Referências

CARDOSO, M. **Desenvolvendo web para o ensino superior.** Rio de Janeiro: Axcel Books do Brasil, 2004.

OLIVIERO, C. A. J. **Faça um site:** JavaScript: orientado por objeto. 6. ed. São Paulo: Erica, 2007.

NIELSEN, Jakob; LORANGER, Hoa. **Usabilidade na web.** Projetando Websites com qualidade. Rio de Janeiro: Campus, 2007.

Rodrigues, Andrea. **Desenvolvimento para Internet.** Curitiba: LT, 2010.

SILVA, Maurício Samy. **Criando sites com HTML:** sites de alta qualidade com HTML e CSS. São Paulo: Novatec, 2008.

HTML 5

Danilo Augusto Bambini Silva

Objetivos de aprendizagem: Nesta unidade você vai conhecer as novidades da versão 5 do HTML e aprofundar nas novas tags HTML que visam organizar o conteúdo de uma página de acordo com o que cada parte representa, ou seja, de acordo com sua semântica.

↳ **Seção 1: Multimedia**

Estudaremos os novos recursos multimídia que proporcionam a inclusão de áudio e vídeo à página HTML de maneira muito simples e fácil.

↳ **Seção 2: Formulários**

Vamos aprender os novos recursos para criação de formulários na web.

↳ **Seção 3: Semântica**

Vamos entender o que é semântica, os benefícios da aplicação dela no código HTML e vamos estudar como esses recursos devem ser utilizados.

Introdução ao estudo

O HTML 5 é o padrão de HTML mais recente criado pela W3C e lançado em 2012. Apesar de ele substituir os padrões HTML 4, o XHTML e o HTML DOM Level2, ele procura manter a compatibilidade com sites escritos utilizando esses padrões. Além disso, ele visa proporcionar suporte total e integração com CSS e Javascript e eliminar a necessidade de plugins para muitas situações comuns, incorporando nativamente vários recursos que antes dependiam de um plugin. E se ainda assim você precisar de um plugin, a forma de incorporá-lo no seu documento foi padronizada e simplificada.

Ademais, ele é desenvolvido para funcionar da mesma forma em diferentes dispositivos, como PC, Tablet, Smartphone ou Smart TV.

O HTML 5 foi lançado em 2012 como um padrão a ser adotado pelos navegadores. Só então os navegadores começaram a adotar o novo padrão, incorporando-o em suas novas versões. Hoje, em 2014, ainda existem vários recursos que não foram adotados por alguns navegadores.

Seção 1 **Multimedia**

Vamos começar apresentando o básico para iniciar uma página com HTML 5, ou seja, a sua estrutura mínima, começando pela declaração DOCTYPE.

1.1 A declaração <!DOCTYPE>

A declaração DOCTYPE não é um tag HTML, ela é uma declaração de qual o tipo do documento. Ela deve estar no início de todo documento HTML para que o navegador possa interpretá-lo corretamente.

Nas versões anteriores do HTML a declaração DOCTYPE permitia que cada programador pudesse personalizar a linguagem HTML. Apesar de ser um recurso interessante, ele adicionava muita complexidade ao código, dificultando a leitura e a manutenção do documento por outros programadores.



Questões para reflexão

Neste ponto você deve estar se perguntando o que o programador vai fazer se ele quiser personalizar o seu HTML. Ele não pode fazer mais?

Pode sim! Só que de uma forma mais simples e organizada. Mais adiante, no tópico sobre novidades, vamos estudar o atributo data-* que foi adicionado na versão 5.

Para efeito de comparação, o Quadro 3.1 mostra as três formas mais simples e mais utilizadas de DOCTYPE no HTML 4. Essas formas devem ser utilizadas quando você **não quer** personalizar a linguagem.

Quadro 3.1 DOCTYPE no HTML 4

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

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

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

Fonte: W3Shcool (2014).

Não vamos entrar em detalhes sobre como funcionava esse recurso no HTML 4, o que precisamos saber é que isso ficou mais simples no HTML 5. O Quadro 3.2 mostra como o DOCTYPE deve ser utilizado no HTML 5.

Quadro 3.2 DOCTYPE no HTML 5

```
<!DOCTYPE html>
```

Fonte: W3Shcool (2014).

Essa declaração deve ser adicionada no início de todo documento para que o navegador saiba que o documento a seguir utiliza o HTML 5.

1.2 Estrutura mínima

A estrutura mínima que um documento HTML 5 deve ter continua a mesma, como mostra o Quadro 3.3.

Quadro 3.3 Estrutura mínima do HTML 5

```
<!DOCTYPE html>
<html>
<head>
<met charset="UTF-8">
    <title>Título do documento</title>
</head>
<body>
    Conteúdo do documento...
</body>
</html>
```

Fonte: Adaptado de W3Shcool (2014).

1.3 Multimídia

Multimídia é praticamente tudo que você pode ver ou ouvir. Por exemplo, texto, imagens, música, sons, vídeos, animações e gráficos.

Os primeiros navegadores de Internet tinham suporte apenas para texto em uma única fonte e uma única cor. As primeiras melhorias vieram com suporte a cores, fontes e imagens.

Com o tempo, foram surgindo plugins que permitiam a inclusão de outros tipos de conteúdo multimídia, como sons, animações e vídeos. Então, quando um usuário acessa uma página que tem conteúdo multimídia incorporado por meio de plugin, o usuário só terá acesso ao conteúdo depois de instalar no seu computador o plugin em questão. Entretanto, alguns plugins podem não ter opção de instalar em um sistema operacional específico ou podem não funcionar em um determinado navegador. Além disso, a instalação de um plugin pode não ser uma tarefa trivial para um usuário leigo.

Era assim antes do HTML 5. Agora a W3C padronizou quais seriam os formatos de áudio e vídeo que todos os navegadores deveriam suportar e incluiu tags visando ao suporte nativo a áudio e vídeo nos navegadores sem depender de plugins.

1.4 Vídeo

Antes do HTML 5, para mostrar vídeos você tinha que pedir para o usuário instalar um plugin no navegador dele. Então, às vezes, a versão do navegador do usuário não aceitava o plugin que você utilizou. Além disso, o código necessário para embutir o vídeo no seu documento era diferente de um plugin para outro, ou seja, não existia um padrão.

E se para colocar um vídeo no seu site fosse tão fácil quanto colocar uma imagem? Pois no HTML 5 é tão simples quanto usar um tag! Apenas tenha o vídeo disponível em uma pasta do seu site e coloque no seu documento o tag<video> apontando para ele.

O Quadro 3.4 mostra um exemplo de código para incluir um vídeo em um documento.

Quadro 3.4 Exemplo de código para o tagvideo

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
    <title>Exemplo de código para a tagvideo</title>
</head>
<body>
    <video width="320" height="240" controls autoplay>
        <source src="videos/filme.mp4" type="video/mp4">
        <source src="videos/filme.ogg" type="video/ogg">
        Atualize seu navegador para ver este vídeo!
    </video>
</body>
</html>
```

Os principais atributos do tagvideo são:

- **width** e **height**: Definem respectivamente a largura e a altura do vídeo em pixels.
- **controls**: Define que é para mostrar os controles ao usuário, como play/pause e volume.
- **autoplay**: Define que o vídeo deve iniciar a sua reprodução assim que for carregado na página.



Para saber mais

Veja a lista completa de atributos na documentação oficial do HTML 5 no site da W3S:
[<http://www.w3schools.com/tags/ref_av_dom.asp>](http://www.w3schools.com/tags/ref_av_dom.asp).

Dentro do tagvideo deve ter um ou mais tags <source>. Cada tag <source> representa um arquivo de vídeo em determinado formato.

O padrão HTML 5 suporta apenas os formatos de vídeo MP4, WebM e Ogg. Essa limitação não deve ser vista como algo ruim, afinal, com menos formatos, praticamente todos os navegadores tenderão a suportar os mesmos formatos, facilitando muito na hora de disponibilizar um conteúdo multimedia, afinal basta que você disponibilize o seu vídeo nesses poucos formatos e todos os navegadores o acessarão. Além disso, existe uma grande tendência à adoção do MP4 por todos os navegadores. O Quadro 3.5 tem uma pequena descrição de cada um dos formatos de vídeo mais utilizados atualmente e que nos ajudam a compreender por que esses três formatos foram os escolhidos.

Quadro 3.5 Formatos de vídeo mais utilizados atualmente

Formato	Arquivo	Descrição
AVI	.avi	AVI (AudioVideoInterleave) foi desenvolvido pela Microsoft e é acessível a partir de todos os computadores com Windows. É muito utilizado em câmeras de vídeo e equipamentos de TV. Entretanto, nem sempre tem suporte para computadores que não utilizam Windows.
WMV	.wmv	WMV (Windows Media Video) também foi desenvolvido pela Microsoft e é acessível a partir de todos os computadores com Windows. É muito utilizado em câmeras de vídeo e equipamentos de TV. Entretanto, nem sempre tem suporte para computadores que não utilizam Windows.
QuickTime	.mov	QuickTime foi desenvolvido pela Apple e é acessível a partir de todos computadores Apple. É muito utilizado em câmeras de vídeo e equipamentos de TV. Entretanto, nem sempre tem suporte para computadores que não são da Apple.
RealVideo	.rm .ram	RealVideo foi desenvolvido pela Real Media com objetivo de permitir streaming de vídeo em Internet com pouca largura de banda. Ainda é utilizado para vídeos online e Internet TV, entretanto, como esse formato prioriza consumir pouca largura de banda, a qualidade também é baixa.

continuação

Flash	.swf .flv	Flash foi desenvolvido pela Macromedia. Ele requer a instalação de um plugin para funcionar.
Ogg	.ogg	TheoraOgg foi desenvolvido pela Xiph.Org Foundation. É um formato código aberto, livre para utilização. A Xiph.Org Foundation (https://www.xiph.org/) é uma comunidade de software livre, com vários projetos que tem como objetivo proteger a sua tecnologia do domínio privado, garantindo a sua utilização livre para todos.
WebM	.webm	WebM é um projeto (www.webmproject.org) criado pelos gigantes da Internet: Mozilla, Opera, Adobe, and Google. É um formato código aberto, livre para utilização. Foi criado com objetivo de ser um formato ideal para vídeos tocados a partir de servidores on-line.
MPEG	.mpg .mpeg	MPEG, desenvolvido pelo Moving Pictures Expert Group. É um formato código aberto, livre para utilização. Já foi o formato de vídeo mais popular na Internet. Ele é suportado pelos principais navegadores de Internet (Chrome, Firefox, Safari e IE). Entretanto, o próprio MPEG já desenvolveu o MPEG4 que é uma melhoria do formato anterior e visa substituí-lo.
MPEG-4 ou MP4	.mp4	MP4 é o formato que será mais utilizado na Internet a nos próximos anos. Ele é suportado pelos principais navegadores de Internet (Chrome, Firefox versão 22, Safari e IE). O YouTube recomenda o uso do MP4 e ele é utilizado pela maioria das novas câmeras de vídeo e equipamentos de TV.

Fonte: Adaptado de W3Schools (2014).



Para saber mais

O formato MP4 atualmente não é suportado pelo Opera e pelo Firefox em versão anterior a 22. Já os formatos WebM e Ogg não são suportados pelos navegadores Internet Explorer e Safari. Por isso, atualmente é recomendável disponibilizar os vídeos em MP4 e Ogg ou em MP4 e WebM.

Os atributos do tagsource são:

- └ **src**: caminho e nome do arquivo de vídeo.
- └ **type**: tipo mime do arquivo.

No final da lista de tags<source> e antes de encerrar o tag<video> deve conter um texto que será mostrado quando o navegador não reconhecer nenhum dos formatos disponibilizados nos tagssource. Por exemplo, o quadro 3.4 mostra o texto: “Atualize seu navegador para ver este vídeo!”.

1.4.1 YouTube

Utilizar o YouTube para adicionar vídeos à sua página ainda é uma das formas mais fáceis de se fazer isso e é, de longe, a mais utilizada para documentos on-line.

Por isso, apesar de você não precisar de conhecimentos de HTML 5 para fazer isso, se o assunto é vídeo, temos que falar do YouTube.

Primeiro você deve escolher o vídeo que quer adicionar ou criar a sua conta e fazer upload do seu vídeo. Depois acesse o vídeo no YouTube e escolha a opção “Compartilhar” e, em seguida, a opção “Incorporar”. Então aparece uma caixa contendo o código que você deverá copiar e colar no seu documento HTML para ter o vídeo no seu site. Muito fácil, não é?

E tem mais, se o tamanho do vídeo não ficou do jeito que você queria, volte no YouTube, na opção “Compartilhar” e “Incorporar” e escolha o tamanho do vídeo e outras opções que o YouTube disponibiliza, e depois copie e cole o código gerado pelo YouTube no seu site.

1.4.2 Áudio

Antes do HTML 5 não existia um padrão para adicionar áudio a uma página. Agora para incluir áudio na sua página também ficou muito fácil! Inclusive é bem parecido com o tagvideo.

O Quadro 3.6 mostra um exemplo de código para incluir um áudio em um documento.

Quadro 3.6 Exemplo de código para o tagaudio

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de código para o tagaudio</title>
</head>
<body>
    <audiocontrols autoplay loop muted>
        <sourcesrc="musica.ogg" type="audio/ogg">
        <source src="musica.mp3" type="audio/mpeg">
        Atualize seu navegador para tocar esta música.
    </audio>
</body>
</html>
```

Os principais atributos do tagaudio são:

- └ **controls**: Define que é para mostrar os controles ao usuário, como play/pause e volume.
- └ **autoplay**: Define que o áudio deve iniciar a sua reprodução assim que for carregado na página.
- └ **loop**: Define que o áudio deve ser reiniciado sempre que chegar ao seu fim.
- └ **muted**: Define que o áudio deve ficar mudo. Entretanto, se o tagcontrols estiver ativo, o usuário pode clicar no controle de volume e ativá-lo quando quiser.



Para saber mais

Veja a lista completa de atributos do tagaudio na documentação oficial do HTML 5 no site da W3S, disponível em: <http://www.w3schools.com/tags/ref_av_dom.asp>.

Dentro do tagaudio deve ter um ou mais tags<source>. Cada tag<source> representa um arquivo de áudio em determinado formato. O primeiro formato que o navegador reconhecer é o que será reproduzido.

O padrão HTML 5 suporta apenas os formatos de vídeo MP3, Wav e Ogg. O Quadro 3.7 mostra quais os formatos de áudio que são suportados por cada navegador.

Quadro 3.7 Formatos de áudio e suporte do navegador

Navegador	MP3	Wav	Ogg
Internet Explorer	Tem suporte	Não suporta!	Não suporta!
Chrome	Tem suporte	Tem suporte	Tem suporte
Firefox	Tem suporte apenas na versão 21 do Firefox, rodando no Windows 7, 8, Vista e Android.	Tem suporte	Tem suporte
Safari	Tem suporte	Tem suporte	Não suporta!
Opera	Não suporta!	Tem suporte	Tem suporte

Fonte: W3Schools (2014).

Do Quadro 3.7 podemos concluir que para o áudio funcionar em todos os navegadores é necessário que ele esteja disponível no formato MP3 e em pelo menos mais um dos demais formatos: Wav ou Ogg.



Para saber mais

Sempre que você for incluir áudio como parte da sua página, lembre-se de que muitos usuários podem achar isso perturbador e podem deixar de acessar o seu site por esse motivo. É recomendado que você inclua sons em momentos que o usuário esteja esperando que um som seja tocado, por exemplo, quando o usuário clicou em algum lugar pedindo para ouvir a gravação.

1.5 Legendas para áudio ou vídeo

O HTML 5 prevê o uso de trilhas de legendas tanto para vídeos usando o tag “video” quanto para áudio usando o tag “audio”. Cada áudio ou vídeo pode ter quantas legendas quiser. Infelizmente, esse recurso ainda é suportado apenas por Chrome, Opera e Internet Explorer 10. Os navegadores Internet Explorer mais antigos, o Firefox e o Safari ainda não implementam esse recurso até o momento em que este livro foi escrito. Mas podem vir a implementar a qualquer momento; então, agora mesmo, enquanto você estuda este livro, é recomendável que verifique se eles já suportam esse recurso. Para isso acesse a página da W3S, que documenta esse recurso: <http://www.w3schools.com/tags/tag_track.asp>.

Para adicionar legendas a um vídeo inclua, dentro do tagvideo, tantos tags “track” quanto forem as possíveis legendas que deseja adicionar a seu vídeo.

O Quadro 3.8 mostra um exemplo de código HTML de um vídeo com legendas embutidas.

Quadro 3.8 Exemplo de código de vídeo com legendas em 3 línguas diferentes

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de código de vídeo com legendas em 3 línguas diferentes.</title>
</head>
<body>
    <videowidth="320"height="240"controls autoplay>
        <sourcesrc="videos/filme.mp4" type="video/mp4">
        <sourcesrc="videos/filme.ogg" type="video/ogg">
        <tracksrc="videos/legenda-pt-br.vtt" kind="subtitles" srclang="pt-br" label="Português" default>
        <tracksrc="videos/legenda-en.vtt" kind="subtitles" srclang="en" label="English">
        <tracksrc="videos/legenda-no.vtt" kind="subtitles" srclang="no" label="Norwegian">
        Atualize seu navegador para ver este vídeo!
    </video>
</body>
</html>
```

Os atributos do tagtrack são:

- └ **default**: Indica que aquela legenda é a padrão. Deve estar atribuído a apenas um dos tagtrack do mesmo vídeo.
- └ **src**: É um atributo obrigatório do tagtrack. Ele deve conter o caminho e o nome do arquivo com o texto da legenda.
- └ **kind**: Especifica o tipo de track que pode ser, se não for especificado será assumido o tipo “subtitles”:

- **subtitles**: Indica que é uma legenda para um vídeo. Utilizado quando para quando o usuário não entende a língua na qual o vídeo foi gravado, proporcionando a tradução através de legendas.
 - **captions**: Indica que é a transcrição de um diálogo e efeitos sonoros, utilizado para fins de acessibilidade para usuários surdos ou para tocar o vídeo em ambientes onde não é possível ouvir o áudio. Quando existe uma track com o tipo captions e o atributo “controls” foi utilizado, o navegador mostra um botão “CC”, que, quando clicado, habilita a exibição do texto dessa track.
 - **chapters**: Define os títulos dos capítulos do vídeo; é útil para auxiliar na navegação dentro do vídeo.
 - **descriptions**: Indica que o texto contém uma descrição textual do vídeo. Utilizado para que usuários cegos possam escutar a descrição do conteúdo do vídeo através do recurso de acessibilidade que lê a página HTML para o cego.
 - **metadata**: Indica que o conteúdo da track é apenas para acesso via programação javascript e, nesse caso, o texto não é mostrado ao usuário.
- ↳ **label**: Define o nome para o conteúdo da track. Em legendas, por exemplo, é comum que o label indique a língua da track. Apesar de esse atributo não ser obrigatório, é recomendado que seja utilizado sempre que o vídeo contém mais do que uma track, afinal esse label será utilizado pelo navegador para mostrar as possíveis “tracks” que o usuário pode escolher.
- ↳ **srlang**: Indica a língua do texto da track. Ele pode ser utilizado para qualquer tipo de track, entretanto, é obrigatório sempre que o tipo for “subtitles” e deve seguir o padrão BCP 47. Esse padrão define dentre outras coisas, as siglas para cada língua. Para consultar essa lista acesse: <<http://people.w3.org/rishida/utils/subtags/>>.

De acordo com a W3C, o formato de arquivo que deve ser utilizado para tracks é o WebVTT; ele é um formato criado especificamente para a finalidade de uso em páginas web com o tagtrack. O Quadro 3.9 mostra um exemplo de uma transcrição de um diálogo escrita em WebVTT.

Quadro 3.9 Exemplo de arquivo no formato WebVTT

```
WEBVTT

00:11.000-->00:13.000
<v RogerBingham>We are in New York City

00:13.000-->00:16.000
<v RogerBingham>We're actually at the Lucern Hotel, just down the street

00:16.000-->00:18.000
<v RogerBingham>from the American Museum of Natural History

00:18.000-->00:20.000
<v RogerBingham>And with me is Neil deGrasse Tyson
```

00:20.000-->00:22.000
 <v RogerBingham>Astrophysicist, Director of the Hayden Planetarium

00:22.000-->00:24.000
 <v RogerBingham>at the AMNH.

00:24.000-->00:26.000
 <v RogerBingham>Thank you for walking down here.

00:27.000-->00:30.000
 <v RogerBingham>And I want to do a follow-up on the last conversation we did.

00:30.000-->00:31.500align:end size:50%
 <v RogerBingham>When we e-mailed—

00:30.500-->00:32.500align:start size:50%
 <v NeildeGrasse Tyson>Didn't we talk about enough in that conversation?
 00:32.000-->00:35.500align:end size:50%
 <v RogerBingham>No! No nonono; 'cos 'cos obviously 'cos

00:32.500-->00:33.500align:start size:50%
 <v NeildeGrasse Tyson><i>Laughs</i>

00:35.500-->00:38.000
 <v RogerBingham>You know I'm so excited my glasses are falling off here.

O tagtrack também pode ser utilizado em conjunto com o tagaudio e tem exatamente as mesmas características do que quando combinado com o tagvideo. O Quadro 3.10 mostra um exemplo.

Quadro 3.10 Exemplo de código para o tagaudio com legenda

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
  <title>Exemplo de código para o tagaudio com legenda</title>
</head>
<body>
  <audiocontrols autoplay>
    <sourcesrc="musica.ogg" type="audio/ogg">
      <source src="musica.mp3" type="audio/mpeg">
        <tracksrc="videos/legenda-pt-br.vtt" kind="subtitles" srclang="pt-br" label="Português" default>
          <tracksrc="videos/legenda-en.vtt" kind="subtitles" srclang="en" label="English">
          <tracksrc="videos/legenda-no.vtt" kind="subtitles" srclang="no" label="Norwegian">
          Atualize seu navegador para ouvir esta música.
        </tracksrc="videos/legenda-no.vtt" kind="subtitles" srclang="no" label="Norwegian">
      </tracksrc="videos/legenda-en.vtt" kind="subtitles" srclang="en" label="English">
    </sourcesrc="musica.ogg" type="audio/ogg">
  </audiocontrols autoplay>
</body>
</html>
```

No exemplo do Quadro 3.10, a “música” será tocada automaticamente quando a página carregar (atributo autoplay) e o usuário terá visíveis os controles para controlar a música (atributo controls). Ela será no formato ogg ou o navegador não suportar no formato mp3. Por padrão, será mostrada uma legenda em português para o áudio e o usuário poderá escolher ver a legenda nos idiomas inglês ou norueguês. Se o navegador não suportar o tagaudio, em vez de tudo isso apenas será mostrado o texto “Atualize seu navegador para ouvir esta música”.

