CSS Avançado

Conteúdo

```
• Rounded Corners: Corners. That are rounded.
• Shadows: Adding “pop” to boxes and text.
• Universal, Child, and Adjacent Selectors: More precise aim with clever
selectors.
• Advanced Colors: Alpha transparency and HSL.
• At-Rules: Importing style sheets, styles for different media types,
specifying the character set of a stylesheet and embedded fonts.
• Attribute Selectors: Targeting boxes by their elements’ HTML attributes.
• CSS Transitions: Creating smooth animations.
• Backgrounds: Multiples, Size, and Origin
• Transformations: Molding the size and shape of a box and its contents.
• Gradients: Linear and radial gradients without image files.
• Media Queries: Optimizing pages for different devices and screen sizes.
```

Rounded Corners/Cantos arredondados

Cantos arredondados costumavam ser o material para restringir imagens de fundo sólidas ou, para caixas flexíveis, várias imagens de fundo, uma por canto, colocadas em vários elementos div aninhados. Argh, feio. Bem, não mais. Agora, com CSS simples, você pode produzir seus designs com mais curvas do que Marilyn Monroe.

border-radius

Sim. Raio nas bordas. A propriedade border-radius pode ser usada para adicionar um canto a cada canto de uma caixa.

Detalhes:

https://www.w3schools.com/cssref/css3_pr_border-radius.asp

```
<style>
#example1 {
  border: 2px solid red;
  padding: 10px;
  border-radius: 25px;
}

#example2 {
  border: 2px solid red;
  padding: 10px;
  border-radius: 50px 20px;
}
</style>
</head>
<body>

<h2>border-radius: 25px:</h2>
<div id="example1">
```

```
  <p>The border-radius property defines the radius of the element's
corners.</p>
</div>

<h2>border-radius: 50px 20px:</h2>
<div id="example2">
  <p>If two values are set; the first one is for the top-left and bottom-
right corner, the second one for the top-right and bottom-left corner.</p>
</div>
```

Multiple values

border-top-left-radius, border-top-right-radius, border-bottom-right-radius e border-bottom-left-radius podem também ser usados para criar cantos arredondados.

Cada vez menos terrivelmente prolixo, você também pode definir todos os raios de canto individualmente com uma lista de valores separados por espaço, trabalhando no sentido horário a partir do canto superior esquerdo, assim como outras propriedades abreviadas: `css #monroe { background: #fff; width: 100px; height: 100px; border-radius: 6px 12px 18px 24px; }` Curvy.

Ao usar dois valores em vez de quatro, você segmenta top-left e bottom-right com o primeiro comprimento (ou porcentagem) e top-right + bottom-left com o segundo. Três valores? Top-left, then top-right + bottom-left, then bottom-right.

As versões 8 e anteriores do Internet Explorer não são compatíveis com border-radius. A única maneira de lidar com isso é se contentar com um design nos navegadores que não tenham cantos arredondados (a maioria das pessoas pode conviver com isso) ou reverter para as imagens de fundo antigas.

Você também pode se deparar com propriedades proprietárias semelhantes, como -webkit-border-radius e -moz-border-radius, que são para versões mais antigas e pouco utilizadas do Safari e Firefox, respectivamente.

Ellipses

Os círculos são quadrados demais para você? Você pode especificar diferentes raios horizontais e verticais dividindo os valores com um “/” e criando elipses.

```
#nanoo {
        background: #fff;
        width: 100px;
        height: 150px;
        border-radius: 50px/100px;
        border-bottom-left-radius: 50px;
        border-bottom-right-radius: 50px;
}
```

Exemplos

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Rounded corners: border-radius</title>
	<style>
		body {
			font: 14px courier;
			background: #06c;
			color: #000;
		}

		p {
			width: 300px;
			height: 100px;
			padding: 50px;
			margin: 20px 0 0 20px;
			background: white;
			float: left;
		}
		code {
			display: block;
			padding: 10px;
			background: #def;
			border-radius: 10px;
		}
		#corners1 {
			border-top-left-radius: 100px;
		}
		#corners2 {
			border-radius: 100px;
		}
		#corners3 {
			border-radius: 0 50px 100px 200px;
		}
		#corners4 {
			padding: 30px;
			background: none;
			border-radius: 50px;
			border: 20px solid #fff;
		}
		#corners5 {
			border-radius: 25%;
		}
		#corners6 {
			border-radius: 200px/100px;
		}
		#borderCollie {
			border-radius: 75px/125px;
			border-bottom-left-radius: 75px;
			border-bottom-right-radius: 75px;
			width: 120px;
			height: 90px;
			padding: 80px 15px 30px 15px;
		}
	</style>
</head>
<body>
	<p id="corners1"><code>border-top-left-radius: 100px;</code></p>
	<p id="corners2"><code>border-radius: 100px;</code></p>
	<p id="corners3"><code>border-radius: 0 50px 100px 200px;</code></p>
```

