Manual de CSS, folhas de estilo

Autores do manual

Este manual foi criado pelos seguintes colaboradores de Criarweb.com:

Miguel Angel Alvarez - Tradução de JML
(19 capítulos)

Federico Elgarte
http://www.cssboulevar.com.ar/
(1 capítulo)

Serviweb
http://www.serviweb.es/
(1 capítulo)

Leonardo A. Correa
http://www.webnova.com.ar
(1 capítulo)

Manu Gutierrez
http://www.tufuncion.com
(1 capítulo)

Miguel Angel Alvarez - Tradução de Celeste Veiga
(3 capítulos)

Introdução às CSS

A linguagem HTML está limitada na hora de aplicar forma a um documento. Isto é assim porque foi concebido para outros usos (sobretudo científicos), diferente dos atuais, que são muito mais amplos.

Para solucionar estes problemas, os web designers utilizaram técnicas tais como a utilização de tabelas imagens transparentes para ajustá-las, a utilização de etiquetas que não são padrões do HTML e outras. Estas "trapaças" causaram muitas vezes problemas nas páginas na hora de sua visualização em distintas plataformas.

Ademais, os desenhadores se viram frustrados pela dificuldade com a qual, mesmo utilizando estes truques, encontravam-se planejando páginas sobre o papel, onde o controle sobre a forma do documento é absoluto.

Finalmente, outro antecedente que fez necessário o desenvolvimento desta tecnologia consiste em que as páginas web têm misturado em seu código HTML o conteúdo do documento com as etiquetas necessárias para lhe dar forma. Isto tem seus inconvenientes já que a leitura do código HTML se faz pesada e difícil na hora de buscar erros ou depurar as páginas. Mesmo que do ponto de vista da riqueza da informação e a utilidade das páginas na hora de armazenar seu conteúdo, é um grande problema que estes textos estejam misturados com etiquetas incrustadas para dar forma a estes: degrada-se sua utilidade.

Nestas páginas de CSS pretendemos tornar conhecida a tecnologia com um enfoque prático para que em poucos capítulos você possa usar as CSS de forma depurada, refletindo toda nossa experiência em seu uso. Não pretendemos explorar todos os aspectos da tecnologia já para realizar isto necessitaríamos uma extensão do livro inteiro.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Características e vantagens das CSS

A forma de funcionamento das CSS consiste em definir, mediante uma sintaxe especial, a forma de apresentação que lhe aplicaremos a:

- Um web inteiro, de modo que se pode definir a forma de todo o web de uma vez só.
- Um documento HTML ou página, que se pode definir a forma, em um pequeno pedaço da página.
- Uma porção do documento, aplicando estilos visíveis em um pedaço da página.
- Uma etiqueta em concreto, chegando inclusive a poder definir vários estilos diferentes para uma só etiqueta. Isto é muito importante já que oferece potência em nossa programação. Podemos definir, por exemplo, vários tipos de parágrafos: em vermelho, em azul, com margens, sem margens...

A potência da tecnologia é óbvia. Mas não fica só por aqui, já que ademais esta sintaxe CSS permite aplicar o documento formato de forma muito mais exata. Se antes o HTML ficava curto para planejarmos as páginas e teríamos que utilizar truques pra conseguir nossos efeitos,

agora temos muitas mais ferramentas que nos permite definir esta forma.

- Podemos definir a distância entre as linhas do documento.
- Pode-se aplicar recuo às primeiras linhas do parágrafo.
- Podemos colocar elementos na página com maior precisão, e sem lugar para erros.
- E muito mais, como definir a visibilidade dos elementos, margens, sublinhados, riscados...

E continuamos mostrando vantagens, já que com o HTML somente podíamos definir atributos nas páginas com pixels e porcentagens, agora podemos definir utilizando muito mais unidades como:

- Pixels (px) e porcentagem (%), como antes.
- Polegadas (in)
- Pontos (pt)
- Centímetros (cm)

Navegadores que o suportam

Esta tecnologia é bastante nova, porém nem todos os navegadores a suportam. De fato, somente os navegadores de Netscape versões da 4 em diante e de Microsoft a partir da versão 3 são capazes de compreender os estilos em sintaxe CSS. Ademais, cabe destacar que nem todos os navegadores implementam as mesmas funções de folhas de estilos, por exemplo, Microsoft Internet Explorer 3 não suporta nada em relação a camadas.

Isto quer dizer que devemos usar esta tecnologia com cuidado, já que muitos usuários não poderão ver os formatos que apliquemos às páginas com CSS. Sendo assim, utilizem as folhas de estilos quando estas não forem a supor causar um problema.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Uso das CSS I

Vamos agora descrever os diferentes usos das CSS introduzidos no capítulo anterior. Vamos por ordem descrevendo os pontos segundo sua dificuldade e importância.

CSS tem uma sintaxe própria, a veremos através de exemplos. Em seguida, veremos com detalhes.

Dividimos este capítulo em duas partes por sua extensão e por haver várias formas distintas de aplicar estilos. Aqui veremos as mais simples e no capítulo seguinte outras mais complicadas, mas mais potentes.

Pequenas partes da página

Para definir estilos em seções reduzidas de uma página se utiliza a etiqueta **<SPAN>**. Com seu atributo **style** indicamos em sintaxe CSS as características de estilos. Vemos com um exemplo, colocaremos um parágrafo onde vamos visualizar determinadas palavras na cor verde

```
<p>
Isto é um parágrafo com várias palavras <SPAN style="color:green">de cor verde</SPAN>. é muito fácil.
</p>
```

Que tem como resultado:

Isto é um parágrafo com várias palavras de cor verde. É muito fácil.

Estilo definido para uma etiqueta

Deste modo podemos fazer com que toda uma etiqueta mostre um estilo determinado. Por exemplo, podemos definir um parágrafo inteiro de cor vermelha e outro de cor azul. Para isso, utilizamos o atributo **style**, que é admitido por todas as etiquetas do HTML (sempre e quando dispormos de um navegador compatível com CSS).

```
<p style="color:#990000">
Isto é um parágrafo de cor
vermelha.
</p>
<p style="color:#000099">
Isto é um parágrafo de cor azul.
</p>
```

Que tem como resultado:

Isto é um parágrafo de cor vermelha.

Isto é um parágrafo de cor azul.

Estilo definido em uma parte da página

Com a etiqueta **<DIV>** podemos definir seções de uma página e aplicar estilos com o atributo **style**, ou seja, podemos definir estilos de uma só vez a todo um bloco da página.

```
<div style="color:#000099; font-weight:bold">
<h3>Estas etiquetas vão em <i>azul e negrito</i></h3>
<p>
Continuamos dentro do DIV, por isso permanecem os
estilos.
</p>
</div>
```

Que tem como resultado:

Estas etiquetas vão em *azul e negrito*

Continuamos dentro do DIV, por isso permanecem os estilos.

No próximo capítulo seguiremos vendo formas mais avançadas de usar as CSS.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Uso das CSS II

Estilo definido para toda uma página

Podemos definir no cabeçalho do documento, estilos para que sejam aplicados a toda página. É uma maneira muito cômoda de dar forma ao documento e muito potente, já que estes estilos continuarão em toda a página e com isso, economizaremos muitas etiquetas HTML que dão

forma ao documento. Além disso, se desejamos mudar os estilos da página simplesmente faremos de uma vez só.

Este exemplo é mais complicado visto que utiliza a sintaxe CSS de forma mais avançada. Mas não se preocupe, pois com os exemplos você irá aprendendo seu uso e mais tarde comentaremos a sintaxe com mais profundidade.

No exemplo vemos que se utiliza a etiqueta <STYLE> colocada no cabeçalho da página para definir os diferentes estilos do documento.

Simplificando, entre as etiquetas <STYLE> e </STYLE>, se coloca o nome da etiqueta que queremos definir os estilos e entre chaves -{ }- colocamos em sintaxe CSS as características de estilos.

```
<html>
<head>
<title>Exemplo de estilos para toda uma página</title>
<STYLE type="text/css">
<!--
H1 {text-decoration: underline; text-align:center}
P {font-Family:arial,verdana; color: white; background-color: black}
BODY {color:black;background-color: #cccccc; text-indent:1cm}
// -->
</STYLE>
</head>

<body>
<h1>Página com estilos</h1>
Bem-vindos...
<p>Desculpe ser tão breve, mas isto é um exemplo sem importância</p>
</body>
</html>
```

Como se pode apreciar no código, definimos que a etiqueta <H1> se apresentará:

- Sublinhada
- Centralizada

Também, por exemplo, definimos que no corpo do texto inteiro da página (etiqueta <BODY> serão aplicados os seguintes estilos:

- Cor do texto negro
- Cor do fundo acinzentado
- Margem lateral de 1 centímetro

Cabe destacar que se aplicamos estilos à etiqueta <BODY>, estes serão herdados pelo resto das etiquetas do documento. Isto será sempre assim, e no caso de que não volte a definir estes estilos nas seguintes etiquetas, dominará a etiqueta mais concreta. Este detalhe pode ser visto na etiqueta <P>, que tem definidos estilos que já foram definidos para <BODY>. Os estilos que são aplicados são os da etiqueta <P> que é mais concreta.

Por último, é válido apreciar os comentários HTML que englobam toda a declaração de estilos:

<!--Declaração de estilos-->. Estes comentários são utilizados para que os navegadores antigos que não compreendem a sintaxe CSS, não incluam esse texto no corpo da página. Caso contrário, os navegadores antigos (por exemplo Netscape 3) escreveriam esse "feio código" na página.

Estilo definido para todo o site web

Uma das características mais potentes da programação com folhas de estilo, consiste em que podemos definir os estilos de todo um site web de uma só vez. Isto se consegue criando um arquivo onde somente colocamos as declarações de estilos de página e linkando todas as páginas do site com esse arquivo. Dessa forma, todas as páginas compartem uma mesma declaração de estilos e, portanto, se a mudamos, mudarão todas as páginas. Todas as vantagens acrescentadas de que se economiza em linhas de código HTML (o que reduz o peso do documento) e se evita a moléstia de definir uma e outra vez os estilos com o HTML, tal como foi comentado anteriormente.

Veremos agora como o processo para incluir estilos com o arquivo externo.

1 - Criamos o arquivo com a declaração de estilos

É um arquivo de texto normal, que pode ter qualquer extensão, apesar de podermos atribuir a extensão .css para lembrarmos que tipo de arquivo é. O texto que devemos incluir deve ser escrito exclusivamente em sintaxe CSS, ou seja, seria errado incluir o código HTML nas etiquetas e etc. Podemos ver um exemplo a seguir:

```
P {
font-size : 12pt;
font-family : arial,helvetica;
font-weight : normal;
}
H1 {
font-size : 36pt;
font-family : verdana,arial;
text-decoration : underline;
text-align : center;
background-color : Teal;
}
TD {
font-size : 10pt;
font-family : verdana,arial;
text-align : center;
background-color : 666666;
}
BODY {
background-color : #006600;
font-family : arial;
color : White;
}
```

2 - Linkamos a página com a folha de estilos

Para isso, vamos colocar a etiqueta <LINK> com os atributos:

- rel="STYLESHEET" indicando que o link é com uma folha de estilos
- type="text/css" porque o arquivo é de texto, em sintaxe CSS
- href="estilos.css" indica o nome do arquivo fonte dos estilos

Vejamos uma página web inteira que linka com a declaração de estilos anterior.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
 <link rel="STYLESHEET" type="text/css" href="estilos.css">
 <title>P&aacute;gina que lê estilos</title>
</head>
<body>
<h1>P&aacute;gina que lê estilos</h1>
Esta p&aacute;gina tem no cabeçalho a etiqueta necessária para linkar com a folha de estilos. É muito
f&aacute;cil.
<br>
<br>
<table width="300" cellspacing="2" cellpadding="2" border="0">
<tr>
   <td>Isto est&aacute; dentro de um TD, logo tem estilo próprio, declarado no arquivo externo</td>
</tr>
<tr>
   <td>A segunda fila do TD</td>
</tr>
</table>
</body>
</html>
```

O resultado conseguido por ser visto aqui.

Regras de importância nos estilos

Os estilos se herdam de uma etiqueta para outra, tal como foi indicado anteriormente. Por exemplo, se temos declarado no <BODY> alguns estilos, em geral, estas declarações também afetarão à etiquetas que estejam dentro desta etiqueta, ou o que é o mesmo, dentro de todo o corpo.

Em muitas ocasiões mais de uma declaração de estilos afeta a mesma parte da página. Sempre se leva em conta a declaração mais particular. Porém, as declarações de estilos podem ser realizados de múltiplas formas e com várias etiquetas, também entre essas formas existe uma hierarquia de importância pra resolver conflitos entre várias declarações de estilos diferentes para uma mesma parte da página. Pode ser visto a seguir, esta hierarquia, primeiro colocamos as formas de declarações mais gerais, e portanto, menos respeitadas em caso de conflito:

- Declaração de estilos com arquivo externo. (para todo o site web)
- Declaração de estilos para toda a página. (com a etiqueta <STYLE> no cabeçalho da página)
- Estilos definidos em uma parte da página. (com a etiqueta <DIV>)
- Definidos em uma etiqueta em concreto. (Utilizando o atributo style na etiqueta em questão)
- Declaração de estilo para uma pequena parte do documento. (Com a etiqueta <SPAN>)

Já vimos como incluir estilos na página, de todas as maneiras possíveis e fizemos uma revisão com a lista anterior. Agora você já está em condições de começar a usar as folhas de estilo em cascata para melhorar sua página e aumentar a produtividade de seu trabalho. Mas esteja atento aos capítulos seguintes onde irá aprende as lições que lhe faltam para dominar bem a

matéria: conhecer a sintaxe, os diferentes atributos de estilos e outras coisas que irão melhorar sua página.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Outra maneira de definir estilos em um arquivo externo

Veremos agora outra maneira de importar uma declaração externa de estilos CSS: @import url ("arquivo_a_importar.css"), que se utiliza para importar umas declarações de estilos salvadas na rota que se indica entre parênteses. (as aspas são opcionais, mas os parêntesis são obrigatórios, pelo menos, em Explorer).

Deve-se incluir na declaração de estilos global a uma página, ou seja, entre as etiquetas <style type="text/css"> e </style>, que se colocam no cabeçalho do documento.

É importante assinalar que a sentença de importação do arquivo CSS deve ser escrita na primeira linha da declaração de estilos, algo parecido ao código seguinte.