O Quadro 3.11 mostra outro exemplo de código com o tagvideo e com vários tipos de “tracks”.

Quadro 3.11 Exemplo de código para o tagvideo com legenda, captions, descriptions e metadata

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Video com legenda, captions, descriptions e metadata</title>
</head>
<body>
    <videocontrols>
        <sourcesrc="filme.ogg" type="video/ogg">
        <sourcesrc="filme.mp4" type="video/mp4">
            <trackkind="captions"src="sampleCaptions.vtt"srclang="en">
            <trackkind="descriptions"src="sampleDescriptions.vtt"srclang="en">
            <trackkind="chapters"src="sampleChapters.vtt"srclang="en">
            <trackkind="subtitles"src="sampleSubtitles_de.vtt"srclang="de">
            <trackkind="subtitles"src="sampleSubtitles_en.vtt"srclang="en" default>
            <trackkind="subtitles"src="sampleSubtitles_ja.vtt"srclang="ja">
            <trackkind="subtitles"src="sampleSubtitles_pt-br.vtt"srclang="pt-br">
            <trackkind="metadata"src="keyStage1.vtt"srclang="en" label="Key Stage 1">
            <trackkind="metadata"src="keyStage2.vtt"srclang="en" label="Key Stage 2">
            <trackkind="metadata"src="keyStage3.vtt"srclang="en" label="Key Stage 3">
        </video>
    </body>
</html>
```

No exemplo do Quadro 3.11, o arquivo “filme” será tocado no formato Ogg ou, se o navegador não suportar, no formato mp4. Se o usuário ativar o recurso de closed-captions, o texto do arquivo “sampleCaptions.vtt” será mostrado ao usuário. O vídeo poderá ser avançado ou retrocedido capítulo por capítulo, de acordo com os capítulos definidos no arquivo “sampleChapters.vtt”, além de mostrar ao usuário em destaque, o título de cada capítulo. O vídeo mostrará uma legenda em inglês por padrão e dará ao usuário a opção de escolher os idiomas dinamarquês (de), japonês (ja) e português brasileiro (pt-br). Além disso, estarão disponíveis ao javascript os dados dos arquivos “keystage1.vtt”, “keystage2.vtt” e “keystage3.vtt”.



Para saber mais

A W3C disponibiliza a documentação completa do formato WebVTT no seguinte site, acesse: <http://dev.w3.org/html5/webvtt/>.

1.6 Plugins

Também conhecido como HTML Helper, ele é um programa que deve ser instalado no computador do usuário e tem o propósito de estender as funcionalidades do HTML. Exemplos de plugins bem conhecidos são os Javaapplets e o Adobe Flash Player.

O propósito de um plugin pode variar muito, afinal existem muito poucas restrições sobre o que um plugin pode fazer; alguns exemplos são: procurar e eliminar vírus, verificar a sua identidade para dar acesso a um banco, adicionar reconhecimento biométrico a uma página, permitir o controle direto ao hardware conectado à máquina do cliente, por exemplo, impressoras fiscais e etiquetadoras.

Também existem plugins para tocar áudio e para tocar vídeo, entretanto, para isso é melhor utilizar os tagsaudio e o video do HTML 5.

Plugins podem ser adicionados a uma página por meio de tagembed ou object. Ambas as tags permitem a inclusão de plugins. Então para entender por que hoje temos duas tags que fazem a mesma coisa, vamos voltar um pouco na história dos navegadores. No HTML 4, o tagobject era o tag reconhecido pela W3C. Enquanto o tagembed era um tag proprietário inventado pela Netscape e adotado no Firefox, que não reconhecia o tagobject. A questão levantada pela Netscape é que apesar de o tagembed não ser reconhecido oficialmente, ele é mais fácil de utilizar e foi adotado por todos os navegadores.

Então a novidade do HTML 5 foi que a W3C reconheceu e oficializou o tagembed. E a outra novidade é que hoje o tagobject também funciona nas novas versões do Firefox.

Assim, enquanto os navegadores nem a W3C decidem qual das duas tags deve prevalecer, vamos estudar as duas.

1.6.1 Embed

O tagembed tem como objetivo definir um container para um plugin. Apesar de uma página HTML 4 com o tagembed funcionar em todos os navegadores, elas não são validadas pela W3C. Já em uma página HTML 5 os tagseembed são validados normalmente.

O Quadro 3.12 mostra um exemplo do uso do tagembed para adicionar um conteúdo em Adobe Flash Player ao documento.

Quadro 3.12 Exemplo de uso do tagembed

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso do tagembed</title>
</head>
<body>
<embed src="helloworld.swf" width="200" height="200" type="application/x-shockwave-flash">
</body>
</html>
```

Os atributos do tagembed são:

- └ **src**: Define o arquivo que contém o conteúdo a ser adicionado à página. Este é o único atributo obrigatório do tagembed.
- └ **width**: Define a largura em pixels que o conteúdo deve ocupar no documento. Se não for definido, a página carregará inicialmente sem reservar espaço para esse conteúdo e, em seguida, se ajustará para acomodar o conteúdo. Ou seja, apesar de ser um atributo opcional, o ideal é sempre definir essa altura para evitar que a página carregue “torta” e vá se ajustando conforme o conteúdo é carregado.
- └ **height**: Define a altura em pixels que o conteúdo deve ocupar no documento. Vale a mesma recomendação do atributo width, ou seja, mesmo que esse atributo seja opcional é recomendável sempre definir essa largura.
- └ **type**: Define o tipo mime do conteúdo. Ajuda o navegador a saber qual o plugin deve ser carregado para se encarregar do arquivo definido no atributo “src”. Se não for definido, o navegador procurará descobrir o tipo mime pela extensão do nome do arquivo. Esse tipo deve ser um dos tipos mime definidos na tabela “IANA MIME types”, disponível no endereço: <<http://www.iana.org/assignments/media-types/media-types.xhtml>>.

1.6.2 Object

O tagobject foi criada com objetivo de ser um contêiner para diferentes tipos de objetos:

- └ **Imagens**: Apesar de o tagobject ainda servir para colocar imagens, o tagimg foi criado em seguida e substituiu essa funcionalidade do tagobject, com a vantagem de ser bem mais simples de usar.
- └ **Outra página HTML**: O tagObject pode servir para incluir uma página html dentro da outra, entretanto, o tagiframe substitui essa funcionalidade do tagobject com a vantagem de ser mais simples de usar.

└ **Plugins:** O tagobject pode ter a mesma função do tagembed, ou seja, prover um recurso de incluir conteúdos que dependem de um plugin para funcionar.

O Quadro 3.13 mostra um exemplo simples de utilização do tagobject que adiciona um conteúdo Flash ao documento.

Quadro 3.13 Exemplo simples de utilização do tagobject

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo simples de utilização do tagobject</title>
</head>
<body>
<object width="400" height="400" data="helloworld.swf" type="application/x-shockwave-flash"></object>
</body>
</html>
```

Fonte: W3Schools (2014).

A seguir vamos estudar mais a fundo cada um dos atributos do tagobject.

└ **width:** Define a largura em pixels que o conteúdo deve ocupar no documento. Se não for definido, a página carregará inicialmente sem reservar espaço para esse conteúdo e, em seguida, se ajustará para acomodar o conteúdo. Ou seja, apesar de ser um atributo opcional, o ideal é sempre definir essa altura para evitar que a página carregue “torta” e vá se ajustando conforme o conteúdo é carregado.

└ **height:** Define a altura em pixels que o conteúdo deve ocupar no documento. Vale a mesma recomendação do atributo width, ou seja, mesmo que esse atributo seja opcional é recomendável sempre definir essa largura.

└ **classid:** Código do executável do plugin no registro do windows. Esse atributo não é suportado no HTML 5. Mesmo assim ele ainda é utilizado para que o conteúdo funcione nas versões anteriores do IE.

└ **align:** Define o alinhamento do elemento object em relação ao conteúdo ao seu redor. Esse atributo não é suportado no HTML 5. Mesmo assim ainda é utilizado para que o conteúdo funcione adequadamente nas versões anteriores do IE. Os valores possíveis para este atributo são:

- **left:** Alinha o objeto para a esquerda.
- **right:** Alinha o objeto para a direita.
- **middle:** Alinha o objeto no centro.
- **top:** Alinha o objeto no topo.
- **bottom:** Alinha o objeto pela sua parte inferior.

- └ **data**: define o caminho e nome do arquivo que contém o conteúdo a ser carregado. É equivalente ao atributo src do tagembed.
- └ **type**: Define o tipo mime do conteúdo. Ajuda o navegador a saber qual o plugin deve ser carregado para se encarregar do arquivo definido no atributo "data". Se não for definido, o navegador procurará descobrir o tipo mime pela extensão do nome do arquivo. Esse tipo deve ser um dos tipos mime definidos na tabela "IANA MIME types", disponível no endereço: <<http://www.iana.org/assignments/media-types/media-types.xhtml>>.

Dentro do tagobject podem conter um ou mais tags param. Esses tags têm por finalidade passar parâmetros adicionais ao plugin que vai carregar o conteúdo.

Os tags param podem ter apenas dois atributos:

- └ **name**: Indica o nome do parâmetro.
- └ **value**: Indica o valor do parâmetro.

Os nomes e os valores, que podem ser utilizados, dependem do plugin que está sendo utilizado. Um plugin pode exigir alguns parâmetros ou não.

Outra coisa que pode conter dentro do tagobject é um conteúdo que vai aparecer, caso o navegador não tenha suporte para o plugin.

O Quadro 3.14 mostra um exemplo de uso do tagobject para tocar um áudio com um tag param "autoplay" com o valor "true", indicando que o áudio deverá iniciar a sua reprodução assim que a página for carregada. Este é apenas um exemplo simples de uso do tag "param"; na prática, você não deve utilizar o tagobject para tocar áudio, prefira usar o tagaudio pra isso.

Quadro 3.14 Exemplo de uso do tag param

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso do tag param</title>
</head>
<body>
<object data="horse.wav">
<param name="autoplay" value="true">
Atualize o seu navegador para poder tocar o áudio.
</object>
</body>
</html>
```

O Quadro 3.15 mostra o tagobject recomendado pela Adobe para adicionar conteúdo Flash a uma página.

Quadro 3.15 Recomendação da Adobe para inclusão de conteúdo Flash

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-  
444553540000" width="550" height="400" id="movie_name" align="middle">  
    <param name="movie" value="movie_name.swf"/>  
    <!--[if !IE]>-->  
    <object type="application/x-shockwave-flash" data="movie_name.  
swf" width="550" height="400">  
        <param name="movie" value="movie_name.swf"/>  
        <!--<![endif]-->  
        <a href="http://www.adobe.com/go/getflash">  
              
        </a>  
    <!--[if !IE]>-->  
    </object>  
    <!--<![endif]-->  
</object>
```

Fonte: Adobe Systems Incorporated (2014).

No exemplo anterior existe um tratamento para tratar as diferenças de interpretação do HTML que o IE tem em relação aos demais browsers. E também ao invés de incluir um texto para avisar que o navegador não suporta o plugin, ele inclui uma imagem do site da adobe que pede para instalar o flash com um link direto para a página de download do flash.



Atividades de aprendizagem

1. Como atividade deste tópico, crie uma página HTML 5 utilizando a estrutura mínima apresentada nesta seção e adicione a ela um vídeo.
2. Em seguida, teste modificar as propriedades width e height.
3. Experimente também como vai funcionar com o atributo autoplay e sem ele; faça o mesmo para o atributo controls.
4. Para finalizar a atividade, inclua na página um áudio e teste trocar os atributos mostrados neste tópico.

Seção 2 Formulários

Nesta seção veremos as facilidades e os novos recursos que o HTML 5 acrescentou nos formulários HTML. Esses recursos dispensam o uso de várias rotinas javascript muito comuns em formulários HTML 4, facilitando e agilizando o trabalho de desenvolvimento web.

2.1 Campo e-mail

Uma novidade muito útil para formulários html são os novos tipos de campos de entrada de dados (tag input) criados no HTML 5. Por exemplo, quando você tinha um campo para o usuário digitar o seu e-mail, era utilizado um código html semelhante ao código do Quadro 3.16.

Quadro 3.16 Título

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
    <title>Exemplo de código para entrada de dados HTML 4</title>
</head>
<body>
<form action="recebe_dados_email.php">
    E-mail:<input type="text" name="email">
    <input type="submit" value="Confirmar">
</form>
</body>
</html>
```



Para saber mais

Uma sugestão pra testar o formulário é utilizar o TryIt da W3School, que está disponível em http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_form_submit.

Então era necessário escrever um código em javascript para verificar se o valor digitado no campo era um endereço de e-mail ou não e avisar ao usuário de que a informação que ele digitou não é válida. Agora, para os campos mais comuns o HTML 5 já faz a validação sem a necessidade de nenhum código javascript.

O Quadro 3.17 mostra como fica o código do Quadro 3.16 em HTML já fazendo toda a validação, ou seja, sem a necessidade de nenhuma linha de javascript.

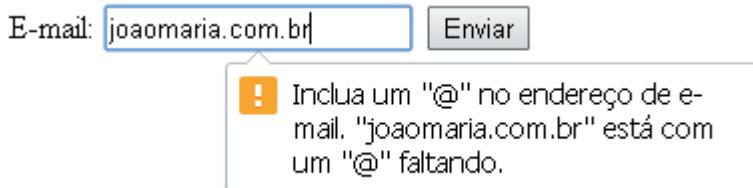
Quadro 3.17 Exemplo de código para entrada de dados HTML 5

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
    <title>Exemplo de código para entrada de dados HTML 5</title>
</head>
<body>
<form action="recebe_dados_email.php">
    E-mail:<input type="email" name="email">
    <input type="submit" value="Confirmar">
</form>
</body>
</html>
```

Ambos os códigos lhe pareceram iguais? Pois eles só não são idênticos por causa do valor do atributo “type”, que no Quadro 3.16 é type=“text” e no Quadro 3.17 é type=“email”.

Utilizando o tagemail e sem escrever código javascript algum, quando o usuário clica no botão “Confirmar”, o campo e-mail é validado e se o usuário não digitou um texto que possa representar um e-mail, o navegador não submete o formulário e mostra um “baloon” com uma mensagem indicando que o valor do campo é inválido. A Figura 3.1 mostra um exemplo dessa mensagem.

Figura 3.1 Exemplo de mensagem de validação de e-mail



Fonte: Do autor (2014).

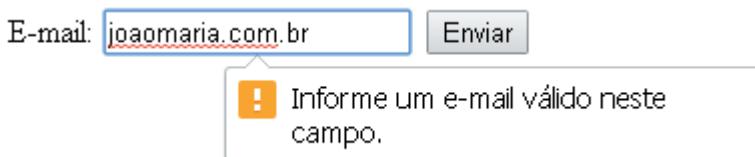
A mensagem mostrada varia de um navegador para outro, de um sistema operacional para outro e até mesmo de uma versão de navegador para outro.

Essa mensagem pode ser personalizada utilizando-se o evento oninvalid e a função javascriptsetCustomValidity, conforme mostra o exemplo do Quadro 3.18.

Quadro 3.18 Exemplo de código com mensagem personalizada para validação de e-mail

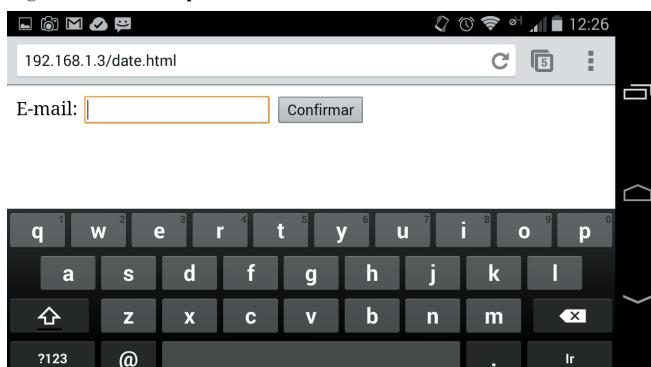
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
    <title>Mensagem personalizada para validação de e-mail</title>
</head>
<body>
<form action="recebe_dados_email.php">
    E-mail:<input type="email" name="email" oninvalid="setCustomValidity('Informe um
    e-mail válido neste campo.')">
    <input type="submit" value="Confirmar">
</form>
</body>
</html>
```

Dessa forma, se o usuário digitar um e-mail inválido será mostrada uma mensagem como a da Figura 3.2.

Figura 3.2 Exemplo de mensagem de validação de e-mail personalizada

Fonte: Do autor (2014).

Outra vantagem dos novos tipos de campos de entrada é que, em dispositivos móveis, o teclado para entrada de dados se modifica para deixar mais acessível a tecla @ e a tecla ponto, como mostra o exemplo da Figura 3.3.

Figura 3.3 Exemplo de entrada de e-mail no Chrome com Android

Fonte: Do autor (2014).

Dependendo do dispositivo, pode haver variações, por exemplo, o iPhone com Safari disponibiliza uma tecla “.com”.

Outro tipo de campo de entrada de dados, que faz a validação por conta do navegador, é o tipo url que exige que seja digitada uma url válida.