```
        <p id="corners4"><code>border-radius: 50px;<br>border: 20px solid
#fff;</code></p>
        <p id="corners5"><code>border-radius: 25%;</code></p>
        <p id="corners6"><code>border-radius: 200px/100px;</code></p>



        <!-- Link back to HTML Dog: -->
        <p id="borderCollie"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Transitions with border-radius and RGBa</title>
        <style>
                * {
                        margin: 0;
                }
                body {
                        font: 36px arial, helvetica, sans-serif;
                        color: #000;
                        background: #06c;
                }
                h1 {
                        margin: 40px;
                        font-size: 24px;
                        color: white;
                        text-align: center;
                }
                p {
                        height: 360px;
                        padding: 40px;
                }
                a, img {
                        display: block;
                        margin: 0 auto;
                        color: rgba(255,255,255,.5);
                        text-align: center;
                        text-decoration: none;
                        -webkit-transition: .5s;
                        transition: .5s;
                }
                a:hover {
                        background: rgba(255,255,255,.5);
                        color: white;
                        border-color: white;
                }

                #daddy {
                        background: rgba(0,0,0,.3);
                }
                #daddy a {
                        width: 120px;
                        height: 120px;
                        padding: 60px;
                        border-radius: 80px;
                        border: 60px solid rgba(255,255,255,.5);
                }
                #daddy a:hover {
```

```
                        border-radius: 180px;
                }

                #spurt {
                        background: rgba(0,0,0,.2);
                }
                #spurt a, #baby a {
                        width: 80px;
                        height: 80px;
                        padding: 40px;
                        border-radius: 60px;
                        border: 40px solid rgba(255,255,255,.5);
                        margin: 60px auto;
                        font-size: 24px;
                }
                #spurt a:hover {
                        width: 120px;
                        height: 120px;
                        padding: 60px;
                        border-width: 60px;
                        border-radius: 180px;
                        margin: 0 auto;
                        font-size: 36px;
                }

                #baby {
                        background: rgba(0,0,0,.1);
                }
                #baby a:hover {
                        width: 40px;
                        height: 40px;
                        padding: 20px;
                        border-width: 20px;
                        border-radius: 60px;
                        margin: 120px auto;
                        font-size: 12px;
                }



                #pet a {
                        width: 120px;
                        margin-top: 125px;
                        background: white;
                        border: 10px solid white;
                        border-radius: 10px;
                        opacity: .7;
                }
                #pet a:hover {
                        background: white;
                        box-shadow: 0 0 100px 50px rgba(255,255,255,.5);
                        opacity: 1;
                }
        </style>
</head>
<body>
        <h1>CSS transitions, using <code>border-radius</code> and RGBa
colors.</h1>
        <p id="daddy"><a href="">Big daddy link!</a></p>
        <p id="spurt"><a href="">Growth spurt link!</a></p>
        <p id="baby"><a href="">Shy baby link!</a></p>

        <!-- Link back to HTML Dog: -->
```

```
        <p id="pet"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

Shadows/Sombras

Veja! É como se alguém iluminasse minha página da web!

Você pode dar para partes de sua página sombras para boxes e para o texto.

box-shadow

box-shadow é a propriedade CSS padrão para você começar e pode ter um valor composto por várias partes:

box-shadow: 5px 5px 3px 1px #999

```
 • O primeiro valor é o deslocamento horizontal (horizontal offset) - o
quanto a sombra é empurrada para a direita (ou para a esquerda se for
negativo)
 • O segundo valor é o deslocamento vertical (vertical offset) - o quão longe
a sombra é empurrada para baixo (ou para cima se for negativo)
 • O terceiro valor é o raio de desfoque (blur radius) - quanto mais alto o
valor, menos nítida é a sombra. ("0" sendo absolutamente nítido). Isso é
opcional - omiti-lo é equivalente a definir "0".
 • O quarto valor é a distância de propagação (spread distance) - quanto mais
alto o valor, maior a sombra ("0" sendo o tamanho herdado da caixa). Isso
também é opcional - omiti-lo é equivalente a definir "0".
 • O quinto valor é uma cor (color). Isso também é opcional.
```

Inner shadows

Você também pode aplicar sombras no interior de uma caixa, adicionando inset à lista:

box-shadow: inset 0 0 7px 5px #ddd;

Você pode encontrar versões específicas do navegador de box-shadow, como -moz-box-shadow e -webkit-box-shadow. Ignore-os. Eles são velhos e estúpidos. A maioria dos navegadores modernos entende box-shadow, incluindo o Internet Explorer versões 9 e posteriores.