```
<style type="text/css">
@import url ("estilo.css");
body{
background-color:#ffffcc;
}
</style>
```

O funcionamento é o mesmo que se escrevêssemos todo o arquivo a importar dentro das etiquetas dos estilos, com a garantia de que, se redefinimos dentro do código HTML (entre as etiquetas </style>) estilos que haviam ficado definidos no arquivo externo, os que se aplicarão serão os que tivermos redefinido.

Assim, no exemplo anterior, mesmo que tivéssemos definido em estilo.css uma cor de fundo para a página, a cor que prevaleceria seria o definido em seguida da importação: #ffffcc

A diferença entre este tipo de importação do tipo e a que vimos anteriormente:

<link rel="stylesheel" type="text/css" href="folha.css">

É que @import url ("estilo.css") costuma ser utilizado quando existe umas pautas básicas no trabalho com os estilos (que se definem em um arquivo a importar) e uns estilos específicos para cada página, que se definem a seguir, dentro do código HTML entre as etiquetas </style>, como é o caso do exemplo visto anteriormente.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Sintaxe CSS

Tal como vimos nos exemplos, a sintaxe é bastante simples e repetitiva. Vamos vê-la:

- Para definir um estilo se utilizam atributos como font-size,text-decoration... seguidos de dois pontos e o valor que lhe desejamos atribuir. Podemos definir um estilo a base de definir muitos atributos separados por ponto e vírgula.
  Exemplo:
  font-size: 10pt; text-decoration: underline; color: black; (o último ponto e vírgula da lista de atributos é opcional)
- Para definir o estilo de uma etiqueta basta a escrevermos seguida da lista de atributos fechados entre chaves.
  Exemplo:
  H1{text-align: center; color:black}
- Os valores que se podem atribuir aos atributos de estilo podem ser vistos em uma tabela no seguinte capítulo. Muitos destes valores são unidades de medida, por exemplo, o valor do tamanho de uma margem ou o tamanh da fonte. As unidades de medida são as seguintes:

Pontos	pt
Polegadas	in
Centímetros	cm
pixels	px

Até aqui deixamos claro tudo relativo à sintaxe. No próximo capítulo você poderá encontrar uma lista de atributos das folhas de estilo em cascata.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Atributos das folhas de estilo

Tanto para praticar em sua aprendizagem como para trabalhar com as CSS o melhor é dispor de uma tabela onde se vejam os distintos atributos e valores de estilos que podemos aplicar às páginas web.

Aqui você pode ver a tabela dos atributos CSS, e tê-la a mão quando for utilizar as CSS.

Nome do atributo	Possíveis valores	Exemplos
FONTES - FONT		
color	valor RGB ou nome da cor	color: #009900; color: red;
Serve para indicar a cor do texto. Admitem quase todas as etiquetas de HTML. Nem todos os nomes de cores são admitidos no padrão, é aconselhável então utilizar o valor RGB.		
font-size	xx-small | x-small | small | medium | large | x-large | xx-large Unidades de CSS	font-size:12pt; font-size: x-large;
Serve para indicar o tamanho das fontes de forma mais rígida e com maior exatidão.		
font-family	serif | sans-serif | cursive | fantasy | monospace Todas as fontes habituais	font-family:arial,helvetica; font-size: fantasy;

Com este atributo indicamos a familia de tipografia do texto. Os primeiros valores são genéricos, ou seja, os exploradores as compreendem e utilizam as fontes que o usuário tenha em seu sistema.
Também podem se definir com tipografias normais, como ocorria em html. Se o nome de uma fonte tem espaços se utilizam aspas para que se entenda bem.

font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	font-weight:bold; font-weight: 200;

Serve para definir a largura dos caracteres, ou com outras palavras, para colocar negritas com total liberdade.
Normal e 400 são o mesmo valor, assim como bold e 700.

font-style	normal | italic | oblique	font-style:normal; font-style: italic;

É o estilo da fonte, que pode ser normal, itálico ou oblíquo. O estilo oblíquo é similar ao itálico.

PARÁGRAFOS - TEXT

line-height	normal e unidades CSS	line-height: 12px; line-height: normal;

A altura de uma linha, e portanto, o espaçamento entre linhas. É uma dessas características que não se podiam modificar utilizando HTML.

text-decoration	none | [underline || overline || line-through]	text-decoration: none; text-decoration: underline;

Para estabelecer a decoração de um texto, ou seja, se está sublinhado, tachado, etc.

text-align	left | right | center | justify	text-align: right; text-align: center;

Serve para indicar o alinhamento do texto. É interessante destacar que as folhas de estilo permitem o justificado de texto, mesmo assim, lembre-se que não tem porque funcionar em todos os sistemas.

text-indent	Unidades CSS	text-indent: 10px; text-indent: 2in;

Um atributo que serve para fazer recuos ou margens nas páginas. Muito útil e novo.

text-transform	capitalize | uppercase | lowercase | none	text-transform: none; text-transform: capitalize;

Permite-nos transformar o texto, fazendo com que tenha a primeira letra em maiúsculas de todas as palavras, tudo em maiúsculas ou minúsculas.

FUNDO - BACKGROUND

Background-color	Uma cor, com seu nome ou seu valor RGB	background-color: green; background-color: #000055;

Serve para indicar a cor de fundo de um elemento da página.

Background-image	O nome da imagem com seu caminho relativo ou absoluto	background-image: url(mármol.gif) ; background-image: url(http://www.x.com/fondo.gif)

Colocamos com este atributo uma imagem de fundo em qualquer elemento da página.

BOX - CAIXA

Margin-left	Unidades CSS	margin-left: 1cm; margin-left: 0,5in;

Indicamos com este atributo o tamanho da margem à esquerda.

Margin-right	Unidades CSS	margin-right: 5%; margin-right: 1in;

Utiliza-se para definir o tamanho da margem à direita.

Margin-top	Unidades CSS	margin-top: 0px; margin-top: 10px;

Indicamos com este atributo o tamanho da margem acima da página.

Margin-bottom	Unidades CSS	margin-bottom: 0pt; margin-top: 1px;

Com ele indica-se o tamanho da margem na parte debaixo da página.

Padding-left	Unidades CSS	padding-left: 0.5in; padding-left: 1px;

Indica o espaço inserido, pela esquerda, entre a borda do elemento que contem e o conteúdo deste. É parecido ao atributo cellpadding das tabelas.
O espaço inserido tem o mesmo fundo que o fundo do elemento que contem.

Padding-right	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;
Indica o espaço inserido, neste caso pela direita, entre a borda do elemento que contém e o conteúdo deste. É parecido ao atributo cellpadding das tabelas. O espaço inserido tem o mesmo fundo que o fundo do elemento que contem.		
Padding-top	Unidades CSS	padding-top: 10pt; padding-top: 5px;
Indica o espaço inserido, por cima, entre a borda do elemento que contem e o conteúdo deste.		
Padding-bottom	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;
Indica o espaço inserido, neste caso por baixo, entre a borda do elemento que contém e o conteúdo deste.		
Border-color	cor RGB e nome de cor	border-color: red; border-color: #ffccff;
Para indicar a cor da borda do elemento da página a qual aplicamo-la. Pode-se colocar cores separadas com os atributos border-top-color, border-right-color, border-bottom-color, border-left-color.		
Border-style	none | dotted | solid | double | groove | ridge | inset | outset	border-style: solid; border-style: double;
O estilo da bordea, os valores significam: none=nenhuma borda, dotted=pontilhado (não parece funcionar), solid=sólido, double=borda dupla, e desde groove até outset são bordas com vários efeitos 3D.		
border-width	Unidades CSS	border-width: 10px; border-width: 0.5in;
O tamanho da borda do elemento ao qual o aplicamos.		
float	none | left | right	float: right;
Serve para alinhar um elemento à esquerda ou à direita fazendo com que o texto se agrupe ao redor de tal elemento. Igual que o atributo align em imagens em seus valores right e left.		
clear	none | right | left	clear: right;
Se este elemento tem a sua altura imagens ou outros elementos alinhados à direita ou à esquerda, com o atributo clear fazemos com que se coloque em um lugar onde já não tenha estes elementos ao lado que indicamos.		

A especificação de estilos CSS é muito ampla, mas acredito que a imensa maioria está aqui definida, mas claro, com a seleção dos mais importantes.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Avançados truques com CSS

As folhas de estilos são um assunto muito longo o qual já foram escritos livros inteiros. Nós nos centramos nos temas mais práticos e por isso vamos já iremos acabar com este capítulo com alguns pontos.

Definir estilos utilizando classes

As classes nos servem para criar definições de estilos que podem ser utilizadas repetidas vezes.

Uma classe pode ser definida entre as etiquetas <STYLE> (no cabeçalho do documento), ou em um arquivo externo à página. Para definí-las utilizamos a seguinte sintaxe, um ponto seguido do nome da classe e entre chaves os atributos de estilos desejados. Desta maneira:

.nomedaclasse {atributo: valor;atributo2:valor2; ...}

Uma vez tendo uma classe, podemos utilizá-la em qualquer etiqueta HTML. Para isso utilizaremos o atributo class, colocando como valor o nome da classe, desta forma:

<ETIQUETA class="nomedaclasse">

Exemplo da utilização de classes

```
<html>
<head>
 <title>Exemplo da utilização de classes</title>
 <STYLE type="text/css">
  .fundonegroletrasbrancas {background-color:black;color:white;font-size:12;font-family:arial}
  .letrasverdes {color:#009900}
 </STYLE>
</head>
<body>
<h1 class=letrasverdes>Título 1</h1>
<h1 class=fundonegroletrasbrancas>Título 2</h1>
<p class=letrasverdes>
Isto é um parágrafo com estilo de letras verdes</p>
<p class=fundonegroletrasbrancas>
Isto é um parágrafo com estilo de fundo negro e as letras brancas. É tudo!</p>
</body>
</html>
```

Ver o exemplo anterior

Estilo nos links

Uma técnica muito habitual, que se pode realizar utilizando as folhas de estilo em cascada e não se podia em HTML, é a definição de estilos nos links, tirando os sublinhados ou fazer links na mesma página com distintas cores.

Para aplicar estilo aos links devemos definí-los para os distintos tipos de links que existem (visitados, ativos...). Utilizaremos a seguinte sintaxe, na declaração de estilos global da página (<STYLE>) ou do site (Definição em arquivo externo):

Links normais
A:link {atributos}

Links visitados
A:visited {atributos}

Links ativos (Os links estão ativos no preciso momento em que se clica sobre eles)
A:active {atributos}

Links hover (Quando o mouse está em cima deles, funciona somente em explorer)
A:hover {atributos}

O atributo para definir links sem sublinhado é**text-decoration:none**, e para dar cor é color:sua cor.

Também podemos definir o estilo de cada link na própria etiqueta <A>, com o atributo style. Desta maneira podemos fazer com que determinados links da página seja vistas de forma distinta.

Exemplo de estilos nos links