2.2 Required

Se você testar o código de exemplo do Quadro 3.18 ou 3.20, perceberá que se não preencher o campo ele vai deixar você confirmar e vai submeter o formulário com o valor vazio para o campo e-mail. Mas se você adicionar o atributo “required” ao elemento “input” ele vai passar a exigir que o campo seja sempre preenchido para submeter o formulário. O Quadro 3.19 mostra um exemplo desse código, assim como a imagem com a mensagem de validação.

Quadro 3.19 Exemplo de código com campo de entrada com preenchimento obrigatório e exemplo de mensagem

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Preenchimento obrigatório </title>
</head>
<body>
<formaction="recebe_dados_email.php">
    E-mail:<input type="email" name="email" required oninvalid="setCustomValidity('
        informe um e-mail válido neste campo.')">
    <inputtype="submit" value="Confirmar">
</form>
</body>
</html>
```

E-mail: | Confirmar

! Informe um e-mail válido neste campo.

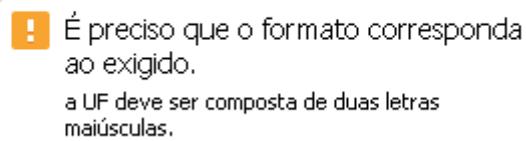
2.3 Pattern

O HTML 5 ainda permite que você crie uma validação personalizada através de expressões regulares. O Quadro 3.20 mostra um exemplo de validação utilizando expressão regular onde são aceitos apenas dois caracteres maiúsculos de A a Z.

Quadro 3.20 Exemplo de código com validação utilizando expressão regular e sua mensagem de validação

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
<title>validação utilizando expressão regular e sua mensagem de validação.</title>
</head>
<body>
<formaction="demo_form.asp">
UF:<input type="text" name="uf" pattern="[A-Za-z]{2}" title="a UF deve ser composta de
duas letras maiúsculas.">
<inputtype="submit" value="Confirma">
</form>
</body>
</html>
```

UF: Confirma



As expressões regulares podem ser utilizadas nos input types: text, search, url, tel, email, password.

No momento que este livro foi escrito o Safari ainda não tinha suporte para esse recurso, assim como versões do Internet Explorer anteriores a 10.

2.4 Novalidate

Se você desejar que um formulário com campos que fazem validação não sejam validados pelo navegador, basta adicionar o atributo novalidate ao tag form, como mostra o exemplo do Quadro 3.21.

Quadro 3.21 Exemplo de código que submete o formulário sem validar

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
<title>Exemplo de código que submete o formulário sem validar</title>
</head>
<body>
<formaction="recebe_dados_email.php" novalidate>
```

```
E-mail:<input type="email" name="email">
<input type="submit" value="Confirmar">
</form>
</body>
</html>
```

2.5 Entradas de dados para números, datas e hora

No HTML 4 tínhamos apenas entradas de dados textuais; se você precisasse que o usuário entrasse com uma data ou um número, para facilitar a entrada de dados e validar o conteúdo entrado, era necessária muita programação. Agora temos tipos de campos específicos para alguns tipos de informações mais utilizadas na web.

Um deles é o tipo **date**. Ele permite a digitação da data pelo teclado e também mostra um botão que, ao ser clicado, abre um calendário onde o usuário pode navegar e escolher uma data. Em dispositivos móveis, como celulares e tablets, é aberto o calendário específico do dispositivo, facilitando a navegação e a escolha da data. O Quadro 3.22 mostra um exemplo de código que usa o tipo date. A Figura 3.4 mostra como ele é visualizado no Chrome com Windows e a Figura 3.5 mostra como ele é visualizado em um celular Android.

Quadro 3.22 Exemplo de código de um formulário com campo de entrada de data

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de código de um formulário com campo de entrada de data</title>
</head>
<body>
<form action="teste.php">
Data de nascimento:<input type="date" name="dnasc">
<input type="submit">
</form>
</body>
</html>
```

Figura 3.4 Exemplo de funcionamento do código do Quadro 3.22 no Chrome com Windows

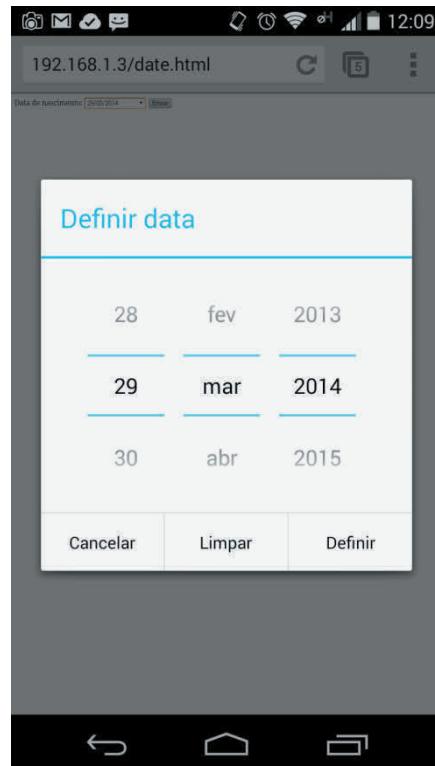
Data de nascimento: dd/mm/aaaa Enviar

março de 2014 ▾

dom	seg	ter	qua	qui	sex	sáb
23	24	25	26	27	28	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Fonte: Do autor (2014).

Figura 3.5 Exemplo de funcionamento do código do Quadro 3.22 no Chrome com Android



Fonte: Do autor (2014).

O tipo “date” atualmente não é suportado pelos navegadores Internet Explorer e Firefox.

Os outros tipos que têm a mesma característica do tipo “date” são:

- └ **datetime**: Permite selecionar uma data e uma hora e o fuso horário a que se refere, mas só funciona nos navegadores Safari e Opera.
- └ **datetime-local**: Permite selecionar uma data e hora sem a informação do fuso horário, mas não é suportado pelos navegadores Internet Explorer e Firefox.
- └ **month**: Permite selecionar um mês e um ano. Não é suportado pelos navegadores Internet Explorer e Firefox.
- └ **time**: Permite selecionar um horário com horas e minutos, mas sem fuso horário. Não é suportado por Firefox e Internet Explorer.
- └ **week**: Permite selecionar a semana e o ano. Não é suportado pelos navegadores Internet Explorer e Firefox.
- └ **number**: Permite informar apenas números; se o usuário digitar algo que não seja um número válido, ao tentar submeter o formulário, o navegador não deixa e mostra uma mensagem indicando que o campo deve receber um número. Ele também mostra uma seta para cima e uma para baixo para o usuário somar ou diminuir um número inteiro através de um clique ou toque. Nos dispositivos móveis, quando esse campo recebe foco, é habilitado o teclado numérico ao invés do normal. Esse tipo de campo ainda não funciona corretamente com números com separadores de milhar ou decimais e não é suportado pelo Firefox.

Todos esses tipos de campos, que representam uma quantidade contável, de dias, horas, meses, semanas ou números, podem receber os atributos **min** e **Max** para definir uma faixa válida de valores para o campo. E também podem receber o atributo **step** para definir intervalos regulares válidos para o campo.

O Quadro 3.23 mostra um exemplo onde um campo do tipo numérico pode receber números de -10 a 10 com intervalos de 3, ou seja, ele vai aceitar apenas os números -10, -7, -4, -1, 2, 5, e 8. A regra é que ele vai aceitar o menor valor e a partir dele, aceita os demais números em uma progressão aritmética de 3 em 3.

Quadro 3.23 Exemplo de código com tipo “number” e atributos “min”, “max” e “step”

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de código com tipo "number" e atributos "min", "max" e "step"</title>
</head>
<body>
<inputtype="number"name="pontos"step="3"min="-10"max="10"value="-10">
</body>
</html>
```

O atributo value também pode ser utilizado para definir o valor padrão para o campo. Outro tipo de controle que representa uma quantidade contável é o **range**. Ele mostra uma barra com um controle deslizante (conhecido como “slider”) que permite selecionar uma quantidade de forma visual, clicando e arrastando. É suportado por todos os navegadores.

O Quadro 3.24 mostra um exemplo de código que gera um controle deslizante, que varia de -10 a 8 com intervalos de 3. A Figura 3.6 exibe como esse controle é mostrado no navegador Chrome com Windows.

Quadro 3.24 Exemplo de código de formulário com controle deslizante

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Título do documento</title>
</head>
<body>
<formaction="demo_form.asp">
<inputtype="range"name="pontos"step="3"min="-10"max="10"value="-10">
<inputtype="submit">
</form>
</body>
</html>
```

Figura 3.6 Exemplo de como o código do Quadro 3.28 é mostrado no Chrome com Windows.



Fonte: Do autor (2014).



Para saber mais

Quando, neste livro, dizemos que um recurso não é suportado por determinado navegador, estamos falando sobre a época em que o livro foi escrito; entretanto, a evolução da Internet é muito rápida, então, é importante que vocês acompanhem essa evolução para saber quando um novo recurso for aceito pelos principais navegadores. Um site muito útil para saber quais browsers suportam um recurso de HTML, CSS e Javascript é o <<http://caniuse.com/>>. Por exemplo, se você digitar na sua busca o tag “Input”, ele listará quais os navegadores suportam o tag input e também cada tipo de input.

2.6 Outros tipos novos de entrada de dados

Outros tipos de campos para entrada de dados, que foram incluídos na especificação do HTML 5, são:

- └ **color**: Mostra ao usuário um seletor de cores onde ele pode escolher uma cor para ser submetida pelo formulário.
- └ **search**: mostra um campo de texto para entrada de uma string de busca.
- └ **tel**: mostra um campo de texto para o usuário entrar com um número de telefone.



Para saber mais

Para saber mais sobre todos os tipos de campos de entrada de dados acesse <http://www.w3schools.com/tags/att_input_type.asp>.

2.7 Listas e autocompletar

Foi criado um novo recurso para campos de entrada de texto com a finalidade de autocompletar um texto ou de buscar um texto predefinido em uma lista.

Para utilizar esse recurso, deve-se usar o atributo “list” no campo “input” que recebe a entrada de texto, onde o valor desse atributo deve ser o id de um tag “datalist” que contém uma lista de tags “option” com os textos a serem buscados representados como valor do campo “value”, conforme mostra o exemplo do Quadro 3.25.

Quadro 3.25 Exemplo de uso do novo atributo list e tagdatalist

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso do novo atributo list e tagdatalist</title>
</head>
<body>
    <formaction="teste.php"method="get">
        <inputlist="browsers" name="browser">
        <datalistid="browsers">
            <optionvalue="Internet Explorer">
            <optionvalue="Firefox">
            <optionvalue="Chrome">
            <optionvalue="Opera">
            <optionvalue="Safari">
        </datalist>
        <inputtype="submit">
    </form>
</body>
</html>
```

A Figura 3.7 mostra o exemplo do Quadro 3.25 em ação no navegador Chrome com Windows, na primeira imagem quando o usuário clica na seta para baixo e na segunda imagem, quando ele digita um texto que coincide com um item da lista.

Figura 3.7 Exemplo do código do Quadro 3.25 em ação

Fonte: Do autor (2014).

Esse recurso não impede que o usuário digite um texto qualquer diferente dos textos da lista, ou seja, ele apenas auxilia na digitação do conteúdo, mas não serve para sua validação. Esse novo recurso não está disponível no navegador Safari.

O HTML 5 também incluiu o atributo “autocomplete” nos campos “input”. Este não deve ser confundido com o recurso list e datalist visto anteriormente. O “autocomplete” é um recurso do próprio navegador que pode estar ativo ou inativo por padrão, de acordo com as configurações do navegador. De forma que um campo que não possui esse atributo pode estar “autocompletando” por conta do navegador. Os textos que o navegador lista como opção para autocompletar são os textos que o usuário já digitou alguma vez em um campo “input”, com o mesmo valor na propriedade “name”.

Os valores válidos para o atributo autocomplete são:

- └ **on**: Informa ao navegador que ele pode autocompletar se ele estiver com esse recurso ativo.
- └ **off**: mesmo que o navegador estiver com esse recurso ativo, nesse campo não é para autocompletar.

Em alguns casos, pode ser interessante dizer ao navegador explicitamente que naquele campo você quer que ele tente autocompletar ou o contrário, naquele campo você não quer que o navegador dê sugestões de texto. Um exemplo disso é o exemplo do Quadro 3.25, afinal, depois que você digitar alguns textos e submeter o formulário, quando você clicar no campo para escrever um texto, serão listados os itens do datalist mais os itens que você tinha submetido anteriormente naquele formulário, como mostra o exemplo da Figura 3.8.

Figura 3.8 Exemplo de autocomplete com datalist

The screenshot shows a dropdown menu from a browser. At the top right of the menu is a button labeled "Enviar". The menu contains the following items:

- Internet Explorer
- Firefox
- Chrome
- Opera
- Safari

Below this list, there is a single item: "asd". At the bottom of the list, the items "Internet Explorer" and "Opera" appear again, possibly indicating a search result or a different list source.

Fonte: Do autor (2014).

2.8 Autofocus

Quando você precisar que ao carregar a página um determinado campo de entrada de dados esteja com foco, basta incluir o atributo “autofocus” nele. E esse novo recurso já é suportado por todos os navegadores (pelo menos em sua última versão).

Esse atributo tem valor booliano, então as três formas mostradas no Quadro 3.26 são válidas, onde o valor false e a omissão do atributo autofocus têm o mesmo efeito.

Quadro 3.26 Três possíveis formas de usar o atributo autofocus

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Três possíveis formas de usar o atributo autofocus</title>
</head>
<body>
Forma 1, apenas o atributo:<input type="text" autofocus>
Forma 2, com valor true:<input type="text" autofocus="true">
Forma 2, com valor false: <input type="text" autofocus="false">
</body>
</html>
```

2.9 Placeholder

Um “placeholder” é um texto que aparece dentro do campo “input”, enquanto o campo ainda estiver vazio, e pode ser utilizado para descrever o conteúdo esperado pelo campo ou dar exemplos de como se espera que o campo seja preenchido.

Esse texto some automaticamente assim que o usuário entra com algum conteúdo no campo.

O Quadro 3.27 mostra um exemplo de código que se utiliza de placeholders e a Figura 3.9 mostra como esse código é executado em um navegador Chrome com Windows.

Quadro 3.27 Exemplo de uso de placeholder

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso de placeholder</title>
</head>
<body>
<formaction="demo_form.asp">
<inputtype="text"name="nomeplaceholder="Nome"><br>
<inputtype="text"name="sobrenomeplaceholder="Sobrenome"><br>
<inputtype="submit"value="Confirma">
</form>
</body>
</html>
```

Figura 3.9 Exemplo do código do Quadro 3.27 executando em um navegador Chrome com Windows

The screenshot shows a simple web form. It consists of two input fields stacked vertically. The top field is labeled 'Nome' and the bottom field is labeled 'Sobrenome'. Below these fields is a single button labeled 'Confirma'.

Fonte: Do autor (2014).



Questões para reflexão

Qual a diferença na produtividade da programação HTML 4 e HTML 5? Pense no diferencial que um programador HTML 5 pode ter no mercado de trabalho.



Atividades de aprendizagem

1. Crie um documento HTML com um formulário para cadastrar os seguintes dados de um aluno: Nome Completo (obrigatório), e-mail, site, data de nascimento, telefone e cor favorita. Use os recursos de HTML 5 para cada um dos campos.
2. Crie um documento HTML com um formulário para login e senha utilizando placeholders em vez de labels. O campo login deve receber o foco automaticamente ao carregar a página. Use os recursos de HTML 5 para cada um dos campos.

Seção 3 Semântica

Semântica é o estudo do significado. No HTML 4 existiam muitas tags que visavam alterar a aparência do documento, função que é mais adequadamente realizada pelo CSS. Então quando o código HTML continha muitos tags para alterar o visual e poucos que davam um significado para o código, o resultado era um código extenso, difícil de entender e desorganizado. No HTML 5 foram criados novos tags, que dividem o conteúdo em seções de acordo com o seu significado, ou seja, sua semântica. Além disso, os tags que eram utilizados para formatação foram revisados de forma que ficaram apenas os que podem ser utilizados para acrescentar semântica ao HTML.

Alguns exemplos de tags sem significado semântico são `<div>`, `` e `<object>`. Eles não dizem nada sobre o seu conteúdo.

Exemplos de tags com conteúdo semântico são `<form>`, `<table>` e ``, afinal eles definem claramente o seu conteúdo.

No HTML 4 era muito comum utilizar o elemento `<div>` para separar as diferentes partes do documento, diferenciando uma `<div>` de outra pela sua classe. Entretanto, cada um pode dar o nome das classes que desejar, não existe um padrão. O Quadro 3.28 mostra um exemplo de documento (bem organizado) que utiliza essa estrutura.

Quadro 3.28 Exemplo de página com estrutura utilizando tags sem semântica

```

<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
<title>Exemplo de página com div</title>
</head>
<body>
<divclass="menu">
<ul>
<li><a href="#">Home</a></li>
<li><a href="#sobre">Sobre</a></li>
<li><a href="#contato">Contato</a></li>
</ul>
</div><!-- /.menu -->
<divclass="cabecalho">
    <h1>Exemplo de página com div</h1>
    <p>Esta página demonstra como era estruturada a página quando não tinham
tags semânticas</p>
</div><!-- /.cabecalho -->
<divclass="corpo">
    <h2>Título do artigo </h2>
    Este artigo trata de como o conteúdo podia ser organizado antes e depois do
HTML 5.
    <h3>HTML 4</h3>

```

Antes usávamos div para tudo (...)

```
<h3>HTML 5</h3>
```

Agora usamos um tag para cada parte do site, deixando o código mais claro para o programador, padronizado e com uma semântica que o navegador e motores de busca podem identificar (...)

```
</div><!-- /.corpo -->
<divclass="painel-lateral">
  <divclass="quadro">
    <h3>Leia também:</h3>
    <ul>
      <li><a href="http://alistapart.com/article/previewofhtml5">A Preview of HTML 5</a></li>
      <li><a href="http://tableless.com.br/sections-elemento-aside/">Sections: elemento aside</a></li>
        <li><a href="http://html5doctor.com/the-article-element/">The articleelement</a></li>
      </ul>
    </div><!-- /.quadro -->
    <divclass="quadro">
      <h3>Questões para Reflexão</h3>
```

O elemento aside não serve apenas para painéis laterais, ele denota todo o conteúdo relacionado ao conteúdo principal e que agrupa algo a ele e pode estar posicionado em posições diferentes da lateral, depende apenas da criatividade do designer.