Text Shadows

box-shadow - box-shadow, como o próprio nome indica, só tem olhos para boxes. Fera inconstante. Mas você também pode aplicar sombras ao contorno do texto com (surpresa!) Sombra de texto:

text-shadow: -2px 2px 2px #999;

Da mesma forma que a sombra da caixa:

```
 • O primeiro valor é o deslocamento horizontal/horizontal offset
 • O segundo valor é o deslocamento vertical/vertical offset
 • O terceiro valor é o raio do desfoque/blur radius (opcional)
 • O quarto valor é a cor (opcional, embora omiti-lo tornará a sombra da
mesma cor do texto em si)
```

Observe que não há distância de propagação ou opção de inserção para sombra de texto.

text-shadow demorou um pouco mais para os navegadores descobrirem. O Internet Explorer 9 e versões anteriores não o compreenderão, por isso sugerimos usá-lo apenas em situações não críticas.

Exemplos

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Box shadows</title>
        <style>
                body {
                        font: 14px courier, monospace;
                        padding: 25px;
                        margin: 0;
                        background: #edc;
                        color: #000;
                }

                p {
                        width: 400px;
                        height: 50px;
                        padding: 50px;
                        margin: 25px;
                        background: white;
                        float: left;
                }
                #shadow1 {
                        box-shadow: 5px 5px;
                }
                #shadow2 {
                        box-shadow: 5px 5px 3px 1px #999;
                }
                #shadow3 {
                        box-shadow: 5px 5px 3px 1px rgba(0,0,0,.4);
                }
                #shadow4 {
                        box-shadow: 0 0 10px 0 rgba(0,0,0,.4);
                }
                #shadow5 {
                        box-shadow: inset 0 0 10px 0 rgba(0,0,0,.4);
                }
```

```
                #shadowBoxer {
                        width: 120px;
                        height: 90px;
                        padding: 30px 40px;
                        box-shadow: 0 0 20px 5px #06c;
                }
        </style>
</head>
<body>
        <p id="shadow1"><code>box-shadow: 5px 5px;</code></p>
        <p id="shadow2"><code>box-shadow: 5px 5px 3px 1px #999;</code></p>
        <p id="shadow3"><code>box-shadow: 5px 5px 3px 1px
rgba(0,0,0,.4);</code></p>
        <p id="shadow4"><code>box-shadow: 0 0 10px 0
rgba(0,0,0,.4);</code></p>
        <p id="shadow5"><code>box-shadow: inset 0 0 10px 0
rgba(0,0,0,.4);</code></p>

        <!-- Link back to HTML Dog: -->
        <p id="shadowBoxer"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Text shadows</title>
        <style>
                body {
                        font: 20px/2 "times new roman", times, serif;
                        color: #000;
                }
                h1 {
                        text-shadow: -2px 2px 2px #999;
                }
                em {
                        font-weight: bold;
                        padding: 8px;
                }
                #elephant {
                        color: #eee;
                        background: #999;
                        text-shadow: 2px 1px 0 #000;
                }
                #plesiosaur {
                        background: #bde;
                        color: #06c;
                        text-shadow: 0 -3px 1px #fff;
                }
                #tourist {
                        color: #efa;
                        background: #be0;
                        text-shadow: 0 0 4px #360;
                }
        </style>
</head>
<body>
        <h1>A little tale accompanied by a little text shadow</h1>
        <p>In Botswana's Chobe National Park, <em id="elephant">an elephant
with an especially large trunk</em> is minding its own business when <em
```

```
id="plesiosaur">a plesiosaur with five heads, two tails, and the legs of a
lion</em> falls out of the sky and lands on the elephant's back.</p>
        <p>"Hah hah! That's hilarious!", says the plesiosaur, "Your trunk is
huge!"</p>
        <p><em id="tourist">A stunned tourist in a nearby Jeep</em> drops his
camera, stares in amazement, and excitedly taps his wife on the shoulder.</p>
        <p>"Look, honey! That's amazing! The talking prehistoric marine
reptile with five heads, two tails, and the legs of a lion that fell out of
the sky is right! That elephant's trunk is ginormous!"</p>

        <!-- Link back to HTML Dog: -->
        <p><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

Universal, child e adjascent

Em tutoriais anteriores, cobrimos seletores de HTML, seletores de classe e ID e como combinar seletores para direcionar caixas de elemento específicas. Com o uso de três caracteres itty-bitty, você pode identificar ainda mais um elemento, reduzindo a necessidade de inchar seu HTML com atributos de classe e ID.