```
<html>
<head>
 <title>Exemplos de estilo nos links</title>
 <STYLE type="text/css">
  A:link {text-decoration:none;color:#0000cc;}
  A:visited {text-decoration:none;color:#ffcc33;}
  A:active {text-decoration:none;color:#ff0000;}
  A:hover {text-decoration:underline;color:#999999;font-weight:bold}
 </STYLE>
</head>
<body>
<a href="http://dominioinexistente.nãofunciona.com">Link normal</a>
<br>
<br>
<a href="links.html">Link visitado</a>
Clicar este link para vê-lo ativo,
passar o mouse por cima para que mude.
</body>
</html>
```

Ver o exemplo anterior

URL como valor de um atributo:

Determinados atributos de estilos, como **background-image** necessitam uma URL como valor, para indicá-las podemos definir tanto caminhos relativos como absolutos. Assim, podemos indicar a URL da imagem de fundo destas duas maneiras:

background-image: url(fundo.gif) No caso de que a imagem esteja no mesmo diretório que a página.
background-image: url(http://www.criarweb.com/imagens/fundo.gif) ç

Ocultar estilos em navegadores antigos

No caso de definir dentro da etiqueta <STYLE> umas declarações de estilos devemos certificarmos que estas não vão ser imprimidas na página com navegadores antigos. Pensar em um navegador que não reconheça a etiqueta <STYLE>, pensará que corresponde com algo que não entende e se esquecerá da etiqueta. O que encontra a seguir é um texto normal e fará com que este se veja na página, como com qualquer outro texto.

Para evitar isso devemos ocultar com comentários HTML (<!-- isto é um comentário -->) tudo que tiver dentro da etiqueta <STYLE>.

Deste modo terminamos nosso manual de CSS. Esperamos poder ter ajudar a fazer páginas melhores e mais rapidamente.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

O que são as camadas?

Vejamos uma pequena introdução ao que são as camadas, a etiqueta HTML <DIV> utilizada para construí-las e os atributos CSS com os quais podemos aplicar estilos.

Como já vimos em nosso manual de CSS, <SPAN> serve para aplicar estilo a uma pequena parte de uma página HTML. Por exemplo, com ela poderíamos fazer com que uma parte de um parágrafo fosse colorida de vermelho. Com <SPAN> não é habitual englobar um pedaço muito grande de texto, por exemplo que ocupe vários parágrafos.

Com <DIV> también podemos aplicar estilo a partes de la página HTML.

A diferença entre <SPAN> e <DIV> é que com esta última sim que podemos aplicar estilo a uma parte mais ampla da página, por exemplo com três parágrafos. **Além do mais, a etiqueta <DIV> tem um uso adicional que é o de criar divisões nas páginas as quais poderemos aplicar uma quantidade adicional de atributos para modificar seus comportamentos**. Por exemplo, com o atributo align de HTML podemos alinhar a divisão ao centro, à esquerda, à direita ou justificado. Porém **seu uso mais destacado é o de converter essa divisão em uma camada**.

Uma camada é uma divisão, uma parte da página, **que tem um comportamento muito independente** dentro da janela do navegador, já que podemos colocá-la em qualquer parte da mesma e poderemos mover por ela independentemente, por exemplo. Muitos dos efeitos mais comuns do DHTML se baseiam no uso de camadas.

As etiquetas <LAYER> e <ILAYER> têm como objetivo construir camadas, mas estas não são compatíveis mais do que com Netscape, por isso é mais recomendável utilizar a etiqueta <DIV> para fazer camadas.

Os atributos que podemos aplicar a estas etiquetas, mas em concreto às duas recomendadas <SPAN> e <DIV>, são principalmente de estilos CSS. Estes atributos nos permitem, como podemos ver no manual de folhas de estilo em cascata de criarweb, modificar de uma maneira muito exaustiva a apresentação dos conteúdos na página. Para aplicar estilos a estas etiquetas utiliza-se o atributo de HTML style, desta forma:

```
<SPAN style="text-decoration:underline;font-weight:bold">...</SPAN>

<DIV style="color:red;font-size:10px">...</DIV>
```

Como podemos ver muitos exemplos no manual de CSS, nos referimos a ele para ampliar esta informação. Mas não havíamos visto ainda uma série de atributos que nos servem para posicionar a divisão na página como uma camada. Estes atributos podem ser aplicados à etiqueta <DIV> que é a qual servia para criar camadas compatíveis com todos os navegadores.

Os atributos para que a divisão seja uma camada são vários e podem ser vistos a seguir.

```
<div id="c1" style="position:absolute; left: 200px; top: 100px;">
Oi!
</div>
```

O primeiro, **position**, indica que se posicione de maneira absoluta na página e os segundos, **left** e **top**, são a distância desde a borda esquerda da página e a borda superior.

Existem outros atributos especiais para camadas como **width** e **height** para indicar a largura e a altura da capa, **Z-index** que serve para indicar que camadas se vêem em cima de que outras, **clip** que serve para recortar uma camada e fazer com que partes dela não sejam visíveis, ou **visibility** para definir se a camada é visível ou não. Estes, e outros atributos veremos no próximo capítulo, onde falaremos do posicionamento de camadas.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Atributos para camadas

Vimos no capítulo anterior o que são as camadas e algumas pequenas mostras sobre como criá-las dando algum estilo.. Agora vamos ver detalhadamente os atributos específicos para aplicar posicionamento a uma camada e outros estilos.

Antes de mais nada, cabe dizer que uma camada pode ter qualquer atributo de estilos dos que vimos no manual de CSS. Assim, o atributo color indica a cor do texto da camada, o atributo font-size indica o tamanho do texto e assim com todos os atributos já vistos.

Agora então, existem uma série de atributos que servem para indicar a forma, o tamanho das camadas, a visibilidade, etc, que não vimos nos capítulos anteriores e que veremos a seguir.

Atributo position

Indica o tipo de posicionamento da camada. Pode ter dois valores, relative ou absolute.

- relative indica que a posição da camada é relativa ao lugar onde se estava escrevendo na página no momento de escrever a camada com sua etiqueta

- absolute indica que a posição da camada se calcula com relação ao ponto superior esquerdo da página.

Atributo top

Indica a distância em vertical onde será colocada a camada. Se o atributo position é absolute, top indica a distância da borda superior da camada com relação a borda superior da página. Se o atributo position era relative, top indica a distância desde onde se estava escrevendo nesse momento na página até a borda superior da camada.

Atributo left

Basicamente funciona igual ao atributo top, com a diferença que o atributo left indica a distância na horizontal a que estará situada a camada.

Atributo height

Serve para indicar o tamanho da camada na vertical, ou seja, sua altura.

Atributo width

Indica a largura da camada

Atributo visibility

Serve para indicar se a camada pode ser vista na página ou se permanece oculta ao usuário. Este atributo pode ter três valores.

- visible serve para indicar que a camada poderá ser vista.
- hidden indicará que a camada está oculta.
- inherit é o valor por padrão, que quer dizer que herda a visibilidade da camada onde está colocada a camada em questão. Se a camada não está colocada dentro de nenhuma outra se

supõem que está colocada na camada do documento, que é toda a página e que sempre está visível.

Atributo z-index

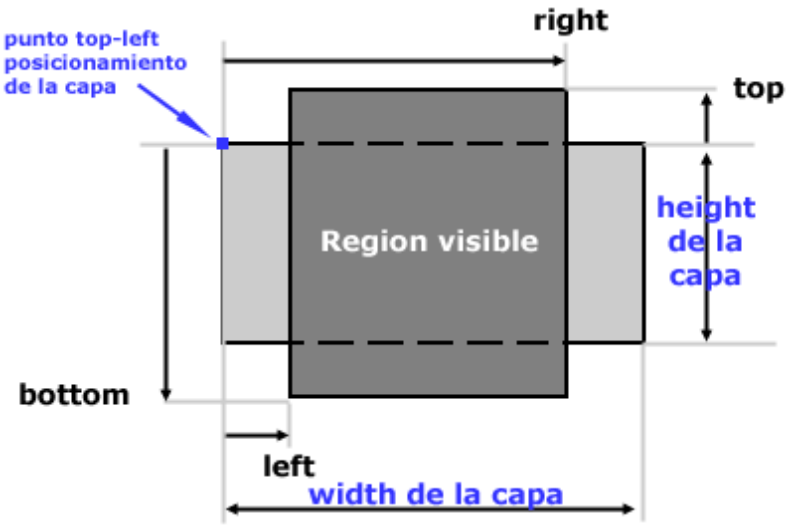
Serve para indicar a posição sobre o eixo que terão as distintas camadas da página. Com outras palavras, com z-index podemos dizer quais camadas serão vistas em cima ou embaixo de outras, no caso de que estejam sobrepostas. O atributo z-index toma valores numéricos e a mayor z-index, mais em cima será vista na página.

Atributo clip

É um atributo um pouco difícil de explicar. Em concreto serve para recortar determinadas áreas da camada e que não podem ser vistas. Uma camada que está visível se pode recortar para ser vista, mas não será visto nenhum pedaço desta. O cliping se indica por meio de 4 valores, com esta sintaxe.

rect (<top>, <right>, <bottom>, <left>)

Os valores <top>, <right>, <bottom> y <left> indicam distâncias que se podem neste esquema.



Este é um exemplo de camada que utiliza todos os atributos que vimos neste artigo e em algum mais para aplicar estilo à camada.

```
<div style="clip: rect(0,158,148,15); height: 250px; width: 170px; left: 10px; top: 220px; position: absolute;
visibility: visible; z-index:10; font-size: 14pt; font-family: verdana; text-align: center; background-color: #bbbbbb">

Esta camada tem um cliping, por isso se vê entrecortada.
<br>
<br>
Isto é uma camada de prova