```
</div><!-- /.quadro -->
</div><!-- /.painel-lateral -->
<divclass="rodape">
  <hr>
  <p>&copy; Company 2014</p>
</div><!-- /.rodape -->
</body>
</html>
```

Os novos elementos semânticos que foram adicionados no HTML 5, com o objetivo de definir claramente cada parte de uma página, são:

- └ **section**: Usado para agrupar um conteúdo relacionado a um mesmo tema. Pode parecer como um elemento `<div>`, mas não é. O elemento `<div>` não tem significado semântico, ele pode representar qualquer coisa.
- └ **aside**: Utilizado para conteúdo tangencialmente relacionado. Apenas pelo fato de visualmente ele estar do lado direito ou esquerdo do conteúdo principal não é razão suficiente para se um aside. O conteúdo de um “aside” se for removido do documento, não deve reduzir o significado do assunto principal. Um exemplo é uma página com um texto que explica algo onde o “aside” conteria os comentários sobre essa explicação.
- └ **header**: Define um cabeçalho de um documento ou de uma seção. Ele deve ser utilizado para denotar um conteúdo introdutório. Um mesmo documento pode ter tantos “header” quantos forem necessários. O mais usual é que tenha um “header” principal para o documento e um “header” para cada seção.

- └ **nav:** Criado com a intenção de conter as informações para navegação do site. Não importa a forma visual dessa navegação, pode ser uma lista de links, um menu, desde que a intenção seja definir onde estão as informações necessárias para navegar por todo o site.
- └ **footer:** O seu nome parece como uma descrição de posicionamento, mas não é. Ele foi criado com a ideia de conter, sobre o conteúdo relacionado, informações como autor, data que foi escrito, direitos de cópia e links para conteúdo relacionado. Usualmente tem um “footer” para o documento inteiro e um para cada seção.
- └ **article:** É a seção principal do documento, quando existir uma. Não faz sentido que tenha mais do que um “article” em um mesmo documento.

O Quadro 3.29 mostra um exemplo de como escrever o código do Quadro 3.28 usando tags semânticas.

Quadro 3.29 Exemplo de página com estrutura utilizando tags com semântica

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
<title>Exemplo de página com div</title>
</head>
<body>
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#sobre">Sobre</a></li>
<li><a href="#contato">Contato</a></li>
</ul>
</nav>
<header>
    <h1>Exemplo de página com Header</h1>
    <p>Esta página demonstra o uso das tags semânticas</p>
</header>
<article>
    <header>
        <h2>Título do artigo </h2>
        Este artigo trata de como o conteúdo podia ser organizado antes e depois do
HTML 5.
    </header>
    <h3>HTML 4</h3>
    Antes usávamos div para tudo (...)

    <h3>HTML 5</h3>
```

Agora usamos um tag para cada parte do site, deixando o código mais claro para o programador, padronizado e com uma semântica que o navegador e motores de busca podem identificar (...)

```

    </article>
    <aside>
        <section>
            <h3>Leia também:<h3>
            <ul>
                <li><a href="http://alistapart.com/article/previewofhtml5">A Preview of
HTML 5</a></li>
                <li><a href="http://tableless.com.br/sections-elemento-aside/">Sections:
elemento aside</a></li>
                <li><a href="http://html5doctor.com/the-article-element/">The article
element</a></li>
            </ul>
        </section>
        <section>
            <h3>Questões para Reflexão</h3>
            O elemento aside não serve apenas para painéis laterais, ele denota todo o
conteúdo relacionado ao conteúdo principal e que agrupa algo a ele e pode estar posicionado
em posições diferentes da lateral, depende apenas da criatividade do designer.
        </section>
    </aside>
    <footer>
        <hr>
        <p>&copy; Company 2014</p>
    </footer>
</body>
</html>
```

3.1 Compatibilidade

Esses tags semânticos são suportados por todos os navegadores atuais, mas não funcionam com Internet Explorer 8 e anteriores. De forma que qualquer estilo CSS aplicando um dos tags semânticos não vai funcionar. Para resolver esse problema, SjoerdVisscher desenvolveu uma forma de contornar esse problema através de um código javascript que ele batizou de “HTML 5 Shiv”. Para utilizar essa solução, faça download do código javascript do site <<http://code.google.com/p/html5shiv/>>. Em seguida, inclua o código do Quadro 3.30 dentro do tag “head” do seu documento:

Quadro 3.30 Código necessário para utilizar o HTML 5 Shiv

```

<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
```

```
<!--[if lt IE 9]>
<script src="dist/html5shiv.js"></script>
<![endif]-->
<title>Código necessário para utilizar o HTML 5 Shiv</title>
</head>
<body>
    Conteúdo do documento...
</body>
</html>
```

Fonte: Neal (2014).



Questões para reflexão

Mas o HTML não é apenas um documento para ser visualizado? Se ele estiver organizado visualmente não basta? Por que o código fonte precisa ter semântica? As vantagens de incluir semântica ao código são inúmeras, você consegue imaginar quais?

Aqui relacionamos alguns motivos para o questionamento levantado nas "Questões para reflexão":

- └ Um motivo é deixar o código mais organizado e fácil de entender.
- └ Outro motivo é que um código HTML bem organizado facilita a aplicação do CSS.
- └ As ferramentas de busca conseguirão extrair do seu código HTML a essência da sua página, abstraindo outros códigos, como propagandas, menus etc. que, apesar de fazerem parte do site, nem sempre têm a ver com o conteúdo principal da página.
- └ Navegadores modernos podem incluir recursos de interação com o conteúdo da sua página conforme a semântica. Por exemplo, o Safari 5 implementa o recurso de leitura sem distrações, que, quando ativado, mostra apenas o "article" da página, omitindo todas as informações que podem distrair o usuário, como menus, rodapés e propagandas que ficam nas laterais.



Atividades de aprendizagem

Quais são os tags semânticos que servem para dividir semanticamente o conteúdo da página?

3.2 Figure

Outro elemento semântico adicionado ao HTML 5 é o <figure>. O seu propósito é especificar um conteúdo que constitui uma peça isolada do fluxo principal do documento, de forma que ela pode ser retirada do fluxo do documento sem alterar o significado do documento. Esse conteúdo pode ser por exemplo uma imagem, um gráfico, um exemplo de código ou uma ilustração.

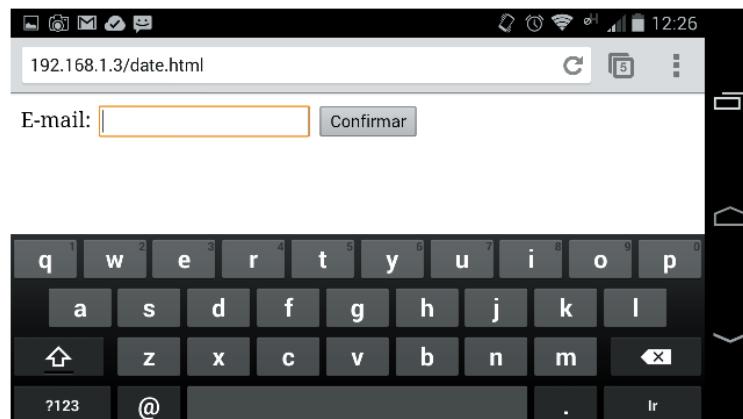
Opcionalmente, é possível definir uma legenda ou cabeçalho para esse conteúdo através do elemento <figcaption>.

O Quadro 3.31 mostra um exemplo de código para incluir uma ilustração usando figure e a Figura 3.10 mostra como esse código é renderizado em um navegador Chrome com Windows.

Quadro 3.31 Exemplo de uso do elemento <figure> com uma imagem

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso do elemento <figure> com uma imagem</title>
</head>
<body>
<p>Outra vantagem dos novos tipos de campos de entrada é que em dispositivos móveis, o teclado para entrada de dados se modifica para deixar mais acessível a tecla @ e a tecla ponto, como mostra o exemplo do quadro 3.22. </p>
<figure>
    <imgsrc="email.png">
    <figcaption>Quadro 22 - Exemplo de entrada de e-mail no Chrome com Android.
</figcaption>
</figure>
</body>
</html>
```

Figura 3.10 Exemplo de como o código do Quadro 3.31 é renderizado no navegador Chrome com Windows



Fonte: Do autor (2014).

O Quadro 3.32 mostra um exemplo de código para incluir uma “figure” que contém um trecho de código de exemplo. A Figura 3.11 mostra como esse código é executado no navegador Chrome com Windows.

Quadro 3.32 Exemplo de uso do elemento <figure> com um trecho de código

```
<!DOCTYPE html>
<html>
<head>
<metacharset="UTF-8">
    <title>Exemplo de uso do elemento <figure> com um trecho de código</title>
</head>
<body>
<p>O Quadro 3.13 mostra um exemplo simples de utilização do tagobject que adiciona um conteúdo Flash ao documento.</p>
<figure>
    <code>
        &lt;object width="400" height="400" data="helloworld.swf" type="application/x-shockwave-flash"&gt;&lt;/object&gt;
    </code>
    <figcaption>Quadro 3.13 - Exemplo simples de utilização do tagobject.
        Fonte: <a href="http://www.w3schools.com/tags/att_object_type.asp">
            http://www.w3schools.com/tags/att_object_type.asp
        </a>
    </figcaption>
</figure>
<p>A seguir vamos estudar mais a fundo cada um dos atributos do tagobject.</p>
</body>
</html>
```

Figura 3.11 Exemplo de como o código do Quadro 3.31 é renderizado no navegador Chrome com Windows

O Quadro 3.13 mostra um exemplo simples de utilização da tagobject que adiciona um conteúdo Flash ao documento.

```
<object width="400" height="400" data="helloworld.swf" type="application/x-shockwave-flash">
</object>
Quadro 3.13 – Exemplo simples de utilização do tagobject. Fonte : http://www.w3schools.com/tags/att\_object\_type.asp
```

A seguir vamos estudar mais a fundo cada um dos atributos do tagobject.

Fonte: Do autor (2014).

3.3 Outros novos elementos semânticos

O HTML 5 disponibiliza ainda mais elementos voltados para oferecer uma melhor estrutura ao documento. São eles:

- └ **main**: Define o conteúdo principal do documento.
- └ **details**: Define detalhes adicionais que você pode mostrar ou esconder.
- └ **summary**: Define um cabeçalho sempre visível para o elemento <details>.
- └ **mark**: Define um trecho de texto marcado ou realçado.
- └ **time**: Define um texto que representa uma data e/ou hora.



Para saber mais

Aprenda mais sobre esses elementos no site da W3S. Disponível em: <http://www.w3schools.com/html/html5_semantic_elements.asp>.



Atividades de aprendizagem

Crie um documento HTML utilizando os elementos <figure>, <figcaption>, <details>, <summary>, <mark> e <time>.

3.4 Javascript

O HTML 5 não trouxe novidades apenas para a linguagem de marcação HTML, mas também para a linguagem de programação javascript.

A seguir, vamos ver um resumo dos novos recursos Javascript.

- **Armazenamento:** Agora além dos cookies, é possível armazenar dados do lado do cliente com javascript de uma forma mais segura e eficiente. Este novo recurso se chama Web Storage e armazena dados na forma de pares de chave e valor. Estude mais sobre WebStorage em: <http://www.w3schools.com/html/HTML5_webstorage.asp> e <<http://tableless.com.br/html5/?chapter=21>>.
- **Geolocalização:** Permite solicitar ao navegador a localização geográfica do usuário na forma de latitude, longitude e altitude. Para saber mais sobre este recurso acesse: <http://www.w3schools.com/html/html5_geolocation.asp> e <<http://tableless.com.br/html5/?chapter=24>>.
- **Draganddrop:** Agora passaram a existir recursos nativos que combinam elementos HTML com javascript para implementar de uma forma muito fácil o recurso de draganddrop nos seus documentos HTML. Mais detalhes sobre este recurso pode ser encontrado em: <http://www.w3schools.com/html/html5_draganddrop.asp>.
- **Web Workers:** Permite rodar código javascript em segundo plano sem afetar o desempenho da página que está sendo carregada. Mais informações sobre este recurso podem ser estudadas em: <http://www.w3schools.com/html/html5_web_workers.asp>.
- **Server-SentEvents:** Permite que a página HTML no cliente fique escutando comandos do servidor. Este novo recurso revoluciona a comunicação via http, que antes sempre dependia de o cliente realizar uma solicitação para algo acontecer. Mais sobre este fantástico recurso pode ser estudado em: <http://www.w3schools.com/html/html5_serversentevents.asp> e <<http://tableless.com.br/html5/?chapter=15>>.

- └ **Acesso à webcam:** Permite o acesso à webcam do usuário através de javascript de uma forma muito simplificada. Mais sobre este recurso pode ser estudado em: <<http://tableless.com.br/html5/?chapter=12>>.
- └ **Gráficos com canvas:** Permite que sejam feitos desenhos e animações na página HTML com comandos Javascript simples. Descubra mais sobre este recurso em: <http://www.w3schools.com/html/html5_canvas.asp>, <http://animateyourhtml5.appspot.com/pres/index.html?utm_source=html5weekly&utm_medium=email#1> e <<http://tableless.com.br/html5/?chapter=14>>.



Para saber mais

Lista de tags HTML5, as novas, as que ficaram obsoletas e as que foram mantidas: <<http://www.w3schools.com/tags/default.asp>> e <<http://www.html5rocks.com/pt/>>

Certificação: <http://www.w3schools.com/html/html_quiz.asp>, <http://www.w3schools.com/html/html_exam.asp> e <http://www.w3schools.com/html/html5_exam.asp>.



Fique ligado!

Nesta unidade vimos como estruturar um documento HTML de forma semântica. Também aprendemos a adicionar conteúdo multimedia de forma fácil com HTML 5.

Conhecemos os novos e fantásticos recursos para criação de formulários na web que permitem a criação de formulários com funcionalidades profissionais com alta produtividade na programação.



Para concluir o estudo da unidade

Aqui nós vimos como o HTML 5 pode aumentar a sua produtividade como programador, facilitando e simplificando muito a resolução de tarefas que antes eram difíceis. Aplique o seu conhecimento nos sites que você e sua equipe produzem e observe o ganho de performance obtido.

Não deixe de se manter atualizado com relação ao HTML 5, afinal ele ainda está em processo de ser adotado integralmente por todos os navegadores, então um elemento que você deixou de usar em um primeiro momento porque ainda não tinha suporte em um determinado navegador pode vir a ter suporte em breve.

Uma boa forma de se manter atualizado é se inscrever em grupos de estudo e fóruns do assunto, como: <<http://www.html5rocks.com/pt/>>.



Atividades de aprendizagem da unidade

1. Utilizando os elementos com semântica, crie um documento HTML com as seguintes características: uma barra de navegação no topo, um cabeçalho logo abaixo com o título do site, em seguida, um espaço para as novidades do site, ao lado das novidades um espaço para links e no rodapé da página um espaço para informações de autoria do documento. Utilize os tags semânticos estudados.
2. Utilizando os novos tags HTML 5 para formulários, crie uma página com os campos:
 - └ nome completo (obrigatório)
 - └ telefone
 - └ e-mail (obrigatório)
 - └ login
 - └ senha
 - └ data de nascimento
 - └ cidade
 - └ UF: só pode aceitar um dos estados do Brasil e deve listar os estados. Usar datalist e pattern.
 - └ site pessoal
 - └ cor preferida
3. Crie uma página com o vídeo do YouTube: <<https://www.youtube.com/watch?v=07lwVVVoUk3Q>> incorporado dentro dela. Utilize as opções de compartilhamento de vídeo do próprio youtube.
4. Crie um documento HTML com um tagvideo que toca o vídeo automaticamente quando carrega a página e que não mostra os controles para o usuário controlar a exibição do mesmo.
5. Crie uma página que toca um som ao carregar, sem opção de o usuário parar o som, e que disponibilize outra música que pode ser tocada se o usuário quiser, com todos os controles disponíveis, utilize o tag <audio>.

Referências

- ADOBE SYSTEMS INCORPORATED. Ajuda do Flash Professional — Sintaxe da marca OBJECT. Disponível em: <<http://helpx.adobe.com/br/flash/kb/object-tag-syntax-flash-professional.html>>. Acesso em: 27 abr. 2014.
- CALZADILLA, Anthony. HTML 5 section, aside, header, nav, footer elements — not as obvious as they sound. Ago. 2010. Disponível em: <<http://www.anthonycalzadilla.com/2010/08/html5-section-aside-header-nav-footer-elements-not-as-obvious-as-they-sound/>>. Acesso em: 27 abr. 2014.
- CASTRO, Elizabeth. Bye Bye Embed. Jul. 2006. Disponível em: <<http://alistapart.com/article/byebyeembed>>. Acesso em: 27 abr. 2014.
- DEVERIA, Alexis. Can I use. Mar. 2014. Disponível em: <<http://caniuse.com>>. Acesso em: 27 abr. 2014.
- EIS, Diego. Sections: elemento aside — Parte 3. Out. 2010. Disponível em: <<http://tableless.com.br/sections-elemento-aside/>>. Acesso em: 27 abr. 2014.
- FREED, Ned. BAKER, Mark. HOEHRMANN, Bjoern. Media Types. Abr. 2014. Disponível em: <<http://www.iana.org/assignments/media-types/media-types.xhtml>>. Acesso em: 27 abr. 2014.
- GORNER, Martin. Animate your HTML 5. Disponível em: <http://animateyourhtml5.appspot.com/pres/index.html?utm_source=html5weekly&utm_medium=email#1a>. Acesso em: 27 abr. 2014.
- H5BP. HTML 5 BOILERPLATE. The web's most popular front-end template. Disponível em: <<http://html5boilerplate.com>>. Acesso em: 27 abr. 2014.
- HUNT, Lachlan. A Preview of HTML 5. Dez. 2007. Disponível em: <<http://alistapart.com/article/previewofhtml5>>. Acesso em: 27 abr. 2014.
- ISHIDA, Richard. Language subtag lookup app. Abr. 2014. Disponível em: <<http://people.w3.org/rishida/utils/subtags>>. Acesso em: 27 abr. 2014.
- LEADBETTER, Tom. The article element. Mai. 2010. Disponível em: <<http://html5doctor.com/the-article-element>>. Acesso em: 27 abr. 2014.
- MOZILLA DEVELOPER NETWORK. Track — HTML. Abr. 2014. Disponível em: <<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track>>. Acesso em: 27 abr. 2014.
- NEAL, Jonathan. HTML 5shiv — HTML 5 IE enabling script. Disponível em: <<http://code.google.com/p/html5shiv>>. Acesso em: 27 abr. 2014.
- NIELSEN, Jakob; LORANGER, Hoa. **Usabilidade na web**. Projetando Websites com qualidade. Rio de Janeiro: Campus, 2007.
- ROBINSON, Mike. Understanding aside. Jun. 2009. Disponível em: <<http://html5doctor.com/understanding-aside>>. Acesso em: 27 abr. 2014.
- ROUSSY, Christophe. HTML 5 best practices; section/header/aside/article tags. Nov. 2011. Disponível em: <<http://stackoverflow.com/questions/4781077/html5-best-practices-section-header-aside-article-tags>>. Acesso em: 27 abr. 2014.

TABLELESS. HTML 5 — Um guia de referência para os desenvolvedores web. Disponível em: <<http://tableless.com.br/html5/>>. Acesso em: 27 abr. 2014.

TEXT TRACKS COMMUNITY GROUP. WebVTT: The Web Video Text Tracks Format. Disponível em: <<http://dev.w3.org/html5/webvtt/>>. Acesso em: 27 abr. 2014.

W3C. Embedded content — HTML 5. fev. 2014. Disponível em: <<http://www.w3.org/TR/html5/embedded-content-0.html#the-track-element/>>. Acesso em: 27 abr. 2014.

W3SCHOOLS. HTML Tutorial. Disponível em: <<http://www.w3schools.com/html/>>. Acesso em: 27 abr. 2014.

CSS (Cascading Style Sheets)

Danilo Augusto Bambini Silva

Objetivos de aprendizagem: Ao finalizar esta unidade você terá um melhor entendimento do que é CSS, suas vantagens e principais regras. Assim como de sua importância na programação web.

↳ Seção 1: **CSS**

Nesta seção será apresentado o conceito do termo CSS, um pouco de sua história e as principais vantagens do seu uso na programação web.

↳ Seção 2: **Regras e sintaxe**

Nesta seção serão vistas as principais regras e sintaxe para CSS e os tipos de vinculação de folhas de estilo.

↳ Seção 3: **Tipos de vinculação de folhas de estilo**

Nesta seção veremos como vincular o CSS ao HTML.

↳ Seção 4: **Principais propriedades de CSS**

Nesta seção veremos alguns conceitos de referência importantes para o estudo do CSS e suas principais propriedades.

Introdução

Nesta unidade vamos ver o que é CSS, um pouco da sua história e importância para a programação web. Também veremos como usar e formular regras CSS, e um pouco das suas principais propriedades. O CSS3 acrescentou uma série de recursos incríveis para a programação web; infelizmente nem todos esses recursos estão disponíveis para os navegadores, até o momento da escrita deste livro, por isso procure se manter atualizado sobre as novidades, faça uso das fontes disponibilizadas, pesquise, e teste os recursos abordados ou não. Mantenha em mente que mesmo que você aprenda todos os recursos disponíveis hoje, sempre teremos novos recursos amanhã.

Seção 1 CSS

CSS ou Cascading Style Sheets e, para nós, folhas de estilo em cascata, é o mecanismo através do qual podemos formatar toda informação entregue pelo HTML. Essa formatação pode ou não ser visual. Ou seja, é por meio do CSS que iremos definir backgrounds, alinhamentos e muito mais. Por exemplo, com o CSS atual, nós também podemos manipular o áudio entregue ao visitante pelo sistema de leitura de tela, controlamos volume, profundidade, tipo da voz ou em qual das caixas de som a voz sairá.

Com as atualizações do CSS3 os designers ganharam uma importante ferramenta. Alguns dos principais módulos acrescentados ao CSS3 são:

- └ Seletores;
- └ Box Model;
- └ Fundos e Bordas;
- └ Os valores de imagem e conteúdo Substituído;
- └ Efeitos especiais em texto;
- └ 2D/3D Transformações;
- └ Animações;
- └ Layout com múltiplas colunas;
- └ Interface com Usuário.

Porém, fique atento com as versões de navegadores que aceitam as novidades do CSS3; os padrões web desde seus fundamentos sempre se preocuparam em nos fornecer ferramentas que garantam o acesso mesmo àqueles com necessidades especiais. Não foi viável abranger todos os recursos disponibilizados pelo CSS3, por isso faça uso das fontes e mantenha-se atualizado sobre as novidades na área.



Questões para reflexão

- └ Como posso aprender mais sobre CSS?
- └ Que navegadores suportam CSS? E o que são as características que eles suportam?
- └ Quais as ferramentas de autoria suportam CSS?
- └ Onde estão as especificações CSS?



Para saber mais

O navegador Argo era parte de um projeto para tornar a Internet acessível a estudiosos das Ciências Humanas. Ele apresentava *plug-ins* (que ele chamou de "applets") antes da Netscape adicioná-los.

1.1 História

A história do CSS começa em 1994. Havia uma grande demanda de produção de páginas web, porém, uma parte crucial de uma plataforma de publicação estava faltando, não havia maneira de estilizar documentos.

Tim Berners-Lee já havia definido em seu NeXT browser/editor, de maneira que ele pudesse determinar o estilo com uma folha de estilo simples. No entanto, ele não publicou a sintaxe para as folhas de estilo, considerando-a uma questão para que cada navegador decidisse a melhor forma de exibição das páginas para seus usuários. Mas em vez de folhas de estilo mais avançadas, os navegadores que se seguiram ofereceram a seus usuários cada vez menos opções para influenciar o estilo.

Em 1995, a Håkon publicou a primeira versão do Cascading Style Sheets HTML. Nos bastidores, Dave Raggett (o principal arquiteto do HTML 3.0) havia incentivado o lançamento do projeto para que este saísse antes da próxima conferência "Mosaic and the Web". A conferência foi realizada em 17-20 de outubro de 1995. Foi a segunda conferência, em que mais tarde se tornou a conferência "WWW Conference" em Chicago. Dave tinha percebido que o HTML nunca deveria se transformar em uma linguagem de descrição de página e que era necessário um mecanismo mais propositadamente construído para satisfazer as necessidades dos autores. Embora a primeira versão do documento fosse imatura, forneceu uma base útil para a discussão.

"Cascading Style Sheets" não foi a única linguagem de estilo proposta na época. Mas CSS tinha uma característica que a distinguiu de todas as outras: levou em conta que na Internet o estilo de um documento não pôde ser projetado pelo autor ou o

leitor por conta própria, mas que os seus desejos tinham que ser combinados, ou “em cascata”, de alguma forma, e, na verdade, não apenas a vontade do autor do leitor, mas também as capacidades do dispositivo de exibição do navegador.

No final de 1995, o W3C criou o Editorial Review Board HTML (HTML ERB) para ratificar especificações HTML futuras. CSS1 foi finalmente lançado em dezembro de 1996, como uma Recomendação W3C.

1.2 Vantagens de utilizar CSS

Os estilos são normalmente salvos em arquivos CSS externos. Folhas de estilo externas permitem alterar a aparência do layout de todas as páginas em um site, apenas editando um único arquivo. Você ainda pode definir diferentes apresentações do mesmo site de acordo com mecanismo, navegador ou público-alvo.



Atividades de aprendizagem

1. Quais motivos levaram à criação do CSS?
2. Quais as vantagens de se usar CSS?

Seção 2 Regras e sintaxe

Um documento CSS é uma série de regras; temos duas formas de regras: regras qualificadas e at-rules. Geralmente, regras qualificadas de estilo se aplicam às propriedades CSS para elementos, e as at-rules definem as regras especiais de processamento ou valores para o documento CSS.

2.1 A regra CSS e sua sintaxe

Uma regra começa com a definição de onde ela se aplica e varia de acordo com o contexto que a regra aparece. Nesse caso, pode ser um seletor que especifica quais os elementos que as declarações se aplicarão. Seguida por um bloco envolto em “{}” contendo uma sequência de declarações. Cada declaração tem um nome, seguido por dois pontos e o valor da declaração. Declarações são separados por ponto e vírgula.

Uma regra típica poderia ser algo como isto:

```
p > a {
    color: blue;
    text-decoration: underline;
}
```

Na regra apresentada anteriormente, “p > a” é o seletor, que seleciona quaisquer elementos <a>, que são filhos de um elemento <p>. E “color: blue;” é uma declaração especificando que, para os elementos que correspondem ao seletor, sua cor de propriedade deve ter o valor azul. Da mesma forma, a propriedade text-decoration define que o valor será sublinhado.

At-rules são todos diferentes, mas eles têm uma estrutura básica em comum. Eles começam com um “@” seguido do seu nome. Algumas at-rules são simples declarações, com o seu nome seguido de seus valores CSS para especificar o seu comportamento, terminando com um ponto e vírgula. Outros são blocos “{}”, semelhante a uma regra qualificada. Até mesmo o conteúdo desses blocos é específico para o dado na regra: às vezes ele contém uma sequência de declarações, como uma regra qualificada; outras vezes, pode conter blocos adicionais, ou pelo menos regras, ou outras estruturas completamente.

Aqui estão alguns exemplos de at-rules que ilustram a sintaxe variados.

```
@import "minha-styles.css";
```

O @import, em regra, é uma declaração simples. Depois de seu nome, é preciso uma função única corda ou url () para indicar a folha de estilo que deveria importar.

```
@page: left {
    margin-left: 4cm;
    margin-right: 3cm;
}
```

A @page consiste em um seletor de página opcional, seguido por um bloco de propriedades que se aplicam à página quando impressa. Dessa forma, é muito seme-

Intante a uma regra de estilo normal, exceto que suas propriedades não se aplicam a qualquer "elemento", mas sim a própria página.

```
@media print {
    body {font-size: 10pt}
}
```

A regra @media começa com um tipo de mídia e uma lista de consultas de mídia opcionais. O bloco contém as regras que só são aplicadas quando são cumpridas as condições do @media.

Os nomes das propriedades são sempre identificadores, que devem começar com uma letra ou um hífen seguido por uma letra, e depois pode conter letras, números, hífens ou sublinhados. Você pode incluir qualquer ponto de código em todos, mesmo aqueles que CSS usa em sua sintaxe, para escapar dele.

Comentários são sempre uma fonte importante de conhecimento em qualquer código. No CSS tudo que estiver entre /* ... */ é considerado comentário.



Questões para reflexão

Manter clareza e constância na definição de nomes, assim como manter uma indentação organizada do código é fundamental para garantir a legibilidade e a manutenção de todo código que você for escrever.



Para saber mais

Para saber sobre anúncios de novos projetos de especificações de CSS, consulte: <www.w3.org/Style/CSS/current-work.html>.

2.2 Relacionamento entre seletores

Para aplicarmos o CSS com eficiência é necessário um bom entendimento sobre os relacionamentos dos elementos do HTML.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title> Titulo do documento </title>
    </head>
    <body>
        <div>
            <h1> Tópico principal </h1>
            <ul>
                <li> <a> Link1 </a> </li>
                <li> <a> Link2 </a> </li>
```

```

<li> <a> Link3 </a> </li>
</ul>
</div>
<div>
    <h2> Tópico secundário </h2>
    <p id="p1"> Paragrafo 1 ... </p>
    <div>
        <p id="p4" > Paragrafo 4 ... </p>
    </div>
    <p id="p2"> Paragrafo 2 ... </p>
    <p> Paragrafo 3 ... </p>
</div>
</body>
</html>

```

Considerando-se o exemplo acima, vamos definir o relacionamento dos elementos citados.

- └ <body> é ancestral de todos os elementos, ou seja todos os elementos são descendentes de <body>,
- └ é ancestral de e de <a>,
- └ é descendente de ,
- └ <p id="p1"> é irmão do <h2>,
- └ <p id="p4"> não é irmão de <p id="p1">,
- └ não é filho de <div>, mas é seu descendente.

2.3 Tipos de seletores

Nos exemplos a seguir vamos considerar a letra E e a letra F como representantes de um elemento qualquer. Exemplo: div, p, a etc.

2.3.1 *

Qualquer elemento.

Exemplo:

```
* /* Todos os elementos terão background-color setado para amarelo */
{
    background-color:yellow;
}
```

ou

```
div * /* Todos os elementos que estiverem dentro da div terão o
background-color setado para amarelo */
```

```
{
    background-color:yellow;
}
```

2.3.2 E

Um elemento qualquer do tipo E.

Exemplo:

```
p /* todos os elementos p terão o background setado para amarelo */
{
    background-color:yellow;
}
```

2.3.3 .class

Qualquer elemento que tiver a classe class.

Exemplo:

```
p.class /* Todos os p que tiverem a classe .class, terão o background
setado para amarelo */
{
    background-color: yellow;
}
```

2.3.4 #id

O elemento setado com o id="id".

Exemplo:

```
#id /* O elemento com o id="id" terá o background-color setado para
amarelo */
{
    background-color: yellow;
}
```

2.3.5 E, F

Seleciona todos os E1, e todos os E2.

Exemplo:

```
div,p /* todos os elementos p e todos os elementos div terão o
background setado para amarelo */
{
    background-color:yellow;
}
```

2.3.6 E.warning

Um elemento E que contenha a classe “warning”.

Exemplo:

```
span.erro /* todo elemento span que contenha a classe erro */
{
    background-color:orange
}
```

2.3.7 :not(p)

Seleciona todos os elementos, exceto o p.

Exemplo:

```

...
<style>
p
{
color:#000000;
}
:not(p)
{
color:#ff0000;
}
</style>
</head>

<body>

<h1> Titulo principal </h1>
<p>Este é um paragrafo</p>
<p>Este é outro paragrafo</p>
<div>Este é um elemento div.</div>
<a href="#">Link !</a>

</body>
...

```

2.3.8 E F

Um elemento F que descenda do elemento E.

Exemplo:

```

...
<style>
div p
{
background-color:yellow;
}
</style>
</head>
<body>

<h1>Bem vindo a minha Homepage</h1>

<div>
<h2>Meu nome é Donald</h2>
<p>Eu vivo aqui.</p>
</div>

<p>Eu falo sozinho e gosto de programar !</p>

</body>
...

```

2.3.9 E > F

Um elemento F que seja filho do elemento E.

Exemplo:

```

...
<style>
div>p

```

```

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

<h1>Bem vindo a minha Homepage</h1>

<div>
<h2>Meu nome é Donald</h2>
<p>Eu vivo aqui.</p>
</div>

<p>Eu falo sozinho e gosto de programar !</p>

</body>
...

```

2.3.10 E + F

Um elemento F que seja imediatamente precedido por um E.

Exemplo:

```

...
<style>
div+p
{
background-color:yellow;
}
</style>
</head>
<body>

<h1>Bem vindo a minha Homepage</h1>

<div>
<h2>Meu nome é Donald</h2>
<p>Eu vivo aqui.</p>
</div>

<p>Eu falo sozinho e gosto de programar !</p>

</body>
...

```

2.3.11 E ~ F

Um elemento F precedido por um elemento E.

Exemplo:

```

...
<style>
div~p
{
background-color:yellow;
}
```

```

</style>
</head>
<body>

<h1>Bem vindo a minha Homepage</h1>

<div>
<h2>Meu nome é Donald</h2>
<p>Eu vivo aqui.</p>
</div>

<p>Eu falo sozinho e gosto de programar !</p>

</body>
...

```

2.3.12 E[foo]

Todos os elementos E com o atributo "foo".

Exemplo:

```

input[disable] /* todos os input definidos como disable terão o
background setado para amarelo */
{
    background-color:yellow;
}

```

2.3.13 E[foo="bar"]

Todos os elementos E, onde o atributo "foo" tem o valor exatamente igual a "bar".

Exemplo:

```

input[type=text] /* todos os input definidos com o type igual a text
terão o background setado para amarelo */
{
    background-color:yellow;
}

```

2.3.14 E[foo~=“bar”]

Todos os elementos E, onde o valor do atributo "foo" contém exatamente a palavra "bar" separada por um espaço em branco.

Exemplo:

```

...
<style>
    [title~=flower]
    {
        border:5px solid yellow;
    }
</style>
</head>
<body>
    
    

```

```

...

```

o único img a ser afetado será o:

```

```

2.3.15 E[foo^="bar"]

Todos os elementos E, onde o valor do atributo “foo” começa exatamente com a string “bar”.

2.3.16 E[foo\$="bar"]

Todos os elementos E, onde o valor do atributo “foo” termina exatamente com a string “bar”.

2.3.17 E[foo*="bar"]

Todos elementos E, onde o valor do atributo “foo” contém a substring “bar”.

2.3.18 E[foo |= "bar"]

Todos elementos E, onde o atributo “foo” tem um hífen separando uma lista de valores iniciando (da esquerda) com “bar”. O valor tem que ser uma palavra inteira, sozinha, como lang = “en”, ou seguida de um hífen (-), como lang = “en-us”.

2.4 Pseudoclasses

```
elemento:pseudoclasse {propriedade:valor;}
ou
elemento.classe:pseudo classe {propriedade:valor;}
```

O conceito de pseudoclass foi introduzido para permitir seleções baseadas em informações que estão fora da árvore de documento ou que não podem ser expressadas usando outros simples seletores. Uma pseudoclass sempre consiste em um “:” seguido pelo nome da pseudoclasse e, opcionalmente, por um valor entre dois parênteses.

Elas são permitidas em todas as sequências de seletores simples contidas em um seletor, depois do principal tipo de seletor ou seletor universal. Seus nomes são case insensitivos. Algumas pseudoclasses são mutualmente exclusivas, quando outras podem ser aplicadas simultaneamente para o mesmo elemento. Pseudoclasses podem ser dinâmicas, no sentido que um elemento pode adquirir ou perder uma pseudoclasse quando um usuário interage com o documento.

Exemplo:

```
a:link {color:#FF0000;} /* link não visitado */
a:visited {color:#00FF00;} /*link visitado */
a:hover {color:#FF00FF;} /* passar o mouse sobre o link (mouse over)
```

```
/*
a:active {color:#0000FF;} /* link selecionado */

<html>
<head>
<style>
p > i:first-child
{
color:blue;
}
</style>
</head>

<body>
<p>Eu sou um <i>strong</i> homem. Eu sou um <i>strong</i> homem.</p>
<p>Eu sou uma <i>strong</i> man. Eu sou uma <i>strong</i> mulher.</p>
</body>
</html>
```

2.4.1 E:root

Um elemento E, do documento root. Em HTML, o elemento raiz é sempre o elemento html.

Exemplo:

```
...
<style>
:root
{
background:#ff0000;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
...
```

2.4.2 E:nth-child(n)

Todos elementos E, em que este é o “n” filho do seu pai.

Exemplo:

```
...
<style>
p:nth-child(2)
{
background:#ff0000;
}
</style>
</head>
<body>

<h1>Aqui vai o titulo</h1>
<p>Primeiro paragrafo, aqui se aplica a regra. </p>
<p>Segundo paragrafo.</p>
<p>Terceiro paragrafo.</p>
```

```
<p>Quarto paragrafo.</p>
</body>
...
```

2.4.3 E:first-child

Um elemento E, onde E é o primeiro filho do seu pai.

Exemplo:

```
...
<style>
p:first-child
{
    background:#ff0000;
}
</style>
</head>
<body>
<p>Primeiro paragrafo, aqui se aplica a regra.</p>
<h1>Aqui vai o titulo</h1>
<p>Segundo paragrafo. </p>
<div>
    <p>Terceiro paragrafo, aqui se aplica a regra.</p>
    <p>Quarto paragrafo.</p>
    <p>Quinto paragrafo.</p>
</div>
</body>
...
```

2.4.4 E:only-child

Corresponde a cada elemento E que é o único filho do seu pai.

Exemplo:

```
...
<style>
p:only-child
{
    background:#ff0000;
}
</style>
</head>
<body>
<div> <p>Primeiro paragrafo, aqui se aplica a regra.</p> </div>
<h1>Aqui vai o titulo</h1>
<p>Segundo paragrafo. </p>
<div>
    <p>Terceiro paragrafo.</p>
    <p>Quarto paragrafo.</p>
    <p>Quinto paragrafo.</p>
</div>
<p> Último paragrafo.</p>
</body>
...
```

2.4.5 E:empty

Todos os elementos E que não têm filhos.

2.4.6 E:link E:visited

Todo elemento E que o seu link ainda não foi visitado (:link) ou foi visitado (:visited).

Exemplo:

```
...
<style>
a:link
{
    background-color:red;
}
a:visited
{
    background-color:orange;
}
</style>
</head>
<body>

<a href="http://www.w3schools.com">W3Schools Home</a><br>
<a href="http://www.w3schools.com/html/">W3Schools HTML Tutorial</
a><br>
<a href="http://www.w3schools.com/css"/>W3Schools CSS Tutorial</
a>
</body>
...
...
```

2.4.7 E:active E:hover E:focus

Seja o elemento E durante as ações citadas. Implementado a partir do CSS1 e CSS2.

Exemplo:

```
<style>
a:active {
    text-decoration: none;
}
a:hover {text-decoration: underline;
    color:#000;
    font-size:150%;
}

</style>
</head>
<body>

<a href="http://www.w3schools.com">W3Schools Home</a><br>
<a href="http://www.w3schools.com/html/">W3Schools HTML Tutorial</
a><br>
<a href="http://www.w3schools.com/css"/>W3Schools CSS Tutorial</
a>
</body>
...
...
```

2.4.8 E:target

Um elemento E que é o alvo de uma URI.

Exemplo:

```
...
<style>
  :target
  {
    border: 2px solid #D4D4D4;
    background-color: #e5eecc;
  }
</style>
</head>
<body>

<h1>Titulo</h1>

<p><a href="#item1">Vai para o item 1</a></p>
<p><a href="#item2">Vai para o item 2</a></p>

<p>Texto qualquer ...</p>

<p id="item1"><b>O conteúdo 1...</b></p>
<p id="item2"><b>O conteúdo 2...</b></p>

</body>
...
```

2.4.9 E:enabled E:disabled

Todos os elementos E que estão habilitados (enabled) ou desabilitados (disabled).

Exemplo:

```
...
<style>
  input[type="text"]:enabled
  {
    background:#ffff00;
  }

  input[type="text"]:disabled
  {
    background:#dddddd;
  }
</style>
</head>
<body>

<form action="">
  Nome: <input type="text" value="Mickey" /><br>
  Sobrenome: <input type="text" value="Mouse" /><br>
  País: <input type="text" disabled="disabled" value="Disneyland" /><br>
</form>

</body>
...
```

2.4.10 E:checked

Todos os elementos E que estão setados com o checked (para um radio-button checkbox).

Exemplo:

```
...
<style>
input:checked
{
background:#ff0000;
}
</style>
</head>
<body>

<form action="" >
<input type="radio" checked="checked" value="homem" name="genero" />
Homem <br/>
<input type="radio" value="mulher" name="genero" /> Mulher <br/>
<input type="checkbox" name="transporte" checked="checked"
value="bicicleta" /> Eu tenho uma bicicleta<br/>
<input type="checkbox" name="transporte" value="carro" /> Eu tenho
um carro
</form>

</body>
...
```

2.4.11 E:first-line

Seleciona a primeira linha do elemento E.