Universal selectors

Usando um asterisco (“*”), você pode direcionar tudo. Você pode usá-lo sozinho para definir estilos globais para uma página ou como um descendente de um seletor para definir estilos de tudo dentro de algo.

O seguinte, por exemplo, definirá a margem e o preenchimento de tudo em uma página como zero e tudo dentro de um elemento com o ID contato a ser exibido como um bloco: ```css * { margin: 0; padding: 0; }

#contact * { display: block; } ``` Um seletor universal autônomo é comumente usado para redefinir muitos dos estilos padrões de um navegador. Definir uma margem como zero, por exemplo, eliminará todo o espaçamento em torno de parágrafos, cabeçalhos e aspas em bloco.

Child selectors

Um símbolo maior que (“>”) pode ser usado para especificar um elemento que é filho de outro elemento, ou seja, algo imediatamente aninhado em algo.

Então, com este HTML ... `css <ul id="genus_examples"> <li>Cats <ul> <li>Panthera</li> <li>Felis</li> <li>Neofelis</li> </ul> </li> <li>Apes <ul> <li>Pongo</li> <li>Pan</li> <li>Homo</li> </ul> </li> </ul>` … E o seguinte CSS…

#genus_examples > li { border: 1px solid red }

… Uma borda vermelha será desenhada ao redor de Gatos e Macacos apenas, ao invés de ao redor de cada item da lista (o que seria o caso com #genus_examples li {border: 1px solid red}). Isso ocorre porque gente como “Panthera” e “Felis” são netos de “genus_examples”, não filhos.

Agora teste com este:

#genus_examples li { border: 1px solid red }

Adjacent selectors/Seletores adjascentes

Um sinal de mais (“+”) é usado para direcionar um irmão adjacente de um elemento, essencialmente, algo imediatamente após algo.

Com o seguinte HTML: `css <h1>Clouded leopards</h1> <p>Clouded leopards are cats that belong to the genus Neofelis.</p> <p>There are two extant species: Neofelis nebulosa and Neofelis diardi.</p>` e CSS

h1 + p { font-weight: bold }

Apenas o primeiro parágrafo, o que segue o título, ficará em negrito.

Um seletor CSS 3, irmão em geral/general sibling usa um til (“~”) e irá combinar um elemento seguindo outro independentemente de sua imediação. Portanto, no exemplo acima,

h1 ~ p {font-weight: bold}

estilizará todos os parágrafos após o título de nível superior, mas se houvesse qualquer ps precedendo h1, eles não seriam afetados.

Advanced Colors

Já sabemos que as cores podem ser definidas por nome, RGB ou valores hexadecimais, mas o CSS 3 também permite colorir com HSL - matiz, saturação e luminosidade - além de estipular transparência.

Não há propriedades superespeciais em jogo aqui - HSL e RGBa (o a que significa alpha, como em transparência alpha) podem ser aplicadas a qualquer propriedade que tenha um

valor de cor, como color, background-color , border-color ou box-shadow, para citar apenas alguns.

Alpha transparenceTransparência alfa

RGBa abre uma nova dimensão excitante para web design, permitindo que você defina a transparência de uma caixa ou texto. Se você quiser que uma pitada de uma imagem de fundo atraente apareça através de um título, por exemplo, você pode usar algo assim: `css h1 { padding: 50px; background-image: url(snazzy.jpg); color: rgba(0,0,0,0.8); }` Um valor padrão de rgb (0,0,0) definiria o título para preto puro, mas o quarto valor, em rgba, define o nível de transparência, 1 sendo completamente opaco, 0 sendo completamente transparente. Então rgba (0,0,0,0.8) está dizendo vermelho = 0, verde = 0, azul = 0, alfa = 0,8, o que, todos juntos, torna 80% preto. Isso não se aplica apenas ao texto, é claro, você pode aplicar uma cor de fundo transparente a uma box inteira, para uma sombra de box transparente ... em qualquer lugar onde você pode usar rgb, você pode usar rgba.