</div>
```

Pode ser visto o exemplo em uma página web, onde também poderá ser apreciado o efeito conseguido ao realizar o cliping.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Problema com o posicionamento absoluto de camadas

Recebi uma consulta em meu correio eletrônico sobre a colocação de camadas de maneira absoluta, mas na que não nos importa a definição da tela do usuário e outros ir e vir dos elementos HTML. Nosso companheiro expressou sua dúvida da seguinte maneira:

Se trabalhamos com position: absolute dando um left e um top funciona se tiver sua página alinhada à esquerda. Minha página está alinhada no centro, então o que acontece é que dependendo da resolução da tela que tenha (largura de 800px,1024px,etc) a página fica sambando e não enquadra nada.

Primeiro de tudo, devemos saber que se trabalhamos com o position relative as camadas se colocam no lugar onde aparecem dentro do código HTML. Deste modo, se colocamos uma camada com position relative dentro de uma célula de uma tabela, tal camada apareceria dentro da célula onde a estamos colocando, independentemente do lugar onde se situa a célula ao mudar a definição da tela.

O problema desta solução é que a camada faria crescer a célula da tabela onde queremos colocá-la (igual a qualquer outro elemento HTML que colocássemos dentro da tabela) e é muito provável que nosso desenho não nos permitia este fato. Certamente já teria notado este problema e se é não é assim, convido-lhe a que crie a camada que você tenta colocar com o atributo position a relative para ver se com isso seu problema já está resolvido.

Em todos os casos, a camada que tentamos colocar terá que ter o position absolute, porque com relative não consertamos totalmente o problema. Então voltaremos ao problema inicial, que era situar a camada com position absolute no lugar exato, independentemente da definição de tela.

A solução final que proponho passa por aplicar algum truquezinho. De fato, estive a alguns dias atrás me perguntando sobre esta questão e afinal encontrei a solução, apesar de não ter vindo de mim, e sim que a extrai de www.cross-browser.com.

A idéia é um pouco complexa e colocá-la em ação devemos realizar uma série de ações que, sinceramente, considero excessivas para um problema inicialmente simples. Sendo assim, não se assustem com o que vou expor a seguir, pois logo tratarei de explicar um pouco melhor.

Nosso esquema de trabalho consistirá em uma camada com posição relativa, que nos servirá de "âncora" e outra com a posição absoluta, onde colocaremos o conteúdo final a mostrar na camada.

Colocaremos a camada relativa no lugar aproximado onde queremos que apareça a camada absoluta. Posicionaremos a camada absoluta, uma vez carregada a página, em um lugar próximo à camada relativa. Supostamente, vamos ter que realizar estas ações com Javascript, que é a linguagem que nos permite atualizar as posições das capas dinamicamente.

Detalhadamente

Dissemos que haveria que colocar uma camada perto do lugar onde tem de aparecer a camada com position absolute. Insisto em que as camadas relativas sejam colocadas no lugar onde as colocamos dentro do código HTML, pois será fácil colocar a capa relativa no lugar exato e que este lugar seja válido para qualquer definição.

A segunda capa, a que Tem o conteúdo final, a colocamos inicialmente em uma posição qualquer e escondida, de forma que não se veja que está mal colocada. Uma vez terminada de carregar a página, poderemos acessar a posição da camada relativa, extraindo seus valores top e left e colocando-os em os correspondentes top e left da capa com posição absoluta. Uma vez marcada a posição da capa absoluta podemos torná-la visível.

Na imagem seguinte colocamos a camada com posição relativa no link. Na verdade haveria três camadas para poder posicionar outras tantas camadas com posição absoluta. A parte que vemos sombreada de verde corresponde ao espaço que abarcaria a camada relativa.

Sua posição seria a que está marcada por um "x " vermelho que aparece em sua esquina esquerda. Tal posição depende do lugar onde apareça os links na página.

Logo, com Javascript deveríamos atribuir a posição da camada absoluta de uma maneira parecida a esta.

left da capa absoluta = left da capa relativa
top da capa absoluta = top da capa relativa + altura da capa relativa

Podemos somar algum píxel a mais à posição da camada, se é que queremos movê-la um pouco abaixo e à direita, tal como vimos na imagem.

Não pretendo neste artigo, explicar como se fazem essas operações de Javascript. Advirto que se não se conhece nada de Javascript será impossível se dispor de uma tarefa tão tediosa como o maneja de camadas. Se por o contrário, já tivemos contato com Javascript e DHTML anteriormente, não deveria ser um problema realizar essas ações.

Artigo por ***Miguel Angel Alvarez - Tradução de JML***

Construção CSS

Neste artigo vamos conhecer a construção de páginas utilizando Folhas de estilos em cascata (CSS). Veremos como realizar este tipo de construção, junto com algumas vantagens e inconvenientes. Para muitos será ainda um campo por explorar. Embora não vamos entrar em grandes detalhes, vamos tentar tornar conhecida a construção com CSS para tirar as possíveis dúvidas por parte do leitor. Nos próximos capítulos ampliaremos a informação e oferecemos tutoriais mais práticos.

Como se pode aprender no Manual de CSS, as folhas de estilo em cascata ajudam a separar o conteúdo da forma, ou seja, os elementos que compõem uma página da forma com a qual se mostram. Ademais, CSS ajuda em grande medida à definição de estilos na página, já que

permite ajustar de uma forma muito mais precisa qualquer aspecto de qualquer elemento da página.

A construção com CSS leva a utilização das folhas de estilo a seu grau máximo, de maneira que qualquer definição do aspecto da página se realiza na declaração CSS que linkamos com o documento HTML. Para definir a situação dos elementos na página se utilizam as camadas, a quais se aplica um posicionamento através também das folhas de estilo.

Para criar as camadas se utilizam etiquetas <DIV>, nas que se introduzem os elementos que quisermos que apareçam na página. Os elementos dentro dos <DIV> também podem se acrescentar, para herdar as propriedades e posicionamento das camadas pai.

Na construção por camadas se definem unicamente etiquetas <DIV> e as tabelas só se utilizam para mostrar informação tabulada, ou seja, mostrada em filas e colunas. Cabe assinalar que na construção tradicional se utilizam as tabelas para ajustar a posição dos elementos na página. Certamente, muito dos leitores estão bem familiarizados com o uso de tabelas para construir uma web, pois se trata de uma técnica bastante simples, embora muito bastante pouco prática e que complica bastante o código das páginas web resultantes.

A construção por tabelas é uma forma de criar webs mais evoluída e que melhora muito a construção tradicional, embora tenha também seus inconvenientes.

Vejamos primeiro, algumas das **vantagens** da construção CSS:

- **A separação do conteúdo da página e do estilo ou aspecto com o qual se devem mostrar**. Ter em conta que, quanto mais separarmos estes dois elementos, mais simples será a manutenção das páginas e o processamento da informação. Com isso também poderemos obter páginas mais limpas e claras.
- **Economia na transferência**. Se todos os estilos e posições dos elementos se introduzem em um documento externo, liberaremos o código da página e ocupará muito menos. Como a declaração de estilos se armazena no cachê do navegador, só se transfere na primeira página que se visita do site, com o qual a segunda e as posteriores páginas que se solicitem se carregarão muito mais rápido.
- **Facilidade para alterar o aspecto da página sem tocar no código HTML**. Como toda a informação dos estilos e o posicionamento das camadas se encontram em um mesmo arquivo, se desejarmos mudar qualquer elemento da página -seja sua posição ou seu aspecto-, só temos que atualizar a folha de estilos, e as mudanças serão vistas automaticamente em todo web.

Como dissemos, também existem algumas desvantagens:

- **Compatibilidade com navegadores antigos**. Necessita-se que o visitante disponha de um navegador bastante avançado. A maioria dos visitantes dispõe de navegadores que suportam características avançadas das CSS, mas ainda há muita gente que não atualizaram suas máquinas ou que navega em sistemas de somente texto. Os navegadores que não suportam folhas de estilos, pelo menos lerão o código da página e o mostrarão sem nenhum posicionamento. Isto pode ser fatigante, mas pelo menos visualizarão todos os dados da página, embora desarrumados e sem estilo.
- **Diferenças entre navegadores**. Dependendo do navegador também mudam as etiquetas de estilos suportadas, podendo fazer com que as páginas não sejam vistas exatamente igual entre um cliente e outro. Também, assim como ocorre com HTML, existem atributos não padronizados ou que têm valores padronizados diferentes. Quando se inicia com a construção em CSS, pode ser um tema bastante complicado e pode dar bastante dor de cabeça, porém, trata-se de, pouco a pouco, ir aprendendo

todos os atributos e os navegadores onde se visualizam ou não.
- **Dificuldade**. Sem dúvida, se estamos acostumados ao HTML, passar a CSS é bastante mais complicado e requer um estudo mais profundo. Entretanto, este passo nos brindará um maior controle dos elementos da página e ampliará nossas fronteiras na hora de construir.

Exemplo CssZenGarden

Existe um site muito interessante e ilustrador que pode nos ajudar a conhecer rapidamente a potência das CSS e termos uma idéia do que pode significar seu uso. É um site onde se mostra um conteúdo e um desenho bastante logrado. Ademais, dispõe de vários links para poder ver o mesmo site, como mesmo conteúdo, mais com distinto aspecto. Dessa forma podemos ver como se pode chegar a alterar o desenho de uma página com somente mudar a folha de estilos.

A URL do site é http://www.csszengarden.com. É muito interessante que selecionem outros desenhos para ver a mudança radical que as páginas web podem ter com distintas folhas de estilos.

Nós exploramos um pouco as capacidades das CSS e realizamos um exemplo de desenho de CssZenGarden por nossa conta. Podemos vê-lo neste link.

Por onde continuar

Em CriarWeb.com pensamos em fazer uma série de artigos para explicar o processo de criação de uma web, projetada completamente em CSS. Entretanto, por enquanto, devemos fazer referência a algumas páginas em inglês que explicam este assunto detalhadamente.

http://www.stylegala.com/
http://www.cssbeauty.com/
http://www.cssvault.com/

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Formas de aplicar estilos em construção CSS

Vamos ver outra vez distintos modos de aplicar estilos às páginas. É um tema que já vimos no manual de CSS, mas vale a pena refrescar conceitos e ampliar a informação que foi oferecida em seu dia.

Aplicação de estilo a etiquetas

Pode-se atribuir o estilo a uma etiqueta concreta de HTML. Para isso, na declaração de estilos escrevemos a etiqueta e entre chaves, os atributos de estilo que desejarmos.

```
body {
   background-color: #f0f0f0;
   color: #333366;
}
```

Podemos aplicar o mesmo estilo em um conjunto de etiquetas. Para isso, indicamos as

etiquetas seguidas por vírgulas e logo, entre chaves, os atributos que quisermos definir.

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

Neste caso se define que os cabeçalhos de nível 1 e os parágrafos, tenham a letra vermelha.

Definição de classes

Podemos utilizar uma classe se desejarmos criar um estilo específico, para logo aplica-lo a distintos elementos da página. As classes na declaração de estilos se declaram com um ponto antes do nome da classe.

```
.minhaclasse{
   color: blue;
}
```

Para atribuir o estilo definido por uma classe em um elemento HTML, simplesmente se acrescenta o atributo class à etiqueta que quisermos aplicar tal classe. O atributo class se atribui ao nome da classe a aplicar. Por exemplo:

```
<p class="minhaclasse">este parágrafo tem o estilo definido na classe "minhaclasse".</p>
```

O parágrafo anterior se apresentaria com a cor azul. A definição de classes e sua utilização é simples, vejamos um exemplo mais detalhado:

Para a seguinte declaração de estilos:

```
body, td, p{
   background-color: #000000;
   color: #ffffff;
}

.inverso{
   background-color: #ffffff;
   color: #000000;
}
```

Foi definido um fundo preto e cor de texto branco para o corpo da página, assim como as células e os parágrafos. Logo, foi declarado uma classe, de nome "inverso", com as cores ao contrário, ou seja, com fundo branco e texto negro.

```
<body>
<p>Olá, isto é um parágrafo normal</p>
<p class="inverso">Parágrafo com as cores invertidas</p>
<table>
<tr>
   <td class="inverso">INVERSO</td>
   <td>NORMAL</td>
</tr>
</table>
</body>
```

Esta página tem, geralmente, o fundo negro e o texto branco. O primeiro parágrafo, que é u parágrafo normal, continua com esta definição geral de estilos, mas o segundo parágrafo, cujo foi aplicado a classe "inverso", tem o fundo branco e o texto em preto. Em relação à tabela,

em sua primeira célula foi atribuída a classe "inverso", portanto será visto com fundo branco e cor de texto em preto. Enquanto que a segunda célula, que não tem atribuído nenhuma classe, se apresentará como foi definido na regra geral.

Para conhecer os resultados obtidos no exemplo anterior podemos vê-lo em uma página a parte.

Estilos que só se utilizam uma vez

Também podemos ter um estilo específico para um único elemento, que não vai se repetir em nenhum outro caso. Para isso temos os estilos atribuídos por identificador. Os identificadores se definem em HTML utilizando o atributo id na etiqueta que desejarmos identificar. O valor do atributo id será o que nós definirmos.

```
<div id="camada1">
```

Na folha de estilos, para definir o aspecto desse elemento com id único, escreve-se o caractere "jogo da velha", seguido do identificador indicado na etiqueta e entre chaves os atributos css que desejarmos.

```
#camada1{
   font-size: 12pt;
   font-family: arial;
}
```

Neste caso, foi atribuído fonte de tamanho 12 pontos e corpo arial.

Como se pode concluir na leitura destas linhas, geralmente é preferível utilizar estilos definidos em classes do que aos definidos com identificadores, a não ser que estivermos certos de que esse estilo não vai se repetir em todo o documento.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Pseudo-element em CSS

Os pseudo-elements (pseudo-elementos, se traduzimos em português) servem para aplicar estilos a partes mais específicas dentro de uma etiqueta. Ou seja, para o exemplo concreto da etiqueta parágrafo, com os pseudo-elementos podemos definir o estilo para a primeira letra do parágrafo e para a primeira linha.

Pseudo-element first-letter

Um efeito habitual de algumas publicações, por exemplo, as de estórias para crianças, é fazer maior a primeira letra de una página e enfeita-la de alguma maneira. Com CSS podemos aplicar estilos específicos e fazer, por exemplo, que essa primeira letra seja maior e tenha uma cor diferente do resto do parágrafo.

Utiliza-se desta maneira:

```
P:first-letter {
font-size: 200%;
```

```
color: #993333;
font-weight: bold;
}
```

Assim, estamos atribuindo um tamanho de letra 200% maior do próprio parágrafo. Também estamos mudando a cor dessa primeira letra.

Dentre todas as propriedades de estilos, somente algumas podem se aplicar aos pseudo-elementos first-letter. São as seguintes, segundo a especificação do W3C: font properties, color properties, background properties,'text-decoration', 'vertical-align' (somente se 'float' estiver atribuído a 'none'), 'text-transform', 'line-height', margin properties, padding properties, border properties, 'float', 'text-shadow' e 'clear'.

Pode-se ver um exemplo de aplicação de um estilo com first-letter.

Pseudo-element first-line

Como adiantava, este pseudo-elemento, serve para atribuir um estilo próprio à primeira linha do texto. É possível que tenhamos visto alguma revista ou jornal que utilize este estilo para remarcar as primeiras linhas do parágrafo. Um exemplo de seu uso seria o seguinte:

```
P:first-line {
text-decoration: underline;
font-weight: bold;
}
```

As propriedades de estilos que podem se aplicar ao pseudo-element first-line são as seguintes: font properties, color properties, background properties, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'text-shadow' e 'clear'.

Pode-se ver um exemplo de aplicação de um estilo com first-line.

Uso de classes com os pseudo-elementos

Em determinadas ocasiões podemos necessitar criar uma classe (class) de CSS a qual atribuir os pseudo-elementos, de modo que estes não se apliquem em todas as etiquetas da página. Por exemplo, podemos desejar que somente se modifique o estilo da primeira linha do texto em alguns parágrafos e não em todos da página.

Uma classe se define com o nome da etiqueta seguido de um ponto e o nome da classe. Se ademais, desejarmos definir um pseudo-elemento, deveríamos indicá-lo a seguir, com esta sintaxe:

```
P.nomeclasse:first-line {
font-weight: bold;
}
```

Pseudo-elementos em CSS2

A parte de first-line e first-letter, nas especificações de CSS 2 existem outros pseudo elementos que vou nomear para conhecimento dos leitores, embora nos aprofundaremos em seu uso. Tratam-se de before e after e servem para inserir "conteúdos gerados" antes e depois

do elemento ao qual atribuirmos estes pseudo-elements.

Um exemplo disso é:

```
P.nota:before {
content: "Nota: "
}
```

Assim, se definiu uma classe de parágrafo chamada "note" na qual se indica que antes da própria nota se deve incluir o texto indicado, ou seja, "Nota: ".

Atenção à compatibilidade com CSS2, que, pelo menos para estes elementos, não está suportada em versões 6 de Internet Explorer. Firefox, por sua vez, sim que é compatível com estas características de CSS2.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Definição de estilos CSS Shorthand

Shorthand:

Vamos explicar como escrever de forma reduzida nossas regras CSS para que nossos arquivos de estilo tenham menos peso e sejam mais entendíveis na hora de uma atualização.

Segunda a W3C há duas formas de escrever a mesma regra de CSS: a padrão e a shorthand. Uma é a longa e a outra é a reduzida.

Propriedade Font (fonte)

font-style || font-variant || font-weight || font-size / line-height || familia da fonte

Exemplo:

```
P {font: italic normal bold 12px/14pt Verdana, Tahoma, Arial}
```

Propriedade Background (fundo)

background-color || background-image || background-repeat || background-attachment || background-position

Exemplo:

```
Body {background: #FFF url(../imagens/exemplo.gif) no-repeat fixed center}
```

Margin (Margem)

longitude | porcentagem | auto

Exemplo:

```
Body {margin: 5px} /* todas as margens a 5px */ P {margin: 2px 4px} /* margens superior e inferior a 2px, margens esquerda e direita a 4px */ DIV {margin: 1px 2px 3px 4px} /* margem superior a 1px, right margin a 2px,
```

bottom margin a 3px, left margin a 4px */

Padding (Enchimento)

longitude | porcentagem | auto

Exemplo:

Body {padding: 2em 3em 4em 5em} /* Se definimos quatro valores estamos aplicando o padding superior, direito, inferior e esquerdo */ Body {padding: 2em 4em) /* Se definimos dois ou três valores, os valores faltantes se tomam do lado oposto: superior e inferior a 2em, direito e esquerdo a 4em */ Body {padding: 5em} /* Se definimos um só valor se aplicam a todos os lados */

Border (Borda)

border-width || border-style || color

Exemplo:

H3 {border: thick dotted blue}

*Artigo por **Federico Elgarte***

Utilizar porcentagens para tamanhos de texto com CSS

A porcentagem é outra das medidas ou unidades que podemos utilizar nos atributos de folhas de estilo em cascada (CSS) para definir um tamanho. Neste artigo, veremos exemplos sobre como modificar os tamanhos dos textos por meio de porcentagens, para conseguir novos tamanhos que sejam relativos aos que se estão utilizando.

Por exemplo, poderíamos definir um estilo para escrever com um texto com o dobro de tamanho do que se estiver trabalhando:

```
<span style="font-size:200%">Este texto tem o dobro de tamanho</span>
```

Isto quer dizer que o texto terá o dobro e tamanho, 2 pelas unidades de texto que estivermos trabalhando. Por exemplo, se estivermos trabalhando com tamanhos de texto de 10pt, o texto dentro do anterior span seria 20pt. O do seguinte código exemplifica este caso concreto:

```
<span style="font-size:10pt;">Ola amigos <span style="font-size:200%">Este texto tem o dobro de tamanho</span> </span>
```

O mesmo pode ser feito, porém para definir um texto menor, atribuindo porcentagens abaixo

de 100%. Por exemplo, se quiséssemos fazer um texto com a metade do tamanho utilizaríamos a seguinte etiqueta:

```
<span style="font-size:50%">Este texto é a metade do anterior<,/span>
```

Se estivéssemos trabalhando com um tamanho de texto de 16pt, com a anterior etiqueta se escreveria com tamanho 8pt. O código seria o seguinte:

```
<span style="font-size:16pt;">Ola amigos
<span style="font-size:50%">Este texto é a metade do anterior</span>
</span>
```

Agora vamos definir um par de classes para um texto maior e menor, que poderíamos utilizar para aumentar e reduzir o texto respectivamente.

```
<style type="text/css">
  .mayor {font-size:150%}
  .menor {font-size:75%}
</style>
```

Este código indica que a classe maior é um texto de 150%, ou seja, a metade mais que o anterior, e a classe menor um texto de 75%, ou seja, três quartas partes do anterior. Poderíamos utilizar estas classes com um código como este:

Este é um texto normal e este é maior, este volta a ser normal e este é menor

Os diferentes exemplos deste artigo podem ser vistos em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Por que desenhar com CSS

As tabelas existem e existiram desde o começo em HTML, porém não se criaram para desenhar um site, e sim para a apresentação de dados tabulares. A utilização do "border=0" e as imagens transparentes tornaram possível criar uma rede que permitiu aos designers organizar textos e imagens, estabelecer tamanhos e localizar objetos. Porém, isto é simplesmente incorreto. As tabelas não se criaram para construir e não devem ser utilizadas para isso, porque desta forma se misturam apresentação e conteúdo.

A solução é clara: CSS+HTML/XHTML. Felizmente, cada vez são mais as empresas que decidem deixar atrás as tediosas tabelas e evoluir desenvolvendo seus sites respeitando os padrões estabelecidos pela W3C (organização internacional que há uns 12 anos se dedica ao desenvolvimento de pautas e padrões web), o que facilita a acessibilidade e a correta visualização das páginas nos navegadores que respeitem tais padrões.

Algumas das vantagens da construção com CSS:

- Separação de forma e conteúdo. Geralmente CSS e HTML se encontram em arquivos separados, o que facilita o trabalho em equipe porque designer e programador podem trabalhar independentemente. Por outro lado, permite o acesso a distintos navegadores e dispositivos.

- Tráfego no servidor. As páginas podem reduzir seu tamanho entre um 40% e um 60%, e os navegadores salvam a folha de estilos no cache, isto reduz os custos de envio de informação.
- Tempos de carregamento. Pela grande redução no peso das páginas, melhora a experiência do usuário, que valoriza de um site o menor tempo no download.
- Precisão. A utilização de CSS permite um controle muito maior sobre o desenho, especificando exatamente a localização e tamanho dos elementos na página. Também se podem empregar medidas relativas ou variáveis para que a tela se acomode ao seu conteúdo.
- Manutenção. Reduz notavelmente o tempo de manutenção quando é necessário introduzir uma mudança porque se modifica um só arquivo, o da apresentação, sem ter que tocar as páginas que contém a estrutura com o conteúdo.
- Desenho unificado e flexibilidade. É possível mudar completa ou parcialmente o aspecto de um site simplesmente com modificar a folha de estilos. Por outro lado, o ter o estilo de uma web em um só arquivo permite manter a mesma aparência em todas as páginas.
- Posicionamento. As páginas desenhadas com CSS têm um código mais limpo porque não levam desenho, só conteúdo. Isto é semanticamente mais correto e a página aparecerá melhor posicionada nos buscadores. Google navega se esquivando do desenho.

Recomendo um site simpático e didático sobre o tema:
http://www.effectivetranslations.com/stupidtables/everything_es.html

*Artigo por **Serviweb***

Truques CSS para não enlouquecer

Não tire ainda o monitor contra a parede! Aqui vão os principais truques CSS para fazer frente aos típicos problemas que os web designers enfrentam quando constroem com CSS. Poderão existir discrepâncias entre os leitores, porém esclareço que estas são técnicas que a mim pessoalmente me deram resultado, depois de muitas provas e tentativas aprendi isto...

Use um contêiner global para todas as caixas (quando as coisas se disparam)
Desta forma estará pré-fixando globalmente a ordem de todas as demais caixas. Em referência a este contêiner ordena o resto das coisas interiores. É como se fizesse uma cerca para que nada se escape. Obviamente estamos falando de sites "fixos" não elásticos.
À vezes é bom usar um contêiner até o corpo do site, logo deixar o rodapé fora.

Exemplo para um contêiner de 900px centrado:

```
#contêiner {
margin-top: 0px;
margin-right: auto;
margin-bottom: 0px;
margin-left: auto;
width: 900px;
}
```

Que flutue à esquerda (quando as caixas se sobrepõem)
Esta é uma ótima forma de evitar incompatibilidades entre navegadores. O uso de hacks de

CSS se devia em grande parte porque se trabalhava "centralizando" as caixas. Se por exemplo, precisar colocar três caixas de 300px em um contêiner de 900px você poderá fazer o seguinte:

Exemplo:
```
#caixa {
float: left;
width: 300px;
}
```

Calcular bem os paddings ou preenchimentos (quando as caixas forem abaixo)
Quase todas as dores de cabeça com o CSS se devem ao mal uso ou à má interpretação que se faz do padding. Porém, é mais simples do que parece.
Para que servem os paddings ou preenchimentos? Bom, o que faz é gerar um "preenchimento" de determinada medida para dar por exemplo, como uma margem aos elementos, porém o faz sobre a largura em píxels que esteja pré-fixada. Por exemplo: se tivermos uma caixa de 300px e lhe aplicarmos um padding de 10px na esquerda, agora teremos uma caixa de 310px. Isto fará transbordar o resto das caixas e as deslocarão para baixo. Aí é quando o designer principiante fica louco. O tema é que se houver uma diferença de até um 1px se produzirão estes transbordamentos, senão observe quando lhe incluir bordas em sua caixa se se produzirão diferenças.
O que se deve fazer é simples, calcular bem e lembrar cada ajuste que se faça dos preenchimentos. Agora teremos que fazer uma caixa de 290px com paddings de 10px à esquerda.

Exemplo:
```
#caixa {
float: left;
width: 290px;
padding-left: 10px;
background-color: #FFE6DD;
}
```

O rodapé da página com largura fixa (quando o rodapé da página enlouquece):
Para entender melhor como funciona o uso de caixas em CSS se pode pensar em um grupo de objetos de diferentes formas que lutam por se adaptar e ocupar o espaço que se pré-fixou. Acontece muitas vezes que os rodapés de página são os que mais problemas trazem quando se constrói um site. Ou vai para cima, ou se alinha à esquerda, ou transborda, etc. Muitos de nós resolvíamos este tema pré-fixando valores fixos às alturas de caixas, porém não faz sentido. O que se deve fazer é estabelecer de novo um valor de largura fixo. Desta forma o rodapé irá definir seu lugar do resto e irá parar aonde tem que ir.

Exemplo:

```
#rodape {
width: 900px;
background-color: #666666;
}
```

Nem tudo é 1+1=2 em CSS (quando as larguras não fecham)
Um problema comum em css é pensar que todas as larguras entre caixas fecham perfeitamente. Às vezes é necessário jogar com os valores dos contêiners, às vezes contrário à

lógica há que adicionar alguns px aos contêiners.

OUTROS TRUQUES RÁPIDOS:

Truques simples, dos que não fazem falta explicar muito, mas que são muito práticos e tornará o trabalho mais fácil e evitarão possíveis erros.

- Use cores diferentes para distinguir as caixas
- Coloque uma palavra descritiva em cada caixa
- Comente o código fonte e assinale os finais dos contêiners grandes
- Não seja mesquinho com os espaços entre os divs
- Não seja um fundamentalista e não queira escrever seu CSS com dois ou três linhas. Se não quiser erros escreva o necessário.
- Cuidado com o tamanho das imagens que você insere, estas mudam a largura dos contêiners.
- Escolha bem os nome de cada div e trate de ser ordenado no código.
- Se for trabalhar com várias caixas, trate de agrupá-las, isto é muito importante. Por exemplo, um contêiner que agrupe três ou quatro caixas.

CONCLUSÃO:

Todos estes parágrafos são simplesmente algumas sugestões ou comentários do que me deu dado um bom resultado. Existem outras muitas "ligações" deste tipo, se tiver alguma dúvida não duvide em comentá-las neste mesmo artigo.

O que acontece quando não puder resolver um problema com CSS ou similar? Para mim o melhor é me levantar e espairecer por um momento, fazer qualquer outra coisa e logo voltar e tentar resolver de novo.

Deixar de renegar e não enlouquecer com CSS dependerá da quantidade de tempo, trabalho e esforço que coloque em seu trabalho. Não duvide!

*Artigo por **Leonardo A. Correa***

10 erros comuns nos CSS

1. Uso desnecessário do valor 0

O código seguinte não necessita a unidade especificada se o valor for zero.

```
padding:0px 0px 5px 0px;
```

Em seu lugar pode ser escrito desta maneira:

```
padding:0 0 5px 0;
```

Da mesma maneira é igual para outros estilos. Ex.:

```
margin:0;
```

Não desperdice espaços agregando unidades tais como px, pt, em, etc, quando o valor for zero. A única razão de fazer isto é se necessitar mudar estes valores mais tarde. Se não declarar estas unidades não tem sentido. Os pixels zero são iguais que os pontos zero.

Entretanto, line-height pode não ter unidade. Por isso, é válido o seguinte:

```
line-height:1;
```

De qualquer maneira você pode utilizar uma unidade em concreto como em se desejar.

2. As cores em formato hexadecimal necessitam uma almofadinha

Isto está mal:

```
color: ea6bc2;
```

Deve ser:

```
color: #ea6bc2;
```

Ou este outro:

```
color: rgb (234.107.194);
```

3. Valores duplicados nos códigos de cores

Não escrever o código desta maneira:

```
color: #ffffff;
background-color:#000000;
border:1px solid #ee66aa;
```

Os valores duplicados podem ser omitidos escrevendo os códigos desta maneira:

```
color:#fff;
background-color:#000;
border:1px solid #e6a;
```

Obviamente isto não deve ser feito com códigos como este!