Exemplo:

```
...
<style>
p:first-line
{
background-color:yellow;
}
</style>
</head>

<body>
<h1>WWF's Missão</h1>
<p>Para parar a degradação do ambiente natural do planeta e construir
um futuro no qual os seres humanos vivam em harmonia com a natureza,
por; conservação da diversidade biológica do mundo, garantindo que o
uso dos recursos naturais renováveis seja sustentável, e promovendo
a redução da poluição e do desperdício consumo.</p>
</body>
...
```

2.4.12 E:first-letter

Seleciona a primeira letra do elemento E.

2.4.13 E:before

Insere um conteúdo antes do conteúdo de cada elemento E.

Exemplo:

```
...
<style>
p:before
{
    content:"Leia isso -";
}
</style>
</head>

<body>
<p>Meu nome é Donald</p>
<p>Eu vivo por aqui</p>
</body>
...
```

2.4.14 E:after

Insere um conteúdo depois do conteúdo de cada elemento E.

Exemplo:

```
...
<style>
p:after
{
    content:"Leia isso -";
}
</style>
</head>

<body>
<p>Meu nome é Donald</p>
<p>Eu vivo por aqui</p>
</body>
...
```



Atividades de aprendizagem

1. Escreva seletores que representem as seguintes regras:
 - └ todos os filhos do elemento estão envolvidos
 - └ os filhos do irmão do elemento
2. O que são pseudoclasses?

Seção 3**Tipos de vinculação de folhas de estilo**

Para que um código CSS, seja interpretado por uma página HTML ele deve estar vinculado a está de alguma forma. Existem três formas de vincular o CSS:

- └ Folha de estilo externa
- └ Folha de estilo interna
- └ Estilo inline

3.1 Folha de estilo externa

Uma folha de estilo externa é ideal quando o estilo é aplicado a muitas páginas. Com uma folha de estilo externa, você pode mudar a aparência de um site inteiro mudando um arquivo. Cada página tem um link para a folha de estilo usando o tag <link>. O tag <link> vai dentro da seção head do html:

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

Uma folha de estilo externa pode ser escrita em qualquer editor de texto. O arquivo não deve conter quaisquer tags HTML. Sua folha de estilo deve ser salva com uma extensão de “.css”. Um exemplo de um arquivo de folha de estilo é mostrado abaixo:

```
/* no arquivo .css você escreve somente as regras que você quer */
hr {
    color:sienna;
}
p {
    margin-left:20px;
}
body {
    background-image:url("images/background.gif");
}
```

3.2 Folha de estilo interna

Uma folha de estilo interna deve ser utilizada quando um único documento tem um estilo único. Você define estilos internos na seção head de uma página HTML, usando o tag <style>, como este:

```
<head>
<style>
hr {
    color: sienna;
}
```

```

p {
    margin-left: 20px;
}
body {
    background-image: url ("images / background.gif");
}
</style>
</head>

```

3.3 Estilo inline

Um estilo inline perde muitas das vantagens de folhas de estilo, misturando conteúdo com apresentação.

Para usar estilos inline você usa o atributo de estilo no tag relevante. O atributo de estilo pode conter qualquer propriedade CSS. O exemplo mostra como alterar a cor e a margem esquerda de um parágrafo:

```
<p style="color:sienna; margin-left:20px;"> Este é um parágrafo. </ p>
```



Para saber mais

Quase todos os navegadores hoje em dia suportam CSS e muitas outras aplicações também. Para escrever CSS, você não precisa de mais do que um editor de texto, mas há muitas ferramentas disponíveis que tornam ainda mais fácil. Claro, todo o software tem bugs, mesmo após diversas atualizações. E alguns programas estão mais adiantados em sua implementação dos mais recentes módulos CSS do que outros. Acesse este link para se inteirar de algumas ferramentas disponíveis: <<http://www.w3.org/Style/CSS/software>>.



Questões para reflexão

- └ Imagine a quantidade de trabalho necessária para se alterar um site onde cada elemento do HTML fosse personalizado dentro do próprio código HTML.
- └ Imagine a vantagem de reunir as personalizações dos elementos HTML em um arquivo separado.



Atividades de aprendizagem

1. Qual a melhor forma de incorporar CSS ao HTML e por quê ?
2. Qual o problema com o estilo inline ?

Seção 4

Principais propriedades de CSS

Antes de começar a ver as principais propriedades CSS, vamos olhar alguns conceitos importantes. A partir do CSS3 muitas propriedades passaram a aceitar os valores: initial e inherit.

- └ **Valor Initial:** seta a propriedade para o valor default.
- └ **Valor Inherit:** o elemento herda essa propriedade do seu elemento pai.

4.1 Unidades de medidas absolutas e relativas do CSS

Nós temos unidades de medida absolutas que são aquelas que não dependem de um valor de referência.

- └ in: Polegada.
- └ cm: Centímetro.
- └ mm: Milímetro.
- └ pt: Ponto. Unidade de medida tipográfica.
- └ pc: Pica. Outra unidade de medida tipográfica. 1pc é igual a 12pt.

Essas unidades são utilizadas quando se tem certeza do dispositivo em que a informação será visualizada. Mais comumente usada para impressões.

Unidades de medida relativas são mais comuns no meio Web, por se adaptarem aos diferentes dispositivos usados para a navegação, são elas:

- └ em: calculada em relação ao tamanho da fonte definida para o elemento em questão. No exemplo abaixo a margin-left será o equivalente a 20px:

```
...
<script>
p {
    font-size: 20px;
    margin-left: 1em;
}
</script>
...
```

- └ ex: 1ex é igual a altura da letra xis minúscula(x) da fonte definida para o elemento em questão.
- └ px: Pixel, é calculada em relação à resolução do dispositivo no qual o documento é apresentado.

4.2 Cores no CSS

Cores em CSS pode ser especificado através dos seguintes métodos:

- └ cores hexadecimais
- └ cores RGB
- └ cores RGBA

- └ cores HSL
- └ cores HSLA
- └ Nomes de cores predefinidas/Cross-browser

4.3 Cor hexadecimal

É definida por seis números hexadecimais precedidos pelo sinal “#”, exemplo: #000000. Formado por 3 pares que configuram a cor, o primeiro par representa a quantidade de vermelho, o segundo de verde, e o terceiro a quantidade de azul que entrou na composição da cor. Nos casos em que os números que formam os pares se repetem como: #FFFFFF, #6699FF etc. Podemos usar a forma abreviada colocando somente o primeiro número dos pares, exemplo: #000, #FFF, #69F.

4.4 Cor RGB

Define a cor por uma lista de três números colocados entre parênteses e separados por vírgula, precedidos pela palavra rgb, exemplo:

```
rgb(125,222,90),
rgb(30%,25%,70%).
```

4.5 Cor RGBA

Valores de cor RGBA são suportados no IE9, o Firefox 3, Chrome, Safari e no Opera 10. Valores de cor RGBA são uma extensão dos valores de cor RGB com um canal alfa — que especifica a opacidade do objeto.

Um valor de cor RGBA é especificado com: rgba (vermelho, verde, azul, alfa). O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (totalmente opaco).

Exemplo:

```
rgba(255,0,0,0.3); /* 0.3 de opacidade */
rgba(0,255,0,0.3); /* 0.3 de opacidade */
rgba(0,0,255,0.3); /* 0.3 de opacidade */
```

4.6 Cor HSL

Valores de cor HSL são suportados no IE9, Firefox, Chrome, Safari e no Opera 10.

Um valor de cor HSL é especificado com: HSL (matiz, saturação, luminosidade). Hue é um grau na roda de cores (de 0 a 360) — 0 (ou 360) é vermelho, é verde 120, 240 é azul. A saturação é um valor percentual; 0% significa um tom de cinza e 100% é a cor cheia. Leveza é também uma percentagem; 0% é preto, 100% é branco.

Exemplo:

```
hsl(120,100%,50%); /* verde*/
hsl(120,100%,75%); /* verde claro*/
hsl(120,100%,25%); /* verde escuro */
```

4.7 Cor HSLA

Valores de cor HSLA são suportados em IE9, Firefox 3, Chrome, Safari e Opera 10.

Valores de cor HSLA são uma extensão dos valores de cor HSL com um canal alfa — que especifica a opacidade do objeto. O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (totalmente opaco).

Exemplo:

```
hsia(120,100%,50%,0.3),
hsia(120,100%,75%,0.3),
hsia(120,100%,25%,0.3).
```

4.8 Cor por palavra chave

Há uma lista de nome de cores em inglês válida para definirmos cores no CSS. Por exemplo: *aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white* e *yellow*.

4.9 Nomes de cores predefinidas/cross-browser

Palavras chaves também são usadas para retirar cores retiradas do sistema operacional, por exemplo: *ActiveBorder*: cor da borda da janela ativa, *ActiveCaption*: cor do título da janela ativa. São elas: *AppWorkspace, Background, ButtonFace, Button-Highlight, ButtonText, CaptionText, GrayText, Highlight, HighlightText, InactiveBorder, InactiveCaption, InactiveCaptionText, InfoBackground, InfoText, Menu, MenuText, Scrollbar, ThreeDDarkShadow, ThreeDFace, ThreeDHighlight, ThreeDLightShadow, ThreeDSgadow, Window, WindowFrame, WindowText*.

4.10 Fontes

A propriedade font-family deve conter vários nomes de fontes como um sistema de "retorno", para garantir o máximo de compatibilidade entre navegadores/sistemas operacionais. Se o navegador não suporta a primeira fonte, ele tenta a próxima fonte.

Comece com a fonte que você quer e termine com uma família genérica, para deixar o navegador escolher uma fonte semelhante na família genérica, se não houver outros tipos de letra estão disponíveis (Quadro 4.1):

Quadro 4.1 Tipos de fontes

Serif Fonts

Font-family	Texto de exemplo
"Palatino Linotype", "Book Antiqua", Palatino, serif	Este é um título Este é um parágrafo
"Times New Roman", Times, serif	Este é um título Este é um parágrafo

Sans-Serif Fonts

Font-family	Texto de exemplo
Arial, Helvetica, sans-serif	Este é um título Este é um parágrafo
"Arial Black", Gadget, sans-serif	Este é um título Este é um parágrafo
Verdana, Geneva, sans-serif	Este é um título Este é um parágrafo

Monospace Fonts

Font-family	Texto de exemplo
"Courier New", Courier, monospace	Este é um título Este é um parágrafo
"Lucida Console", Monaco, monospace	Este é um título Este é um parágrafo



Para saber mais

Livros de interesse para saber mais:

SILVA, Maurício Samy. **CSS3**: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3 (Novatec, 2011).

SILVA, Mauricio Samy. **Construindo sites com CSS e (X)HTML** ("Building sites with CSS and (X)HTML").

4.11 Folhas de estilo aural

Folhas de estilo Aural usam uma combinação de síntese de voz e efeitos sonoros para fazer o usuário ouvir a informação, em vez de informações de leitura. Apresentações Aural podem ser usados:

- └ por pessoas cegas;
- └ para ajudar os usuários a aprender a ler;
- └ para ajudar os usuários que têm problemas de leitura;
- └ para entretenimento em casa;
- └ no carro;
- └ pelas comunidades com deficiência de impressão.

A apresentação aural converte o documento em texto simples e alimenta um leitor de tela (um programa que lê todos os caracteres na tela).



Para saber mais

Referência do Estilo Aural. Disponível em: <http://www.w3schools.com/cssref/css_ref_aural.asp>.



Questões para reflexão

Ao se desenvolver um CSS, deve-se ter em mente a forma que o público-alvo acessará o site e sua informação. O site será mais visitado por meio mobile ? As informações serão requisitadas para impressão? Nesse caso, é importante uma visualização diferente da informação? Usuários com necessidades especiais terão acesso facilmente as informações ?

4.12 Principais propriedades do CSS

4.12.1 Color

Seta a cor do texto.

Exemplo:

```
color: color|initial|inherit;
```

Color, é o valor da cor e pode ser definida das seguintes formas :

```
body
{
    color:red;
}
h1
{
    color:#00ff00;
}
p
{
    color:rgb(0,0,255);
}
```

Initial, configura esta propriedade com o valor padrão.

Inherit, herda essa propriedade do seu elemento pai.

4.12.2 Opacity

Seta o nível de opacidade do elemento. Exemplo:

```
...
<style>
div
{
background-color:red;
```

```

    opacity:0.5;
    filter:Alpha(opacity=50); /* IE8 and earlier */
}
</style>
</head>
<body>

<div>This element's opacity is 0.5! Note that both the text and the
background-color are affected by the opacity level!</div>

</body>
...

```

4.12.3 Background

Define as propriedades para o fundo do elemento HTML. Você pode definir todas as propriedades de background em uma só linha de CSS ou definir cada propriedade separadamente. Em uma só declaração, nós temos:

```
background: background-color, background-position, background-size,
background-repeat, background-origem, background-clipe, background-attachment, background-image|initial|inherit;;
```

Nota: se uma das propriedades na declaração é a propriedade *background-size*, você deve usar uma / (barra) para separá-lo da propriedade *background-position*, por exemplo:

```
background: url (smiley.gif) 10px 50px 20px/50px;
```

Irá resultar em uma imagem de fundo, posicionada 10 pixels a partir da esquerda, 20 pixels do topo, e o tamanho da imagem será de 50 pixels de largura e 50 pixels de altura.

Separadamente, podemos definir:

```
background-color: #FFF;
background-position: left top;
```

Seguem abaixo todas as propriedades disponíveis para o *background*:

└ *background-color*: especifica a cor que sera usada. Sintaxe:

```
background-color: color|transparent|initial|inherit;
```

- *color*: especifica a cor. Exemplo: #000;
- *transparent*: valor default. Especifica que a cor do fundo deve ser transparente.

└ *background-position*: especifica a posição da imagem de *background*. Sintaxe:

```
background-position: value;
```

- *left top*: canto superior esquerdo,
- *left center*: à esquerda centralizado verticalmente,
- *left bottom*: canto inferior esquerdo,
- *right top*: canto superior direito,
- *right center*: à direita centralizado verticalmente,

- right bottom: canto inferior direito,
- center top: centralizado horizontalmente, no topo,
- center center: centralizado,
- center bottom: centralizado horizontalmente, na parte baixa da página.
- x% y%: o primeiro valor define a posição horizontal e o segundo, a posição vertical. O canto do topo esquerdo é 0% 0%. O canto do topo direito é 100% 0%. Se você especificar somente um valor, o outro valor será 50%. O valor default é: 0% 0%.
- xpos ypos: o primeiro valor é a posição horizontal e o segundo valor é a posição vertical. O canto superior esquerdo é 0 0. Pode-se usar unidade em pixels (0px 0px) ou qualquer outra unidade CSS. Se você somente especificar um valor, o outro valor será de 50%. Você pode misturar % e posições.
- initial.
- inherit.

└ background-size: especifica o tamanho da imagem do background. Sintaxe:

```
background-size: auto|length|cover|contain|initial|inherit;
```

- auto: valor default, define a altura e a largura contida na imagem.
- length: define a largura e a altura da imagem de fundo. O primeiro valor define a largura, o segundo valor define a altura. Se apenas um valor é dado, o segundo é definido como "auto". Pode ser definida em unidade ou porcentagem.
- cover: dimensiona a imagem de fundo para ser tão grande quanto possível, de modo que a área de fundo é completamente coberta pela imagem de fundo. Algumas partes da imagem de fundo podem não ficar à vista dentro da área de posicionamento fundo.
- contain: dimensiona a imagem para o maior tamanho, de tal modo que tanto a sua largura e a sua altura pode encaixar no interior da área de conteúdo.

└ background-repeat: especifica como a repetição da imagem irá trabalhar.

Sintaxe:

```
background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;
```

- repeat: A imagem de fundo será repetida vertical e horizontalmente. Este é o padrão,
- repeat-x: A imagem sera repetida horizontalmente,
- repeat-y: A imagem sera repetida verticalmente,
- no-repeat: A imagem não sera repetida,
- initial: Seta essa propriedade para o valor default,
- inherit: Herda essa propriedade do seu elemento pai.

└ background-origin: define o posicionamento da imagem em relação ao elemento de origem. Disponível a partir do CSS3. Sintaxe:

```
background-origin: padding-box|border-box|content-box|initial|inherit;
```

- padding-box: este é o valor default da propriedade. A imagem de background é posicionada relativa ao padding do box,
- border-box: o background é posicionada relativa à borda do box,
- content-box: o background é posicionada relativa ao conteúdo do box.

↳ background-clip: define o a área de pintura do fundo. Disponível a partir do CSS3. Sintaxe:

```
background-clip: border-box|padding-box|content-box|initial|inherit;
```

- border-box: valor default da propriedade. O background começa a partir da borda do elemento
- padding-box: o background começa a partir do padding do elemento
- content-box: o background cobre o conteúdo do elemento

↳ background-attachment: define se a imagem de fundo permanecerá fixa ou rola junto com a tela. Sintaxe:

```
background-attachment: scroll|fixed|local|initial|inherit;
```

- scroll: valor default. A imagem de background permanece na tela mesmo que o usuário role o conteúdo.
- fixed: o fundo é fixo em relação ao navegador.
- local: ao rolar o conteúdo você rola junto a imagem de fundo.

↳ background-image: define uma ou mais imagens de fundo a ser usada pelo elemento. Sintaxe:

```
background-image: url|none|initial|inherit;
```

- url('URL'): especifica o “caminho” de uma ou mais imagens (separadas por vírgula).
- none: valor default. Define o background sem imagem de fundo.

↳ initial: Disponível a partir do CSS3.

↳ inherit.

Exemplo de definição de background:

```
...
<style>
body
{
    background: #00ff00 url('smiley.gif') no-repeat fixed center;
}
div
{
    border:1px solid black;
    padding:35px;
    background-image:url('smiley.gif');
    background-repeat:no-repeat;
    background-position:left;
}
</style>
...
```

4.12.4 Border

Vamos primeiro analisar as propriedades básicas de border implementadas desde o CSS1. Assim como a propriedade background, você tem duas formas de definir a propriedade border. A forma abreviada define todas as propriedades de border em uma declaração. As propriedades que podem ser definidas são (em ordem):

`border: border-width border-style border-color|initial|inherit;`

- └ border-width: define a largura da borda. O valor default é “medium”. Pode ser definida da seguinte forma:

`border-width: medium|thin|thick|length|initial|inherit;`

- medium: define uma largura média para a borda, é o valor default da propriedade.
- thin: define uma borda fina.
- thick: define uma borda espessa.
- length: permite que você defina uma esspessura para a borda, em uma unidade de medida reconhecida pelo CSS.

- └ border-style: define o estilo da borda, o valor default é none.

`border-style:none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset|initial|inherit;`

Com essa propriedade você também pode definir bordas diferentes para cada lado do elemento em uma só linha, “ border-style: estilos-topo estilos-direita estilos-inferior estilos-esquerda”. Exemplo: `border-style:dotted solid double dashed;`

- none: valor padrão. Define elemento sem borda.
- hidden: o mesmo que “none”, exceto na resolução de conflitos de bordas para os elementos de uma tabela.
- dotted: borda pontilhada.
- dashed: borda tracejada.
- solid: borda solida.
- double: borda dupla.
- groove: define uma borda 3D com ranhuras. O efeito depende do valor de border-color.
- ridge: define uma borda ondulada 3D. O efeito depende do valor de border-color.
- inset: define uma borda 3D onde o sombreado se concentra no canto superior do elemento.
- outset: define uma borda 3D onde o sombreado se concentra no canto inferior do elemento.

- └ border-color: define a cor a ser usada para borda, e assim como na propriedade border-style, você pode definir diferentes cores para cada borda do elemento em uma só linha, “border-color: cor-topo cor-direita cor-inferior cor-esquerda”.

Exemplo:

```
border-color:red green blue pink;
```

A sintaxe para essa propriedade é :

```
border-color: color|transparent|initial|inherit;
```

- color: pode ser definida com uma das formas de nomeação da cor aceita pelo CSS.

- transparent: especifica que a cor da borda é transparente.

Todas as propriedades acima se aplicam para:

- ↳ border-bottom: borda inferior.
- ↳ border-left: borda esquerda.
- ↳ border-right: borda direita.
- ↳ border-top: borda superior.

Exemplo:

```
border-top: border-width border-style border-color|initial|inherit;
p
{
    border-style:solid;
    border-top:thick double #ff0000;
}
```

A partir do CSS3 temos uma maior opção para configuração de bordas. Você pode criar bordas arredondadas, adicionar sombra para caixas e usar uma imagem como uma borda — sem usar um programa de desenho, como o Photoshop. Mas fique atento com as versões de navegadores que aceitam essas propriedades. Exemplo: O Internet Explorer só aceita border-radius a partir da versão 9.

- ↳ border-radius: é usada para criar cantos arredondados. Ela é definida da seguinte forma:

```
border-radius: 1-4 length|% / 1-4 length|%|initial|inherit;
```

Onde, length: define a forma dos cantos o valor padrão é 0. Você pode defini-lo em uma unidade de medida válida ou em porcentagem.

Exemplo:

```
div
{
    border:2px solid;
    border-radius:25px;
}
```

- ↳ box-shadow: esta propriedade cria sombras nos elementos. A sintaxe para essa propriedade é :

```
box-shadow: none|h-shadow v-shadow blur spread color
|inset|initial|inherit;
```

Onde:

- none: valor default. Não mostra sombra.
- h-shadow: a posição horizontal da sombra, valores negativos são permitidos.
- v-shadow: a posição vertical da sombra, valores negativos são permitidos.
- blur: a distância do borrão.
- spread: o tamanho da sombra.

- color: a cor da sombra.

Exemplo:

```
div
{
  box-shadow: 10px 10px 5px #888888;
}
```

- └ border-image: a propriedade border-image é um atalho para definir as propriedades border-image-source, border-image-slice, border-image-width, border-image-outset e border-image-repeat. Valores omitidos são definidos para seus valores padrões. Sintaxe:

```
border-image: border-image-source border-image-slice border-image-width border-image-outset border-image-repeat|initial|inherit;
```

Onde:

- border-image-source: define o caminho da imagem a ser usada na borda. Exemplo: border-image-source: url(border.png);
- border-image-slice: o deslocamento interior da border-image. Exemplo: border-image-slice: 30;
- border-image-width: define o tamanho da border-image. Exemplo: border-image-width: 10px;
- border-image-outset: define o valor pelo qual a área da border-image se estende para além do limite do elemento. Exemplo: border-image-outset: 5px;
- border-image-repeat: se a imagem será repetida e como será repetida. Sintaxe: border-image-repeat: stretch|repeat|round|space|initial|inherit; Onde:
 - stretch: valor default da propriedade. A imagem é aumentada para preencher a área.
 - repeat: a imagem é repetida para preencher a área.
 - round: a imagem é repetida para preencher a área. Porém se não preencher a área com um número inteiro de repetições, a imagem é redimensionada para que caiba.
 - space: a imagem é repetida para preencher a área. Porém, se não preencher a área com um número inteiro de repetições, o espaço extra é distribuída entre as imagens. Exemplos:

```
<style>
p
{
  border:5px solid red;
}
</style>

<style>
div
{
  border:15px solid transparent;
  width:250px;
```

```

        padding:10px 20px;
    }

#round
{
    -webkit-border-image:url(border.png) 30 30 round; /* Safari
5 */
    -o-border-image:url(border.png) 30 30 round; /* Opera */
    border-image:url(border.png) 30 30 round;
}

#stretch
{
    -webkit-border-image:url(border.png) 30 30 stretch; /* Safari
5 */
    -o-border-image:url(border.png) 30 30 stretch; /* Opera */
    border-image:url(border.png) 30 30 stretch;
}

```

4.13 Básico da propriedade box

Todos os elementos de HTML podem ser considerados como box. Em CSS, o termo "box model" é usado quando se fala em design e layout. O box model do CSS é essencialmente uma caixa que envolve elementos HTML e é composto por: margens, bordas, preenchimento e conteúdo real. O box model nos permite colocar uma borda em torno de elementos e elementos espaciais em relação a outros elementos. Para isso nós temos as propriedades apresentadas a seguir.

4.13.1 Margin

Corresponde a uma área ao redor da borda. A margem não tem uma cor de fundo.

Sintaxe: margin: length|auto|initial|inherit;

A margem pode ser definida de cinco formas diferentes:

└ **margin:10px 5px 15px 20px;**

- a margem superior é 10px
- a margem direita é 5px
- a margem inferior é 15px
- a margem esquerda é 20px

└ **margin:10px 5px 15px;**

- a margem superior é 10px
- a margem direita e a esquerda são 5px
- a margem inferior é 15px

└ **margin:10px 5px;**

- a margem superior e a inferior são 10px
- a margem direita e a esquerda são 5px

└ **margin:10px;**

- todas as margens são 10px
- └ Definindo previamente o lado da margem a ser configurado
 - margin-top
 - margin-left
 - margin-bottom
 - margin-right

Exemplo:

```
p.ex1{
    margin-top:5px;
    margin-left: 15px;
}
```

4.13.2 Border

A borda que gira em torno do preenchimento e do conteúdo. A borda é herdada da propriedade de cor do box. Já foi vista nesta unidade.

4.13.3 Padding

Corresponde a uma área em torno do conteúdo. O preenchimento é afetado pela cor de fundo do box. Assim como margin, temos quatro formas de defini-la:

- └ **padding:10px 5px 15px 20px;**
 - o padding superior é 10px
 - o padding à direita é 5px
 - o padding inferior é 15px
 - o padding à esquerda é 20px
- └ **padding:10px 5px 15px;**
 - o padding superior é 10px
 - o padding à direita e à esquerda são 5px
 - o padding inferior é 15px
- └ **padding:10px 5px;**
 - o padding superior e à inferior são 10px
 - o padding à direita e à esquerda são 5px
- └ **padding:10px;**
 - todos os padding são 10px
- └ Definindo previamente o lado do padding a ser configurado
 - padding-top
 - padding-left
 - padding-bottom
 - padding-right
- └ Content: o conteúdo do box, podendo ser imagens e textos.

A largura total de um elemento deve ser calculado da seguinte forma:

Largura total do elemento = width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

A altura total de um elemento deve ser calculado da seguinte forma:

Altura total do elemento = height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

4.13.4 Bottom

Para elementos posicionados absolutamente, a propriedade de fundo define o limite inferior de um elemento a uma unidade acima ou abaixo da borda inferior do elemento que a contém. Para os elementos posicionados relativamente, a propriedade de fundo define o limite inferior de um elemento a uma unidade acima ou abaixo de sua posição normal. Sua sintaxe é :

```
bottom: auto|length|initial|inherit;
```

Exemplo:

```
img
{
    position:absolute;
    bottom:5px;
}
```

4.13.5 Clear

Define quais os lados de um elemento onde outros elementos flutuantes não são permitidos. Sintaxe:

```
clear: none|left|right|both|initial|inherit;
```

Exemplo:

```
...
mg
{
    float:left;
}
p.clear
{
    clear:both;
}
...
```

4.13.6 Clip

A propriedade clipe permite que você especifique um retângulo para cortar um elemento absolutamente posicionado. O retângulo é especificado como quatro coordenadas, tudo a partir do canto superior esquerdo do elemento a ser cortada. Isso é útil quando se tem uma imagem maior do que a contém. Sua sintaxe é:

```
clip: auto|shape|initial|inherit;
```

onde:

`shape`: é a forma do recorte, definida da seguinte forma: `rect (top, right, bottom, left)`

Exemplo:

```
img
{
    position:absolute;
    clip:rect(0px,60px,200px,0px);
}
```

4.13.7 Display

Define como um elemento html será mostrado. Sintaxe:

```
display: value;
```

Os valores disponíveis são:

- `inline`: valor default. Mostra o elemento como um ítem inline. Exemplo: ``
- `block`: mostra o elemento como um elemento block. Exemplo: `<p>`
- `flex`: mostra o elemento como um contêiner block-level flexível. Novo no CSS3.
- `inline-block`: exibe um elemento como um recipiente bloco de nível incorporado. O interior desse bloco é formatado como caixa em nível de bloco, e do próprio elemento é formatado como uma caixa de nível de linha
- `inline-table`: o elemento é exibido como uma tabela de nível de linha
- `list-item`: Deixe o elemento se comportar como um elemento ``
- `inline-flex`: Exibe um elemento como um recipiente flex-nível interno. Novo em CSS3
- `run-in`: mostra um elemento como um bloco ou inline, dependendo do contexto
- `table`: mostra o elemento como um elemento `<table>`
- `table-caption`: mostra o elemento como um elemento `<caption>`
- `table-column-group`: mostra o elemento como um elemento `<colgroup>`
- `table-header-group`: mostra o elemento como um elemento `<thead>`
- `table-footer-group`: mostra o elemento como um elemento `<tfoot>`
- `table-row-group`: mostra o elemento como um elemento `<tbody>`
- `table-cell`: mostra o elemento como um elemento `<td>`
- `table-column`: mostra o elemento como um elemento `<col>`
- `none`: O elemento não será exibido (não tem efeito no layout)

4.13.8 Float

Define como e se um elemento irá “flutuar” no layout. Sintaxe:

```
float: none|left|right|initial|inherit;
```

Onde:

- └ none: o elemento não flutua,
- └ left: o elemento flutua para a esquerda,
- └ right: o elemento flutua para a direita.

Exemplo:

```
img
{
    float:right;
}
```

4.13.9 Height

Define a altura do elemento. Sintaxe:

```
height: auto|length|initial|inherit;
```

Exemplo:

```
p.ex
{
    height:100px;
    width:100px;
}
```

4.13.10 Width

Define a largura do elemento. Sintaxe:

```
width: auto|length|initial|inherit;
```

Exemplo:

```
p.ex
{
    height:100px;
    width:100px;
}
```

4.13.11 Left | right | top (esquerda | direita | topo)

Para elementos posicionados absolutamente (position: absolute ou position: fixed), a propriedade left|right define o quanto o elemento está distante da margem esquerda|direita|topo do elemento que o contém.

Para os elementos posicionados relativamente, a propriedade left|right|topo desloca o elemento a uma distância em unidade a esquerda|direita|topo da sua posição normal. Porém, é importante ressaltar que, se você estiver usando a propriedade "position: static", a propriedade left|right não tem efeito.

Sintaxe:

```
left: auto|length|initial|inherit;
right: auto|length|initial|inherit;
```

Exemplo

```
img
{
    position:absolute;
    left:5px;
}
```

4.13.12 Overflow

Define o que acontece se o conteúdo for maior que o elemento que o contém.

Sintaxe:

```
overflow: visible|hidden|scroll|auto|initial|inherit; /* (CSS2) */
```

A sintaxe acima se aplica a qualquer direção em que o conteúdo seja maior que o elemento que o contém. Abaixo temos a sintaxe overflow-x, que define a regra para quando o conteúdo horizontal for maior que o elemento, e overflow-y para quando o conteúdo for verticalmente maior que o elemento que o contém.

```
overflow-x: visible|hidden|scroll|auto|initial|inherit; /* (CSS3) */
overflow-y: visible|hidden|scroll|auto|initial|inherit; /* (CSS3) */
```

Onde:

- └ visible: o conteúdo não é recortado, e pode ser mostrado fora do “box” do elemento. Ou seja o conteúdo “vaza” para fora dos limites do elemento.
- └ hidden: o conteúdo que passar do limite do conteúdo não será mostrado. É como se o elemento recortasse o conteúdo para que este caiba.
- └ scroll: o conteúdo parece recortada, porém barras de rolagem serão incorporadas ao elemento, para que se tenha acesso a este.
- └ auto: deve causar um mecanismo de rolagem a ser fornecido para quando o conteúdo transbordar do elemento.

4.13.13 Position

A propriedade position define o método de posicionamento usado para o elemento. Sintaxe:

```
position: static|absolute|fixed|relative|initial|inherit;
```

onde:

- └ static: valor default. Mostra o elemento na mesma ordem em que este foi colocado no documento.
- └ absolute: o elemento é posicionado relativamente ao seu primeiro elemento (não estático) ancestral.
- └ fixed: o elemento é posicionado relativo a janela do browser.
- └ relative: o elemento é posicionado relativo a sua posição normal, então “left:20” adiciona 20 pixels para a esquerda na posição do elemento.

Exemplo:

```

h2
{
  position:absolute;
  left:100px;
  top:150px;
}

```

4.13.14 Visibility

Define se o elemento é ou não visível. Sintaxe:

```
visibility: visible|hidden|collapse|initial|inherit;
```

onde:

- └ visible: valor default. O elemento é visível.
- └ hidden: o elemento é invisível (mas ocupa lugar na página).
- └ collapse: Apenas para os elementos de tabela. Remove uma linha ou coluna, mas não afeta o layout da tabela. O espaço ocupado pela linha ou coluna estará disponível para outros tipos de conteúdo. Se o colapso é usado em outros elementos, ele processa como "hidden".

Exemplo:

```

h2
{
  visibility:hidden;
}

```

É importante dizer que todo elemento invisível ocupa espaço na página, para criar um elemento invisível que não ocupa lugar na página use a propriedade "display: none".

4.13.15 Vertical-align

Define o alinhamento vertical do elemento. Sintaxe:

```
vertical-align: baseline|length|sub|super|top|text-top|middle|bottom
|text-bottom|initial|inherit;
```

Onde:

- └ baseline: valor default. Alinha a linha do elemento baseado no alinhamento do seu elemento pai.
- └ length: sobe ou abaixa o elemento de acordo com o valor informado. Valores negativos são permitidos.
- └ sub: alinha o elemento como se fosse um subscrito.
- └ super: alinha o elemento como se fosse um sobrescrito.
- └ top: a parte superior do elemento estará alinhada com a parte superior do elemento mais alto na linha.
- └ text-top: o topo do elemento é alinhado com o topo da fonte do elemento pai.
- └ middle: o elemento é posicionado no meio do elemento pai.
- └ bottom: o elemento é posicionado na parte mais baixa do elemento pai.

- └ text-bottom: o fundo do elemento é alinhado com a parte inferior da fonte do elemento pai

Exemplo:

```
img
{
    vertical-align:text-top;
}
```

4.13.16 Z-index

Define a ordem de sobreposição do posicionamento de um elemento. Um elemento com um maior z-index estará na frente de um elemento com um z-index inferior. Sintaxe:

```
z-index: auto|number|initial|inherit;
```

Onde:

- └ auto: é o valor default. Determina que sua ordem é a mesma que a do seu elemento pai.
- └ number: é um valor inteiro que representa a ordem de sobreposição do elemento. Valores negativos são permitidos

Exemplo:

```
img
{
    position:absolute;
    left:0px;
    top:0px;
    z-index:-1;
}
```

4.14 Propriedades de layouts flexíveis

4.14.1 Align-content

Determina o alinhamento vertical entre linhas dentro de um contêiner flexível, quando os itens não estão usando todo o espaço disponível. Disponível a partir do CSS3. Sintaxe:

```
align-content: stretch|center|flex-start|flex-end|space-between|space-around|initial|inherit;
```

Utilize a propriedade justify-content para alinhar os itens no eixo principal (na horizontal). Além disso, deve haver várias linhas de artigos para essa propriedade para ter qualquer efeito, onde:

- └ stretch: valor default. Os itens são esticados para caber no contêiner.
- └ center: os itens são posicionados no centro do contêiner.
- └ flex-start: os itens são posicionados no início do contêiner.
- └ flex-end: os itens são posicionados no final do contêiner.
- └ space-between: os itens são posicionados com espaços iguais entre as linhas.

- ↳ space-around: os itens são posicionados com espaços antes entre e depois das linhas.

Exemplo:

```
div
{
    display: flex;
    flex-flow: row wrap;
    align-content:space-around;
}
```

4.14.2 Aling-itens

Determina o alinhamento por itens contidos em um contêiner flexível. Disponível a partir do CSS3. Sintaxe:

```
align-items: stretch|center|flex-start|flex-end|baseline|initial|
inherit;
```

onde:

- ↳ stretch: valor default, Os itens são esticados para caber no contêiner.
- ↳ center: os itens são posicionados no centro do contêiner.
- ↳ flex-start: os itens são posicionados no início do contêiner.
- ↳ flex-end: os itens são posicionados no final do contêiner.
- ↳ baseline: os itens são posicionados na linha de base do contêiner

Exemplo:

```
div
{
    display: flex;
    align-items:center;
}
```

4.14.3 Align-self

Define o alinhamento para os itens selecionados dentro de um contêiner flexível. Disponível a partir do CSS3. Sintaxe:

```
align-self: auto|stretch|center|flex-start|flex-end|baseline|initial
|inherit;
```

Vale ressaltar que a propriedade align-self substitui a propriedade align-items, em que:

- ↳ auto: valor default. O elemento herda alinhar-itens de propriedade do seu contêiner pai, ou "estica" se não tem nenhum contêiner pai.
- ↳ stretch: o elemento é posicionado para preencher todo o contêiner.
- ↳ center: o elemento é posicionado no centro.
- ↳ flex-start: o elemento é posicionado no início do contêiner.
- ↳ flex-end: o elemento é posicionado no final do contêiner.
- ↳ baseline: o elemento é posicionado na linha de base do contêiner.

Exemplo:

```
#myBlueDiv
{
    align-self:center;
}
```

4.14.4 Flex

Define o tamanho do item, relativo ao restante dos itens dentro do mesmo contêiner. Esta propriedade pode ser escrita em uma só linha ou declarando separadamente seus valores. Disponível a partir do CSS3. Sintaxe:

```
flex: flex-grow flex-shrink flex-basis|auto|initial|inherit;
```

Onde:

- └ flex-grow: um número que especifica quanto o item irá crescer em relação ao resto dos elementos flexíveis.
- └ flex-shrink: um número que especifica quanto o item irá encolher em relação ao resto dos elementos flexíveis
- └ flex-basis: o comprimento do elemento. Valores aceitos: "auto", "inherit" ou um número seguido de "%", "px", "em" ou qualquer outra unidade de comprimento
- └ auto: o mesmo que 1 1 auto.

Exemplo:

```
#main div
{
    flex:1;
}
div:nth-of-type(2)
{
    flex-basis: 80px;
}
```

4.14.5 Flex-wrap

Define se os filhos do elemento são forçados a uma única linha ou se os itens podem fluir em várias linhas.

4.14.6 Justify-content

Define o alinhamento horizontal entre os itens dentro de um contêiner flexível, quando os itens não usam todo o espaço disponível. Disponível a partir do CSS3. Sintaxe:

```
justify-content: flex-start|flex-end|center|space-between|space-around
|initial|inherit;
```

Onde:

- └ flex-start: valor default. Os itens são posicionados no início do contêiner.
- └ flex-end: os itens são posicionados no final do contêiner.
- └ center: os itens são posicionados no centro do contêiner.
- └ space-between: os itens são posicionados com espaços iguais entre as linhas.
- └ space-around: os itens são posicionados com espaços antes, entre e depois das linhas.

Exemplo:

```
div
{
    display: flex;
    justify-content: space-around;
}
```

4.14.7 Max-height

Define a altura máxima que o elemento pode ter. Sintaxe:

```
max-height: none|length|initial|inherit;
```

Exemplo:

```
p
{
    max-height:50px;
}
```

4.14.8 Min-height

Define a altura mínima para o elemento.