```
color: #fe69b2;
```

4. Evitar repetições de código desnecessária

Evite usar várias linhas quando se pode conseguir com uma só. Por exemplo, ao fixar as bordas, algumas vezes se deve fazer separadamente, porém em casos como o seguinte não é necessário:

```
border-top:1px solid #00f;
border-right:1px solid #00f;
border-bottom:1px solid #00f;
border-left:1px solid #00f;
```

Poderíamos resumi-lo em uma única linha como esta:

```
border:1px solid #00f;
```

5. A duplicação é necessária com os estilos em cascata

Nos estilos em cascata é aceitável repetir o mesmo código para um elemento duas vezes, se significa evitar a repetição mencionada no ponto acima. Por exemplo, digamos que temos um elemento onde somente é diferente a "border" esquerda. Ao invés de colocar cada "border" escrita usando quatro linhas, uso só duas:

```
border:1px solid #00f;
border-left:1px solid #f00;
```

Neste caso, primeiro definimos todas as "borders" com a mesma cor, porém mais tarde para economizarmos duas linhas de código redefinimos a "border" esquerda a outra cor, desta maneira economizamos duas linhas de código.
O exemplo desperdiçando espaço ficaria assim:

```
border-top:1px solid #00f;
border-right:1px solid #00f;
border-bottom:1px solid #00f;
border-left:1px solid #f00;
```

É claro que esta economia de carregamento supõe um atraso no carregamento da página, pois estamos definindo a "border" esquerda duas vezes, porém o carregamento deste processo é insignificante.

6. Os estilos inválidos não fazem nada

Um exemplo é suficiente para explicar este erro:

```
padding:auto
```

Este estilo só pode ser aplicado a width e height, porém não a padding.

7. Código Específico para cada navegador

Logicamente este tipo de código só funcionará no navegador ao que vai destinado, porém há que pensar se é rentável visto que só alguns usuários poderão apreciar essas mudanças.

8. Espaço perdido

Não estou seguro do porquê, mas muitos designers estão empenhados em desperdiçar o espaço em seu código, usando um montão de quebras de linha desnecessárias. Lembre que isso só será visto por você e estará fazendo um uso excessivo de largura de banda. Também seu código será mais fácil de ler visto que terá menos "brechas".

Por suposto que é sábio deixar um certo espaço para mantê-lo legível, embora haja pessoas que gostem de condensar todo, não deixando nenhum espaço.

9. Especificar as cores sem usar palavras

Definir as cores usando as palavras que as definem não é uma boa idéia visto que estaríamos confiando no navegador para que interprete que cor e código devem aplicar. As tonalidades para um mesmo nome de cor mudam muito de um navegador a outro.

É uma boa prática especificar sempre a cor por seu código hexadecimal.
Ex.: utilizar "#fff" ao invés de branco.

10. Agrupar estilos idênticos

É comum ver os estilos escritos às vezes com o mesmo código, mesmo quando o estilo é igual.

Seria conveniente agrupá-los e assim otimizaríamos espaço:

```
h1, p, #footer, .intro {
font-family:Arial,Helvetica,sans-serif;
}
```

Também nos fará muito mais fácil a tarefa de atualizar o código.

*Artigo por **Manu Gutierrez***

Introdução a CSS3

Desde que CSS começou passaram muitos anos e já vamos pela especificação de CSS3, que incorpora uma série de novidades que vamos tratar de resumir neste artigo.

O que é CSS

Se você não sabe o que é CSS provavelmente lhe interessaria começar lendo nosso manual de CSS ou a seção de CSS a fundo. Não obstante, caberia dizer que CSS é uma linguagem para definir o estilo ou a aparência das páginas web, escritas com HTML ou dos documentos XML. CSS se criou para separar o conteúdo da forma, ao mesmo tempo que permite aos designers manter um controle muito mais preciso sobre a aparência das páginas.

Com CSS3, mais controle sobre a forma

O objetivo inicial de CSS, separar o conteúdo da forma, se cumpriu já com as primeiras especificações da linguagem. Entretanto, o objetivo de oferecer um controle total aos designers sobre os elementos da página foi mais difícil de cobrir. As especificações anteriores da linguagens tinham muitas utilidades para aplicar estilos às webs, porém os desenvolvedores ainda continuam usando truques diversos para conseguir efeitos tão comuns ou tão desejados como as bordas arredondadas ou a sombra de elementos na página.

CSS 1 já significou um avance considerável na hora de desenhar páginas web, trazendo muito maior controle dos elementos da página. Porém, como ainda ficaram muitas coisas que os designers desejavam fazer, mas que CSS não permitia especificar, estes deviam fazer uso de truques para o design. O pior destes truques é que muitas vezes implica alterar o conteúdo da página para incorporar novas etiquetas HTML que permitam aplicar estilos de uma maneira mais elaborada. Dada à necessidade de mudar o conteúdo, para alterar o desenho e fazer coisas que CSS não permitia, estava-se acabando com algum dos objetivos para os que CSS foi criado, que era o de separar por completo o conteúdo da forma.

CSS 2 incorporou algumas novidades interessantes, que hoje já utilizamos habitualmente, porém CSS 3 ainda avança um pouco mais na direção, de dar mais controle sobre os elementos da página.

Sendo assim, a novidade mais importante que traz CSS 3, para os desenvolvedores de webs, consiste na incorporação de novos mecanismos para manter um maior controle sobre o estilo com o qual se mostram os elementos das páginas, sem ter que recorrer a truques ou hacks, que a muitas vezes complicavam o código das webs.

Propriedades novas em CSS3

Segue aqui uma lista das principais propriedades que são novidade em CSS3.

Bordas

- border-color
- border-image
- border-radius
- box-shadow

Fundos

- background-origin
- background-clip
- background-size
- fazer camadas com múltiplas imagens de fundo

Cor

- cores HSL
- cores HSLA
- cores RGBA
- Opacidade

Texto

- text-shadow
- text-overflow
- Ruptura de palavras longas

Interface

- box-sizing
- resize

- outline
- nav-top, nav-right, nav-bottom, nav-left

Seletores

- Seletores por atributos

Modelo de caixa básico

- overflow-x, overflow-y

Outros

- media queries
- criação de múltiplas colunas de texto
- propriedades orientadas a discurso ou leitura automática de páginas web
- Web Fonts

Esta lista de novas propriedades de CSS3 foi tirado de: http://www.css3.info/preview/. É um site em inglês, mas pode ser bom visitar para ir conhecendo mais detalhes sobre CSS3.

Em futuros artigos daremos algumas chaves e explicações sobre várias destas propriedades, pelo menos as mais interessantes, com especificações detalhadas, assim como exemplos que sirvam para ir conhecendo esta nova especificação de CSS.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Bordas arredondadas em CSS 3

CSS 3 incorpora novas propriedades para o controle de bordas dos elementos. Agora se permitem bordas com as esquinas arredondadas, bordas com imagens (inclusive pode-se utilizar várias imagens para definir o aspecto da borda, sombras, etc.

Neste artigo vamos explicar como realizar bordas arredondadas com CSS3.

Temos a propriedade border-radius, que permite definir bordas arredondadas nas esquinas, especificando as medidas do radio que devem dar à curva das esquinas

```
border-radius: 5px;
```

Definiria um radio de 5 pixels no arredondamento das esquinas do elemento. Até agora Mozilla adotou este atributo com um nome especial, que é válido para produtos como Firefox, enquanto que as especificações de CSS3 não tiverem alcançado o estado "Candidate Recommendation", que é quando se supõe que os diferentes navegadores devem implementá-las. O nome do atributo por enquanto é:

```
-moz-border-radius
```

Quanto à Internet Explorer há que dizer que ainda não suporta este atributo de CSS 3, porém esperemos que o pessoal de Microsoft possa implementar em breve no navegador, ou que as especificações de CSS3 passem rápido ao estado "Candidate Recommendation", que seria a chamada para que se implementem na generalidade dos navegadores.

Por enquanto, para navegadores Mozilla, o atributo border-radius se utilizaria, por exemplo, assim:

```
DIV {
border: 1px solid #000000;
-moz-border-radius: 7x;
padding: 10px;
}
```

Com isto conseguimos que todos os div tenham uma borda arredondada nas esquinas de radio de 7pixels.

Podemos ver o exemplo em uma página a parte. Porém, há que lembrar que só se verá o efeito utilizando Firefox ou navegadores baseados em Mozilla.

Contudo, o atributo border-radius tem outras possíveis configurações, na qual se podem definir os valores para o radio das quatro esquinas separadamente. Desta maneira:

```
-moz-border-radius: 7px 27px 100px 0px;
```

Assim estaríamos definindo uma borda arredondada com radio de 7 pixel para a esquina superior esquerda, logo 27px para a esquina superior direita, de 100px para a inferior direita e 0px para a inferior esquerda. (Há que explicar que um border-radius de 0px é uma borda com esquina em ângulo reto)

Podemos ver este ejemplo em uma página a parte.

As bordas arredondadas com CSS 3, como se poderá imaginar, só são vistas se tiver sido definida alguma borda visível ao elemento que as atribuímos, seja solid, dotted, etc. Isso é o que define as especificações de CSS3, embora no momento de escrever o artigo devo assinalar que inclusive Mozilla ou Firefox (o único que até agora suporta este atributo de CSS3) não funciona inteiramente corretamente com isto e só mostra as bordas arredondadas com solid.

Outra coisa que definem as especificações de CSS e que por enquanto não está funcionando inteiramente bem, é que as imagens de fundo devem se ajustar às bordas arredondadas. Ocorre, até este momento, que as imagens de fundo saem por fora das bordas arredondadas. Confiamos em que no futuro isto possa ser revisado e otimizado, para que tudo funcione corretamente. Devemos lembrar que no momento de escrever o artigo ainda se têm que terminar de definir as especificações de CSS 3 e depois, os navegadores web deveram se atualizar em um certo espaço de tempo para suportá-las completamente.

Nota: Oferecemos uma lista das características principais de CSS 3 no artigo Introdução a CSS3.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Listagem de distintos frameworks CSS

Neste artigo vamos enumerar e comentar algumas coisas sobre frameworks CSS, se é que se pode chamar de frameworks, visto que esse conceito se usa muitas vezes para sistemas que facilitam a programação de aplicações e no caso dos CSS, não é programação, e sim que se utilizam para obter ajudas na construção de webs.

Como sabemos, no momento atual as páginas se constroem com CSS. Com HTML especificamos os conteúdos e com CSS a forma ou disposição com a qual deve se apresentar ao usuário nos navegadores. CSS, portanto, é uma linguagem que serve para especificar o estilo das páginas, porém muitas vezes fazemos coisas repetitivas, como divisões de página em colunas, caixas de determinados tipos, etc. Pois os Frameworks CSS nos ajudam a realizar essas tarefas de construção básicas, que muitas vezes temos que implementar repetidas vezes em diversos lugares, para gerar as estruturas de elementos da página.

Os frameworks CSS dispõe de uma série de classes (de folhas de estilo) já criadas com as quais ajuda a posicionar elementos na página e criar estruturas de construção, mais ou menos versáteis. Sendo assim, no desenvolvimento de páginas novas, ou no redesenho de páginas antigas, os frameworks CSS podem nos ajudar para dispor de uma rede onde posicionar os distintos componentes de nosso desenho. Com isso economizaremos o tempo de ter que criar de novo dezenas de classe que estamos entediados de implementar para criar construções a 2, 3 ou 4 colunas, com divisões de cabeçalho, corpo e rodapé, etc.

Neste artigo pretendo simplesmente enumerar este e outros frameworks CSS, em uma lista que pretende dar uma idéia das possibilidades que nos oferece neste momento o mercado.

960 Grid System

É, talvez, o mais utilizado dos frameworks CSS, cujas páginas se constroem em larguras de 960 pixels (daí vem seu nome). Oferece duas possibilidades de construção de páginas, com uma rede de 12 ou 16 colunas. Nós escolhemos este framework CSS para explicá-lo aos nossos leitores de CriarWeb.com, justamente por ser tão popular. Em nosso trabalho com este sistema podemos comprovar que é muito versátil e sobretudo, simples de utilizar.

Página web de 960 Grid System

Simple

Este framework CSS é apresentado por um desenvolvedor chileno, com o qual todos os exemplos e documentação que puder encontrar está em espanhol. O criador o realizou para poder simplificar as coisas, não só no desenvolvimento das páginas, como também na aprendizagem.

Página web de Simple

Blueprint

É um framework CSS que pretende reduzir o tempo de desenvolvimento das páginas web. Oferece uma estrutura sólida na que fundar seu trabalho de desenho, por meio da típica rede. Porém, não se limita simplesmente a isso, e sim que oferece outra série de classes muito úteis para estilizar componentes típicos, como formulários, botões, abas, tipografias ou para que

suas páginas web se possam imprimir de maneira ótima em papel.

Página web de Blueprint

YUI Grids CSS

O framework CSS de Yahoo! É um código CSS que permite construir páginas com distintas larguras (750px, 950px, e 974px) e fazer todas as coisas típicas que se podem desejar em uma página. Tem 6 planilhas pré-definidas e a possibilidade de criar mais de 1.000 combinações de construção, em regiões de 2, 3 e 4 colunas. Faz parte da Yahoo! User Interface Library, o que dá uma garantia adicional, por vir de tão renomado buscador.

Página web de YUI Grid CSS

Tripoli

Tripoli não é um framework CSS e segundo remarcam os criadores, isso significa que não te obriga a desenvolver sua página de uma maneira determinada. Em contra, o que oferece é um código CSS que "resetea" os estilos pré-definidos dos navegadores e os redefine, conseguindo uma base estável sobre a que realizar um site e que se veja igual em qualquer cliente web. Pode ser interessante porque tenta não cair nos problemas que têm os frameworks CSS.

Página web de Tripoli

Boilerplate

Este framework me pareceu interessante porque está criado por um dos desenvolvedores iniciais de Blueprint, que fundou o novo projeto para colocar em funcionamento suas idéias particulares. Como ele diz, este framework está pensado para simplificar as coisas e ser ligeiro, oferecendo todo o básico para poder construir uma web, porém sem as complexidades que tem Blueprint e com convenciones de nomes que dão mais sentido e significado ao que estamos codificando.

Pagina web de Boilerplate

BlueTrip

Segundo seus criadores, BlueTrip é uma combinação do melhor de distintos frameworks CSS dos que já falamos. Seu nome vem da união de BLUEprint - TRIpoli, fazendo referência a essa união dos melhores framewoks, entre os que também se inspiraram em nosso framework preferido nestes momentos, 960 grid, por sua simplicidade.

Página web de BlueTrip

Outros Framework CSS

Coloco outros frameworks CSS que encontrei e que não investiguei tanto as possibilidades que oferecem, embora também possam ser interessantes, sobretudo pode que dêem enfoques diferentes que possam ser úteis em certas ocasiões.

Elements
ESWAT
Content with style
My CSS Framework
Hartija
Malo (Framework CSS ultra pequeno)
emastic

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Fluxo HTML e atributos CSS

Vale a pena determos para explicar mais detalhadamente o que é o fluxo HTML, pois é um conceito simples e básico para poder entender muitos assuntos sobre o posicionamento web e mais concretamente o posicionamento com CSS.

O fluxo da página é algo assim como o fluxo de escritura de elementos dentro da tela que nos apresenta o navegador. Sabemos que as páginas web são codificadas em HTML e os elementos aparecem no código em uma posição dada. O navegador, no momento que interpreta o código HTML da página, vai colocando na página os elementos (definidos por meio de etiquetas HTML) segundo vai encontrando-os no mesmo código.

Por exemplo, pensemos em uma página que tem um titular com H1, logo vários parágrafos e alguma imagem. Pois se o primeiro que aparece no código HTML for o cabeçalho H1, pois esse cabeçalho aparecerá na página também em primeiro lugar. Logo, se colocarão os parágrafos e se a imagem aparecia no código por último, na página também aparecerá ao final. Ou seja, os elementos aparecem colocados tal como estiverem ordenados no código. Isto é chamado de fluxo HTML, a colocação dos elementos no lugar que corresponda segundo seu aparecimento no código.

Isto em geral ocorre com qualquer dos elementos da página. Entretanto, há alguns atributos HTML que podem marcar distintas propriedades no fluxo, como que uma imagem se alinhe à direita, com align="right", com o texto do parágrafo que possa ter a seguir rodeando a imagem. Porém, com HTML, se por exemplo, uma imagem vai antes que um parágrafo, nunca vamos poder trocar suas posições e colocar a imagem detrás do parágrafo que lhe segue no código.

Isto não ocorre de igual maneira quando trabalhamos com CSS, visto que existem diversos atributos que podem mudar radicalmente a forma na que se mostram na página, por exemplo, o atributo position que pode definir valores como absolute, que rompe o fluxo da página, o melhor dito, tira do fluxo da página ao elemento que lhe atribui.

Comportamentos inline e block e como afetam ao fluxo da página

Quando tratamos com etiquetas, existem dois modos principais de comportamento. Etiquetas como uma imagem, ou um negrito, que funcionam em linha ("inline"), ou seja, que se colocam na linha onde se está escrevendo e onde os elementos seguintes, sempre que também sejam "inline" se posicionam todo seguido na mesma linha. Temos por outra parte os elementos que funcionam como bloco ("block") que implicam quebras de linha antes e depois do elemento.

Por exemplo, os parágrafos ou cabeçalhos são elementos com comportamento pré-determinado tipo "block".

Duas etiquetas muito utilizadas na construção CSS são DIV e SPAN. Uma das diferenças principais é que DIV funciona com comportamento "block" e SPAN funciona como "inline". Na verdade, este é o comportamento padrão, visto que nós com CSS em qualquer momento podemos mudá-lo por meio do atributo display. Por exemplo:

```
<div style="display: inline;">
Este elemento funcionará em linha
</div>
```

Ou então:

```
<span style="display: block;">
Este span agora funciona como bloco
</span>
```

Realmente ambas possibilidades funcionam dentro do fluxo HTML normal, por isso, tanto os elementos display inline como display block, se encontram dentro do fluxo HTML padrão, a única diferença é que os blocos se escrevem em linhas independentes, ou seja, com quebras de linha antes e depois do elemento, assim como uma quantidade de margem acima e abaixo que depende do tipo de elemento de que se trate.

Atributo CSS Float e o fluxo

Outro atributo que afeta o fluir dos elementos na página é o atributo float de CSS, que se utiliza bastante na construção web. Este atributo podemos utilizá-lo sobre elementos da página de tipo "block" e o que faz é convertê-los, em "flutuantes" que é um comportamento parecido ao que seria o mencionado anteriormente "inline". Com float podemos indicar tanto left como right e conseguiremos que os elementos se posicionem à esquerda ou à direita, com o conteúdo que se coloque a seguir rodeando ao elemento flutuante. A diferença é que os elementos continuam sendo tipo "block" e aceitam atributos como a margem (atributo CSS margin), para indicar que haja um espaço em branco aos lados e acima e abaixo do elemento.

Por exemplo, os elementos das listas (etiqueta LI) são por padrão de tipo "block", por isso aparecem sempre um abaixo do outro, em linhas consecutivas. Porém, nós poderíamos mudar esse comportamento com:

```
li{
  float: right;
}
```

Assim, uma lista como esta:

```
<ul>
<li>Elemento1</li>
<li>Elemento2</li>
<li>Elemento3</li>
</ul>
```

Veríamos como o primeiro elemento aparece à direita de tudo e os outros elementos vão se colocando na mesma linha no seguinte espaço livre que houver. Assim, o segundo elemento se colocaria na mesma linha, todo à direita que puder, conforme o espaço que tiver no contêiner aonde estiverem.

Fluxo e o atributo position

O atributo position de CSS sim que é capaz de mudar radicalmente o fluxo dos elementos da página. Este atributo, que explicaremos com detalhe mais adiante em outros artigos de CriarWeb.com, por padrão, tem o valor "static", que indica que o elemento faz parte do fluxo HTML normal da página.

Entretanto, com o atributo CSS position, podemos indicar outros valores que fazem com que os elementos saiam do fluxo HTML e se posicionem em lugares fixos, que não têm a ver com a posição na que apareçam no código HTML. Por exemplo:

```
<div style="position: absolute; top: 10px; left: 100px;">
Este elemento tem posicionamento absoluto
</div>
```

Faz com que esse elemento fique fora do fluxo de elementos na página e então, apareceria no lugar que se indica com os atributos top e left (top indica a distância desde a parte de cima e left a distância desde a borda esquerda). Os outros elementos que fazem parte do fluxo da página não ficam afetados pelos elementos com posicionamento absoluto.

Outro valor para o atributo position que faz com que os elementos fiquem posicionados fora do fluir normal de elementos na página é "fixed", cujo comportamento veremos mais adiante em outros artigos.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

O atributo Overflow de CSS

Neste artigo do Manual de CSS de CriarWeb.com vamos explicar uma propriedade interessante de CSS, contemplada na especificação CSS 2, o Overflow. Este é um desses atributos que nos servirão para maquetar as camadas de uma web de uma maneira mais versátil e detalhada.

Overflow serve no modelado de caixas para indicar ao navegador o que ele deve fazer com o conteúdo que não cabe dentro de uma camada, segundo as dimensões que lhe foram atribuídas.

Como sabemos, às camadas (elementos DIV) podemos atribuir-lhes um tamanho, em largura e altura. Porém, muitas vezes, o conteúdo que colocamos na camada ultrapassa o espaço que destinamos a ela. Então, o que costuma ocorrer é que a camada cresce o suficiente para que o conteúdo colocado dentro caiba. Habitualmente as camadas crescem em altura, razão pela qual quanto mais conteúdo mais tamanho terá em altura. Este é um comportamento que podemos alterar com o uso do atributo overflow.

Dito de outro modo, overflow permite que se oculte o conteúdo de uma camada, para mostrar unicamente o conteúdo que caiba, segundo suas dimensões. Para ter acesso ao conteúdo que não se mostra, porque não cabe na camada, pode se configurar overflow de modo que apareçam barras de rolagem.

Assim, passemos diretamente a ver quais são os atributos possíveis com o atributo overflow:

- **visible:** Este valor indica que se deve exibir todo o conteúdo da camada, ainda que não caiba no tamanho com a que a tenhamos configurado. Na Internet Explorer ocorre que a camada cresce em tamanho o suficiente para que caiba todo o conteúdo que colocamos dentro. No Firefox ocorre que a camada tem o tamanho marcado, mas o

conteúdo continua sendo visto, embora fora do espaço da camada , podendo sobrepor-se a um texto ou a uma imagem que haja embaixo. O conteúdo não se oculta em nenhum caso, ou seja, sempre estará visível.

- **hidden:** Este valor indica que os conteúdos que, pelo tamanho da camada, não caibam nela, se devem ocultar. Por isso, a camada terá sempre o tamanho configurado, mas os conteúdos em, algumas ocasiões, poderão não se ver por completo.
- **scroll:** Este valor indica que a camada deve ter o tamanho que se tenha configurado inicialmente e que, além disso, se devem mostrar barras de rolagem, para mover o conteúdo da camada dentro do espaço da mesma. As barras de rolagem sempre aparecem, solicitadas ou não.
- **auto:** Com este valor também se respeitarão as dimensões atribuídas a uma caixa. O conteúdo será ocultado, mas aparecerão as barras de rolagem para movê-lo. No entanto, neste caso, as barras de rolagem poderão aparecer ou não, dependendo se são necessárias ou não para exibir todo o conteúdo da camada.

Assim, o atributo overflow nos permitirá ter um maior controle sobre os espaços que destinamos a cada caixa de nosso desenho. É muito utilizado para mostrar textos longos, que se desejam integrar dentro de outro texto ou uma interface onde não temos espaço disponível para colocá-los ou não desejamos que cresçam mais da conta. Por exemplo, para mostrar código fonte dentro do texto de um artigo, é o seguinte:

```
<html>
<head>
  <title>Título</title>
</head>

<body>

Corpo...

</body>
</html>
```

Este exemplo, podemos apreciar a barra de rolagem vertical, se obtém com um atributo overflow: auto;. O código utilizado é o seguinte:

```
<div style="overflow: auto; width: 300px; height: 100px; background-color:#ededed; border: 1px solid #990000;">
CONTEUDO....
</div>
```

Agora vejamos outro exemplo, no qual simplesmente se oculta o texto que não cabe na camada. Indicamos overflow: hidden, razão pela qual o texto que sobra não vai ser visualizado.

Esta camada tem um conteúdo maior do que o cabe no espaço que designei com o atributo width e height. Como coloquei overflow: hidden, o que ocorrerá é que parte do texto que estou colocando não vai ser visto...

Neste caso vemos como o texto aparece ocultado, porque não cabe no espaço atribuído da camada. O código seria o seguinte:

```
<div style="overflow: hidden; width: 200px; height: 50px; border: 1px solid #990000;">
CONTEUDO...
</div>
```

Aqui se podem ver vários exemplos de uso de overflow.

*Artigo por **Miguel Angel Alvarez - Tradução de Celeste Veiga***

Posicionamento CSS

No Manual de CSS de CriarWeb.com tratamos em vários pontos do posicionamento CSS, com as distintas técnicas para criar e colocar partes da página de maneira absolutamente precisa. Em todos os casos explicamos diversos detalhes desde diferentes enfoques que sem dúvida dão uma visão particular sobre o posicionamento de elementos na página web. Por exemplo, já falamos sobre o que são as camadas, e também os atributos para seu posicionamento, bem como de outros assuntos como a construção CSS.

No entanto, ficou faltando oferecer algumas explicações gerais e detalhadas sobre o posicionamento CSS, que podem dar aos leitores uma idéia global sobre este assunto tão interessante. Sem dúvida, é um tema que vale a pena ser estudado e também praticado. Neste artigo vamos oferecer diversos conhecimentos teóricos ao tempo que estamos preparando um vídeo no qual mostraremos na prática as várias opções para o posicionamento web.

Atributos para posicionamento CSS

Existem numerosos atributos para posicionar qualquer elemento da página com CSS. Além disso, à medida que vão sendo apresentadas novas versões de CSS, estes atributos e seus possíveis valores vão aumentando. Em CSS 2 contamos com diversos atributos que veremos a seguir.

Atributo position:
Este atributo é, digamos, o principal para definir o tipo de posicionamento de um elemento. Vale a pena vê-lo separadamente e em detalhes. Trataremos dele mais adiante em outro artigo, mas adiantamos que ele vai permitir vários valores para estabelecer como se posicionará o elemento na página e se fará parte do fluxo normal de HTML. Seus valores possíveis são absolute, fixed, relative,static e inherit

Atributos top, left, right, bottom:
Servem para indicar a posição de um elemento, quando ele tem os valores de position "absolute", "relative" ou "fixed" (em outros valores do atributo position estes atributos são ignorados). O atributo top indica a distância da margem superior da página e left da margem da esquerda. Também se pode indicar opcionalmente a posição com bottom que é a distância de baixo e right , que é a distância da direita.

Atributos float e clear:
Float serve para estabelecer que um elemento tem que "flutuar", colocando-se os valores "right" ou "left" para que flutuem à esquerda ou à direita. Como esclarecimento, os elementos flutuarem é algo assim como o que acontece quando definimos o atributo HTML align="right" ou align ="left nas imagens ou tabelas. Com o atributo clear fazemos com que o elemento se coloque na primeira área livre que tenha lugar onde se indique. Por exemplo, o valor de clear "right" faz com que o elemento se coloque em primeiro lugar onde não tenha nenhum elemento flutuando á direita. O valor de clear "both" faz com que o elemento se coloque onde não haja elementos flutuando, tanto à direita quanto à esquerda.

Atributo clip:
Estabelece uma área de recorte da porção visível de um elemento. Esta área de recorte se estabelece com vários valores, como detalhado no artigo atributos para camadas:

Atributo display:
Especifica o tipo de caixa que deve ter um elemento que pode ser de diversas formas. Este atributo também tem bastante utilização e entre os valores mais comuns poderíamos destacar: "none" que faz com que esta caixa ou elemento não apareça na página nem se reserve espaço para ela, "block" serve para que a caixa seja um bloco e se mostre em uma linha ou linhas independente de outros elementos da página,"inline", que indica que essa caixa tem que aparecer na mesma linha que outros elementos escritos antes ou depois.

Atributo overflow:
Este atributo serve para dizer o que acontece com os elementos que não cabem em uma caixa devido às dimensões da mesma e do conteúdo que tenha.

Atributo visibility:
Atributo para definir a visibilidade de um elemento. Com este atributo podemos dizer que certos elementos da página sejam visíveis ou invisíveis, mas atenção porque embora um elemento seja invisível, continua ocupando espaço na página. Se quisermos que não seja invisível e nem se reserve espaço na página, há que se utilizar o atributo display com o valor "none".Os valores mais comuns de visibility são: "visible", que faz com que o elemento se veja (valor por defeito) e "hidden", que faz com que o elemento seja invisível, embora continue ocupando espaço.

Atributo z-index:
Este atributo tem como valor qualquer número inteiro. Serve para indicar que camada se tem que ver por cima ou por baixo de outra ou outras , no caso de várias camadas estarem superpostas. Com maiores valores de z-index, a camada se coloca mais à frente, cobrindo outras camadas que tenham valores menores que z-index.

Esta foi uma revisão geral dos distintos atributos de folhas de estilo que estão implicados no que se conhece como posicionamento em CSS. Como referência, aos interessados, recomendamos a leitura dos artigos mencionados no começo do Manual de CSS, especialmente o artigo sobre atributos para camadas.

No seguinte artigo veremos vários casos de utilização do atributo position, que é a chave para entender o posicionamento CSS.

*Artigo por **Miguel Angel Alvarez - Tradução de Celeste Veiga***

Tipos de posicionamento com o atributo position de CSS

No que se conhece como posicionamento CSS, o atributo de folhas de estilo em cascata que mais importância tem é o position. Vamos dedicar todo este artigo para explicar os diferentes valores de position, e propor exemplos que esperamos esclareçam os possíveis valores que pode adquirir.

Recordemos que no artigo anterior deste manual de CSS já se introduziu o conceito de posicionamento CSS e vimos uma lista dos atributos existentes até CSS 2 para posicionar elementos na página. Passemos então ao tema em questão, vendo as possibilidades que nos oferece esta linguagem. Além disso, ao longo deste artigo de CriarWeb.com, vamos mencionar repetidas vezes um conceito que também foi explicado anteriormente: o fluxo do HTML normal.

position: static

É o valor predeterminado do atributo e o posicionamento normal dos elementos na página. Significa que os elementos se colocarão segundo o fluxo normal do HTML, ou seja, segundo estejam escritos no próprio código HTML. Dito de outra maneira, static não provoca nenhum posicionamento especial dos elementos e portanto, os atributos top, left, right e bottom não se levam em conta.

Podemos ver um exemplo de posicionamento static:

```
<div style="position: static; background-color: #ff9; padding: 10px; width: 300px;">Isto é uma camada com posicionamento estático</div>
<div style="position: static; background-color: #f9f; padding: 10px; width: 500px;">posicionamento static, predeterminado.</div>
<h1>CSS</h1>
<div style="background-color: #9ff; padding: 10px; width: 400px;">Posicionamento static, embora neste caso não se tenha indicado o atributo position static, pois não é necessário.</div>
```

Pode ser visto em uma página à parte.

position: absolute

O valor absolute no atributo position permite posicionar elementos de maneira absoluta, isto é, de maneira definida pelos valores dos atributos top, left, bottom e right, que indicam a distância em relação a um ponto. As camadas ou elementos com posicionamento absoluto ficam à parte do fluxo normal do HTML, isto significa que não são afetados pelo lugar onde apareçam no código HTML e que também não afetam outros elementos que façam parte do fluxo normal do HTML.

Os valores top, left, bottom e right se expresam com unidades CSS e são uma distância referente ao primeiro elemento container que tenha um valor de position diferente de static. Se todos os containers onde esteja a camada posicionada com position absolute (todos seus pais até chegar a BODY) são static, simplesmente se posiciona em relação ao lado superior da página, para o caso de top, ou inferior para bottom, do lado esquerdo para left ou o direito, no caso de utilizar right.

Vejamos o seguinte código HTML no qual preparamos várias camadas com position absolute, porém com diferentes características:

```
<div style="position: absolute; width: 300px; height: 140px; top: 100px; left: 30px; background-color: #ff8800; color: #fff; padding: 15px;z-index: 2;">
Esta camada tem posicionamento absoluto.
<br>
<br>
Permite especificar top e left para colocá-la em relação ao canto superior esquerdo.
</div>

<div style="position: absolute; width: 820px; height: 30px; padding: 10px; background-color: #ddf; top: 150px; left: 10px; z-index: 1;">Posicionamento absoluto com z-index menor (a camada aparece abaixo das outras que se sobrepõem com z-index maior.</div>

<div style="position: absolute; width: 100px; height: 20px; padding: 10px; background-color: #ddf; bottom: 10px; right: 10px;">Posicionamento absoluto com atributos bottom e right</div>

<h1>Posicionamento CSS</h1>
```

Podemos ver o exemplo em uma página à parte.

A primeira camada (chamamos assim aos elementos DIV que têm posicionamento CSS), tem

como todas as do exemplo, posicionamento absoluto. Os atributos top: 100px e left: 30px significam que se posiciona a 100 pixels da parte superior da página e a 30 pixels da esquerda. Neste caso as distâncias top e left para localizar a camada com position absolute se referem ao canto superior esquerdo da área disponível do navegador, pois esta camada não está dentro de nenhuma outra com posicionamento diferente de static. Vale chamar a atenção nesta primeira camada sobre o atributo z-index: 2, que servirá para indicar ao navegador a posição da camada, na terceira dimensão, com relação a outras que se possam sobrepor, para que saiba qual deve estar abaixo e qual deve estar acima.

Podemos ver que a segunda camada tem um z-index:1. Isso quer dizer que, no caso de se posicionar no mesmo lugar se ocultará pela primeira camada, que tem um z-index maior.

Na terceira camada testamos o posicionamento utilizando os atributos bottom e right, assim que a estamos posicionando em relação ao canto inferior direito.

Vejamos um segundo exemplo onde vamos colocar uma camada com posicionamento absoluto e dentro várias camadas também posicionadas com absolute.

```
<div style="position: absolute; top: 100px; left: 200px; background-color: #ff9966; width: 400px; height: 100px;">
<div style="position: absolute; top: 10px; left:10px;">
Uno
</div>
<div style="position: absolute; top: 10px; left:100px;">
Dos
</div>
<div style="position: absolute; top: 10px; left:200px;">
Tres
</div>
</div>
```

Neste caso, a primeira camada, que não está dentro de nenhuma outra, se posiciona com top e left em relação ao canto superior esquerdo do espaço disponível no navegador para o corpo da página. As camadas aninhadas também estão com position: absolute, porém, por estarem dentro de outra camada que tem posicionamento diferente de static, seus valores top e left são relativos ao canto superior esquerdo da camada que as contém.

Podemos ver o exemplo em funcionamento em uma página à parte.

position: relative

O valor relative no atributo position indica que a camada sim faz parte do fluxo normal de elementos da página, razão pela qual sua posição dependerá do lugar onde esteja no código e do fluxo HTML. Além disso, as camadas com posicionamento relative, admitem os valores top e left para definir a distância em que se colocam em relação ao ponto onde esteja nesse momento o fluxo normal do HTML. Como afetam o mencionado fluxo do HTML, os elementos colocados depois das camadas relative, levarão em conta suas dimensões para continuar o fluxo e saber onde se colocar. No entanto, não se levam em conta os top e left configurados.

Vejamos um exemplo que talvez esclareça as coisas.

```
<h1>Hola</h1>
<div style="background-color: #606; color:#ffc; padding:10px; text-align: center; width: 300px;">Olá isto é um
teste</div>
<div style="position: relative; width: 300px; padding: 10px; background-color: #066; color:#ffc; top:100px; left:
30px;">camada de posicionamento relative<br>Leva- se em conta esta camada para posicionar as seguintes.</div>
<h2>olá de novo!</h2>
```

Podemos ver a página em funcionamento.

As etiquetas H1 e H2 respeitam o fluxo HTML e também temos um elemento DIV onde não se

especificou nada em position, assim é static e portanto também é afetada pelo fluxo. Há uma camada relative, no segundo elemento DIV, que também se posiciona em relação ao fluxo normal. Como tem um top e left, aparece um pouco fora do lugar que lhe caberia em relação ao fluxo.
O último H2 que aparece se coloca tendo em conta o fluxo e a camada relative, por isso deixa um espaço vazio acima, mas não atende à posição real desta, que foi marcada com os atributos top e left.

position: fixed

Este atributo serve para posicionar uma camada com posicionamento absoluto, mas sua posição final será sempre fixa, ou seja, embora se desloque o documento com as barras de rolagem do navegador, sempre aparecerá na mesma posição.

O lugar onde se "ancorará" a camada sempre é relativo ao corpo (o espaço disponível do navegador para a página). Se utilizamos top e left, estaremos marcando sua posição em relação ao canto superior esquerdo e se utilizamos bottom e right sua posição será relativa ao canto inferior direito.

Vejamos um exemplo.

```
<div style="position: fixed; width: 300px; height: 140px; top: 100px; left: 30px; background-color: #ff8800; color: #fff; padding: 15px;z-index: 1;">
Esta camada tem posicionamento fixed.
<br>
<br>
Permite especificar top e left para colocá-a em relação ao canto superior esquerdo.
</div>

<div style="position: fixed; width: 700px; height: 30px; padding: 10px; background-color: #d0f; top: 150px; left: 10px; z-index: 2;">Posicionamento fixed</div>
<h1>Ola</h1>
<div style="position: fixed; width: 100px; height: 30px; padding: 10px; background-color: #0df; bottom: 10px; right: 10px; z-index: 4;">Posicionamento fixed</div>
<br>
<br>
<br>
<br>
Ponho texto para que se veja!!
<br>
<br>
<br>
Isto faz deslocamento, com tanto br
<br>
<br>
...
<br>
```

Pode-se ver a página em funcionamento com este código.

Pode-se ver que há varias camadas com position: fixed e muito de BR para que a página possa ter um deslocamento. Se vemos a página em funcionamento e fazemos scroll para baixo com a barra de rolagem, veremos que as camadas fixed mantém sempre a mesma posição.

> **Nota:** O valor fixed no atributo position funciona em todos os navegadores, mas no caso de Internet Explorer só funciona na versão 7 e superiores. Além disso, para que funcione no Explorer tem que se declarar um DOCTYPE!. Servem vários tipos de DOCTYPE!, no entanto, devem ser declarados com o formato completo. Algo assim como:

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

Ou em outro exemplo:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

position: inherit

O valor inherit indica que o valor de position tem que ser herdado do elemento pai. Não funciona no Explorer, pelo menos até a versão 8. Tem na verdade pouca utilidade e além disso, como não funciona no navegador mais utilizado na atualidade, faz ainda menos sentido usá-lo. Por isso, não colocamos exemplos.

Conclusão sobre o atributo position de CSS

Esperamos que com as explicações anteriores e com os exemplos se tenha entendido bem as diferentes possibilidades do atributo position, que é sem dúvida chave para o posicionamento CSS. O mais comum para a construção web é utilizar o posicionamento static, mas o posicionamento absoluto, junto com posicionamento fixed, e inclusive relative, podem ser muito úteis para desenhos mais complexos, onde se requeira uma maior precisão na colocação dos diferentes elementos ou as camadas.

Além disso, para realizar efeitos Javascript e DHTML em geral, se utilizam frequentemente posicionamentos absolutos. São muito úteis porque permitem que os elementos dinâmicos não façam parte do fluxo normal do HTML, e portanto, podemos situá-los em qualquer lugar da área disponível do navegador, e inclusive movê-los dinamicamente ao trocar suas propriedades top e left mediante scripts no lado do cliente. Todos estes comportamentos dinâmicos já ficam fora da temática deste texto, embora os expliquemos com detalhe em distintos manuais da seção Javascript a Fundo, de CriarWeb.com.

*Artigo por **Miguel Angel Alvarez - Tradução de Celeste Veiga***