4.14.9 Order

Define a ordem do item, relativo ao restante dos itens no mesmo contêiner. Disponível a partir do CSS3. Sintaxe:

```
order: number|initial|inherit;
```

Exemplo:

```
div#myRedDIV {order:2;}
div#myBlueDIV {order:4;}
div#myGreenDIV {order:3;}
div#myPinkDIV {order:1;}
```

4.15 Propriedade para textos

4.15.1 Letter-spacing

Incrementa ou decremente o espaço entre os caracteres no texto. Sintaxe:

```
letter-spacing: normal|length|initial|inherit;
```

Exemplo:

```
h1 {letter-spacing:2px}
h2 {letter-spacing:-3px}
```

4.15.2 Line-height

Define a altura da linha. Sintaxe:

```
line-height: normal|number|length|initial|inherit;
```

Onde:

- └ normal: valor default. Tamanho normal.

- └ number: um número que sera multiplicado com o atual tamanho da fonte setado no line-height.

length:

Exemplo:

```
p.small {line-height:90%}
p.big {line-height:200%}
```

4.15.3 Text-align

Define o alinhamento horizontal do texto. Sintaxe:

```
text-align: left|right|center|justify|initial|inherit;
```

Exemplo:

```
h1 {text-align:center}
h2 {text-align:left}
h3 {text-align:right}
```

4.15.4 Text-indent

Define a identificação da primeira linha em um bloco de texto.

4.15.5 Text-transform

Controla a capitalização do texto. Sintaxe:

```
text-transform: none|capitalize|uppercase|lowercase|initial|inherit;
```

Exemplo:

```
p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}
```

4.15.6 White-space

Define como os espaços em branco dentro de um elemento é tratado. Sintaxe:

```
white-space: normal|nowrap|pre|pre-line|pre-wrap|initial|inherit;
```

Onde:

- └ normal: valor default. Sequências de espaços em branco sera fundido em um único espaço em branco. O texto quebra para a próxima linha quando necessário.
- └ nowrap: Sequências de espaços em branco sera fundido em um único espaço em branco. Texto nunca vai quebrar para a próxima linha. O texto continua na mesma linha até que um tag
 for encontrado.
- └ pre: espaços em branco são preservados pelo browser. O texto só irá envolver em quebras de linha, se for encontrado um tag <pre> em HTML.
- └ pre-line: sequências de espaços em branco serão fundidos em um único espaço em branco. O texto vai quebrar quando necessário.
- └ pre-wrap: espaços em branco são preservados pelo browser. Texto vai quebrar quando necessário, e com quebras de linha.

Exemplo:

```
p
{
  white-space:nowrap;
}
```

4.15.7 Word-break

Especifica as regras de quebra de linha para os scripts não CJK. Os scripts CJK são scripts chinês, japonês e coreano ("CJK"). Disponível a partir do CSS3. Sintaxe:

```
word-break: normal|break-all|keep-all|initial|inherit;
```

Onde:

- └ normal: valor default. Quebra as palavras de acordo com suas regras habituais.
- └ break-all: as linhas podem quebrar entre quaisquer duas palavras.
- └ keep-all: quebras são proibidas entre pares de letras.

Exemplo:

```
p.test {word-break:break-all;}
```

4.15.8 Word-spacing

Incrementa ou decremente o espaço entre as palavras em um texto. Sintaxe:

```
word-spacing: normal|length|initial|inherit;
```

Exemplo:

```
p
{
    word-spacing:30px;
}
```

4.15.9 Word-wrap

Permite que palavras longas, sejam quebradas para a próxima linha. Disponível a partir do CSS3. Sintaxe:

```
word-wrap: normal|break-word|initial|inherit;
```

Exemplo:

```
p.test {word-wrap:break-word;}
```

4.15.10 Text-decoration

A propriedade text-decoration especifica a decoração adicionada ao texto. Sintaxe:

```
text-decoration: none|underline|overline|line-through|initial|inherit;
```

Onde:

- └ none: valor default. Define um texto normal.
- └ underline: define uma linha abaixo do texto.
- └ overline: define uma linha acima do texto.
- └ line-through: define uma linha através do texto.

Exemplo:

```
h1 {text-decoration:overline}
h2 {text-decoration:line-through}
h3 {text-decoration:underline}
```

4.15.11 Text-decoration-color

Define a cor da decoração do texto. Disponível a partir do CSS3. Sintaxe:

```
text-decoration-color: color|initial|inherit;
```

4.15.12 Text-decoration-line

Define o tipo de linha da decoração do texto. Disponível a partir do CSS3. Sintaxe:
`text-decoration-line: none|underline|overline|line-through|initial
|inherit;`

Onde:

- └ none: valor default. Define um texto normal.
- └ underline: define uma linha abaixo do texto.
- └ overline: define uma linha acima do texto.
- └ line-through: define uma linha através do texto.

Exemplo:

```
p
{
    text-decoration-line: overline;
    -moz-text-decoration-line: overline; /* Code for Firefox */
}
```

4.15.13 Text-shadow

Adiciona sombra para o texto. Disponível a partir do CSS3. Sintaxe:
`text-shadow: h-shadow v-shadow blur color|none|initial|inherit;`

Onde:

- └ h-shadow: a posição horizontal da sombra, valores negativos são permitidos.
- └ v-shadow: a posição vertical da sombra, valores negativos são permitidos.
- └ blur: a distância do borrão.
- └ color: a cor da sombra.
- └ none: valor default. Não há sombra.

Exemplo

```
h1
{
    text-shadow: 2px 2px #ff0000;
}

h2
{
    text-shadow: none;
}
```

4.15.14 Direction

Define a direção de escrita. Sintaxe:

`direction: ltr|rtl|initial|inherit;`

Onde:

- └ ltr: valor default. Define a escrita da esquerda para a direita.
- └ rtl: define a escrita da direita para a esquerda

Exemplo:

```
div
{
    direction:rtl;
}
```

4.15.15 Font

Define todas as propriedades possíveis para fonte em uma só declaração. Sintaxe:

```
font: font-style font-variant font-weight font-size/line-height|
caption|icon|menu| message-box|small-caption|status-bar|initial|
inherit;
```

Assim como na propriedade background, também podemos definir suas propriedades separadamente; seguem definição e a sintaxe:

- └ font-style: define o estilo da fonte. Sintaxe: font-style: normal|italic|oblique|initial|inherit;
- └ font-variant: especifica se ou não um texto deve ser exibido em uma fonte small-caps. Sintaxe: font-variant: normal|small-caps|initial|inherit;
- └ font-weight: define a espessura da fonte. Sintaxe: font-weight: normal|bold|bolder|lighter|number|initial|inherit;
- └ font-size/line-height: define o tamanho da fonte e o tamanho da linha. Sintaxe: font-size: medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|length|initial|inherit; line-height: normal|number|length|initial|inherit;
- └ caption|icon|menu|message-box|small-caption|status-bar: em vez de especificar propriedades individuais, uma palavra-chave pode ser usada para representar uma fonte de um sistema específico.

Exemplo:

```
p.ex1
{
    font:15px arial,sans-serif;
}
p.ex2 /* exemplo com o font-size/line-height: */
{
    font:italic bold 12px/30px Georgia, serif;
}
p.ex3 /* exemplo de uma caracteristica definida separadamente */
{
    font-style:italic;
}
p .ex4 /* usa a mesma fonte que o status bar da janela esta usando*/
{
font: status-bar
}
```

4.15.16 @font-face

Com a regra @font-face, os web designers já não têm que usar uma das fontes "web-safe". Na nova regra @font-face é necessário primeiro definir um nome para

a fonte (Ex.: minhaFonte) e, em seguida, apontar para o arquivo fonte. Disponível a partir do CSS3. Sintaxe:

```
@font-face
{
    font-properties
}
```

Dica: Use letras minúsculas para a URL da fonte. Letras maiúsculas podem dar resultados inesperados no IE!

Exemplo:

```
@font-face
{
    font-family: myFirstFont;
    src: url(sansation_light.woff);
}
div
{
    font-family: myFirstFont;
}
```

4.16 Propriedades para tabelas

4.16.1 Border-collapse

Define se ou não as bordas da tabela podem ser recolhidas. Sintaxe:

```
border-collapse: separate|collapse|initial|inherit;
```

Onde:

- └ separate: este é o valor default. As bordas são separadas e espaços vazios não serão ignorados.
- └ collapse: bordas são fundidas em uma borda simples sempre que possível, e espaços vazios serão ignorados.

Exemplo:

```
table
{
    border-collapse: collapse;
}
```

4.16.2 Border-spacing

Define a distância entre as bordas adjacentes das células. Sintaxe:

```
border-spacing: length|initial|inherit;
```

Exemplo:

```
table
{
    border-collapse: separate;
    border-spacing: 10px 50px;
}
```

4.16.3 Caption-side

Define a colocação da legenda de uma tabela. Sintaxe:

```
caption-side: top|bottom|initial|inherit;
```

Exemplo:

```
caption
{
    caption-side:bottom;
}
```

4.16.4 Empty-cells

Define se mostra ou não bordas e background para células vazias na tabela.

Sintaxe:

```
empty-cells: show|hide|initial|inherit;
```

Exemplo:

```
table
{
    border-collapse:separate;
    empty-cells:hide;
}
```

4.16.5 Table-layout

Define o algoritmo de layout usado pela tabela. Disponível a partir do CSS2.

Sintaxe:

```
table-layout: auto|fixed|initial|inherit;
```

Onde:

- └ auto: este é o padrão. A largura da coluna é definida pelo maior conteúdo inquebrável nas células. Pode ser lento, uma vez que precisa ler todo o conteúdo da tabela, antes de determinar o layout final.
- └ fixed: a disposição horizontal só depende da largura da tabela e da largura das colunas, não do conteúdo das células. O navegador pode começar a exibir a tabela, uma vez que a primeira linha foi recebida.

Exemplo:

```
table
{
    table-layout:fixed;
}
```

4.17 Propriedades para listas

4.17.1 Counter-increment

Incrementa um ou mais contadores. Sintaxe:

```
counter-increment: none|id|initial|inherit;
```

Onde:

- └ none: valor default. O contador não será incrementado.
- └ id: o “id” define qual contador será incrementado. O número de conjuntos de quanto o contador será incrementado em cada ocorrência do seletor. O incremento padrão é 1. 0 ou valores negativos são permitidos. Se o id refere-se a um contador que não foi inicializado por counter-reset, o valor inicial padrão é 0.

Exemplo:

```
body
{
  counter-reset:section;
}

h1
{
  counter-reset:subsection;
}

h1:before
{
  counter-increment:section;
  content:"Section " counter(section) ". ";
}

h2:before
{
  counter-increment:subsection;
  content:counter(section) "." counter(subsection) " ";
}
```

4.17.2 Counter-reset

Cria ou reseta um ou mais contadores. Sintaxe:

```
counter-reset: none|name number|initial|inherit;
```

Onde:

- └ none:
- └ name number:

Exemplo:

```
body
{
  counter-reset:section;
}

h1
{
  counter-reset:subsection;
}

h1:before
{
  counter-increment:section;
  content:"Section " counter(section) ". ";
}
```

```
h2:before
{
    counter-increment:subsection;
    content:counter(section) "." counter(subsection) " ";
}
```

4.17.3 List-style

Define todas as propriedades para uma lista em uma só declaração. Assim como no background, você pode definir as propriedades individualmente. Sintaxe:

```
list-style: list-style-type list-style-position list-style-
image|initial|inherit;
```

Onde:

- ↳ **list-style-type**: define o tipo do marcador para a lista. Sintaxe: `list-style-type: value;` onde value pode ser: disc, armenian, circle, cjk-ideographic, decimal, decimal-leading-zero, georgian, hebrew, hiragana, hiragana-iroha, katakana, katakana-iroha, lower-alpha, lower-greek, lower-latin, lower-roman, none, square, upper-alpha, upper-latin, upper-roman.
- ↳ **list-style-position**: define a posição do marcador. Sintaxe: `list-style-position: inside|outside|initial|inherit.`
 - **inside**: recua o marcador do texto. Os marcadores aparecem dentro do fluxo de conteúdo.
 - **outside**: mantém o marcador para a esquerda do texto. Os marcadores aparecem fora do fluxo de conteúdo. Este é o padrão.
- ↳ **list-style-image**: Especifica uma imagem para o marcador de lista. Sintaxe: `list-style-image: none|url|initial|inherit;`

Exemplo:

```
ul.ex1
{
    list-style:square url("sqpurple.gif");
}

ul.circle {list-style-type:circle}
ul.square {list-style-type:square}
```

Básico das propriedades para interfaces.

4.17.4 Box-sizing

A propriedade de dimensionamento de caixa permite que você defina certos elementos para atender uma área de uma determinada maneira. Por exemplo, se você quiser duas caixas delimitadas lado a lado, ele pode ser alcançado através da criação de caixa de dimensionamento para "border-box". Isso força o navegador renderizar a caixa com a largura e altura especificadas. Coloque a borda e o preenchimento dentro da caixa. Disponível a partir do CSS3. Sintaxe:

```
box-sizing: content-box|border-box|initial|inherit;
```

Onde:

- └ content-box: valor padrão. Este é o comportamento de largura e altura, tal como especificado por CSS2.1. A largura especificada e altura (e min / max) aplicam-se à largura e à altura, respectivamente, da caixa de conteúdo do elemento. O preenchimento da borda do elemento são definidos e desenhados fora a largura e a altura especificadas.
- └ border-box: a largura especificada e altura (e min / max) propriedades neste elemento determinam a borda do elemento. Ou seja, qualquer preenchimento ou borda especificada no elemento é apresentado e desenhado dentro desta largura e altura especificadas. A largura do conteúdo e altura são calculadas subtraindo-se as larguras de fronteira e de preenchimento dos respectivos lados da "largura" especificados e propriedades 'altura' .

Exemplo:

```
div
{
  box-sizing:border-box;
  -moz-box-sizing:border-box; /* Firefox */
  width:50%;
  float:left;
}
```

4.17.5 Content

Usada com os pseudoelementos :before e :after para inserir conteúdo. Sintaxe:

```
content: normal|none|counter|attr|string|open-quote|close-quote|no-
open-quote|no-close-quote|url|initial|inherit;
```

Onde:

- └ normal: valor padrão. Define o conteúdo se especificado.
- └ none: nada.
- └ counter: define o conteúdo como um contador.
- └ attr: define o conteúdo como um atributo do seletor.
- └ string: define o conteúdo com o texto que você especificar.
- └ open-quote: define o conteúdo como uma aspas aberta.
- └ close-quote: define o conteúdo como uma aspas fechada.
- └ no-open-quote: remove a citação de abertura a partir do conteúdo, se especificado
- └ no-close-quote: remove a citação de fechamento a partir do conteúdo, se especificado.
- └ url: Define o conteúdo para ser um tipo de mídia (uma imagem, um som, um vídeo etc.).

Exemplo:

```
<style>
a:after
{
```

```

        content: " (" attr(href) ")";
    }

p
{
    counter-increment: myIndex;
}

p:before
{
    content:counter(myIndex);
}
</style>

<style>
p:before
{
    content:"Read this -";
}

p#hometown:before
{
    content:none;
}
</style>

```

4.17.6 Cursor

Define o tipo do cursos que será mostrado. Sintaxe:

`cursor: value;`

Onde:

↳ `value`: alguns dos valores que ele pode assumir:

- alias: indica um apelido para algo está a ser criado.
- all-scroll: indica que o conteúdo pode ser rolado, e em qual direção.
- auto: valor padrão, utiliza o cursor padão do browser.
- cell: o cursor indica que a célula pode ser selecionada.
- help: o cursor indica que o help está disponível.

Nota: confira no http://www.w3schools.com/cssref/pr_class_cursor.asp todas as opções.

Exemplo:

```

span.crosshair  {cursor:crosshair}
span.help      {cursor:help}
span.wait      {cursor:wait}

```

4.17.7 Icon

Permite ao autor a habilidade de estilizar um elemento com um ícone equivalente.

Disponível a partir do CSS3. Sintaxe:

`icon: auto|URL|initial|inherit;`

Exemplo:

```
img
{
    content:icon;
    icon:url(imgicon.png);
}
```

4.17.8 Outline

Define todas as propriedades outline (contorno) em uma declaração. Sintaxe:

`outline: outline-color outline-style outline-width|initial|inherit;`

Onde:

- └ `outline-color`: especifica a cor do contorno.
- └ `outline-style`: especifica o estilo do contorno.
- └ `outline-width`: especifica a espessura do contorno.

Exemplo:

```
<style>
p
{
    border:1px solid red;
    outline:green dotted thick;
}
</style>
```

4.17.9 Resize

Define se o elemento pode ser redimensionado ou não pelo usuário. Disponível a partir do CSS3. Sintaxe:

`resize: none|both|horizontal|vertical|initial|inherit;`

Nota: A propriedade de redimensionamento é suportado no Firefox, Chrome e Safari.

Exemplo:

```
div
{
    resize:both;
    overflow:auto;
}
```

4.17.10 Text-overflow

Define o que pode acontecer quando um texto pode passar por cima do conteúdo do elemento. Disponível a partir do CSS3. Sintaxe:

`text-overflow: clip|ellipsis|string|initial|inherit;`

Onde:

- └ `clip`: valor default. Recorta o texto.
- └ `ellipsis`: reinderiza com “...” para mostrar onde o texto foi cortado
- └ `string`: renderização a string dada para representar texto recortado

Exemplo:

```
div.test  
{  
    text-overflow:ellipsis;  
}
```



Atividades de aprendizagem

1. Qual a diferença entre as unidades de medida linear e absoluta?
2. Para que serve o estilo Aural?

Fique ligado!



Nesta unidade vimos como surgiu o CSS e sua importância para o desenvolvimento web, assim como suas principais propriedades e formas de uso. Mantenha-se atualizado sobre os recursos que surgem do CSS, assim como a capacidade dos navegadores para utilizar esses novos recursos.

Para concluir o estudo da unidade



O estudo desta unidade deu a você o conhecimento geral sobre o que é o CSS, como ele se tornou importante, assim como suas regras, conceitos e principais propriedades. Tudo isso junto lhe fornece as ferramentas necessárias para aplicar, ao html, a aparência planejada por um designer de forma eficiente e profissional. Além do contínuo estudo sobre as novidades e atualizações da W3S deixo aqui a recomendação de uma ferramenta útil no desenvolvimento web, a BootStrap: <<http://getbootstrap.com/>>.



Atividades de aprendizagem da unidade

1. Crie uma estrutura HTML com os seguintes elementos:
 - └ uma div no topo representando o cabeçalho da página;
 - └ uma div representando o conteúdo;
 - └ uma lista com 5 itens dentro da div que representa o conteúdo;
 - └ uma div representando o rodapé da página.
2. Defina uma imagem personalizada para os marcadores utilizando o estilo inline.
3. De acordo com o exercício anterior, defina um background-color para o body.
4. Um background diferente do body para os elementos cabeçalho, conteúdo e rodapé.
5. Determine que a div “conteudo” tenha no máximo 800px. Dentro da div “conteudo” adicione um elemento “p” com um conteúdo aleatório. Faça com que os elementos dentro da div “conteudo” tenham uma margem de 10px entre eles e defina um backgroud-color diferente para ele do background-color da div “conteúdo”.

Referências

THE CSS SAGA. Disponível em: <<http://www.w3.org/Style/LieBos2e/history/>>. Acesso em: 27 abr. 2014.

CSS — Curso W3C Escritório Brasil. Disponível em: <<http://www.w3c.br/pub/Cursos/CursoCSS3/css-web.pdf>>. Acesso em: 27 abr. 2014.

W3SCHOOLS.CSS Tutorial. Disponível em: <http://www.w3schools.com/css/css_intro.asp>. Acessado em: 27 abr 2014.

CSS Syntax Module Level 3. Disponível em: <<http://www.w3.org/TR/2014/CR-css-syntax-3-20140220/>>. Acesso em: 27 abr. 2014.

SELECTORS LEVEL 3 W3C RECOMMENDATION 29 SEPTEMBER 2011. Disponível em: <<http://www.w3.org/TR/selectors/>>. Acesso em: 27 abr. 2014.



ISBN 978-85-68075-59-3

A standard linear barcode representing the ISBN number 978-85-68075-59-3.

9 788568 075593 >