Hue, saturation, and lightness

Nomes de cores à parte, as cores da web sempre foram tendenciosas de vermelho-verde-azul, seja por meio de códigos hexadecimais ou RBG explícito (ou RGBa). Embora um pouco menos direto (especialmente se seu cérebro for treinado para dividir as cores em vermelho, verde e azul), o HSL pode realmente ser mais intuitivo porque dá a você controle direto sobre os aspectos da tonalidade de uma cor em vez de seus ingredientes lógicos.

É usado de maneira semelhante ao rgb:

#smut { color: hsl(36, 100%, 50%) }

Em vez de cada subvalor ser parte do espectro de cores, eles são:

```
 • Hue/Matiz ("36" no exemplo acima): qualquer ângulo, de 0 a 360, tirado de
uma roda de cores típica, onde "0" (e "360") é vermelho, "120" é verde e
"240" é azul.
 • Saturation/Saturação ("100%" no exemplo): Quão saturado você deseja que a
cor seja, de 0% (nenhum, então um nível de cinza dependendo da luminosidade)
a 100% (o golpe completo, por favor).
 • Lightness/Luminosidade ("50%" no exemplo): De 0% (preto) a 100% (branco),
sendo 50% "normal".
```

Portanto, o exemplo usado aqui produzirá uma laranja (36°) que é rica (saturação de 100%) e vibrante (50% lightness/Luminosidade). Isto é o equivalente de #ff9900, #f90 e rgb(255, 153, 0).

HSLa

Essa nova transparência e HSL podem ser combinados?! É melhor você acreditar. Aqui está HSLa:

#rabbit { background: hsla(0, 75%, 75%, 0.5) }

Você pode descobrir o que isso faz, certo?

hsl e hsla são suportados pela maioria dos navegadores modernos, incluindo IE versões 9 e superiores.

Exemplos

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Colors</title>
        <style>
                body {
                        font: 100 1.5em Helvetica, Arial, sans-serif;
                        color: white;
                        background-color: black;
                }

                #p1 {
                        color: #f83333;
                        background-color: #444;
                }
                #p2 {
                        color: rgb(0,255,127);
                        background-color: rgba(50%,50%,0%,0.5);
                }
                #p3 {
                        color: hsl(240,100%,75%);
                        background-color:  hsla(0,0%,100%,0.2);
                }
        </style>
</head>
<body>
        <h1>Colors</h1>
        <p>CSS color values can take <a
href="/references/css/values/color/">one of several forms</a>.</p>
        <p>This page's body is set to color: white; background-color:
black;</p>

        <p id="p1">This paragraph is set to color: #f83333; background-color:
#444;</p>

        <p id="p2">This paragraph is set to color: rgb(0,127,255);
background-color: rgba(0%,50%,50%,0.5);</p>

        <p id="p3">This paragraph is set to color: hsl(240,100%,75%);
background-color: hsla(0,0%,100%,0.2);</p>
```

```
        <!-- Link back to HTML Dog: -->
        <p><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Box shadows</title>
        <style>
                body {
                        font: 14px courier, monospace;
                        padding: 25px;
                        margin: 0;
                        background: #edc;
                        color: #000;
                }

                p {
                        width: 400px;
                        height: 50px;
                        padding: 50px;
                        margin: 25px;
                        background: white;
                        float: left;
                }
                #shadow1 {
                        box-shadow: 5px 5px;
                }
                #shadow2 {
                        box-shadow: 5px 5px 3px 1px #999;
                }
                #shadow3 {
                        box-shadow: 5px 5px 3px 1px rgba(0,0,0,.4);
                }
                #shadow4 {
                        box-shadow: 0 0 10px 0 rgba(0,0,0,.4);
                }
                #shadow5 {
                        box-shadow: inset 0 0 10px 0 rgba(0,0,0,.4);
                }


                #shadowBoxer {
                        width: 120px;
                        height: 90px;
                        padding: 30px 40px;
                        box-shadow: 0 0 20px 5px #06c;
                }
        </style>
</head>
<body>
        <p id="shadow1"><code>box-shadow: 5px 5px;</code></p>
        <p id="shadow2"><code>box-shadow: 5px 5px 3px 1px #999;</code></p>
        <p id="shadow3"><code>box-shadow: 5px 5px 3px 1px
rgba(0,0,0,.4);</code></p>
        <p id="shadow4"><code>box-shadow: 0 0 10px 0
rgba(0,0,0,.4);</code></p>
        <p id="shadow5"><code>box-shadow: inset 0 0 10px 0
rgba(0,0,0,.4);</code></p>
```

```
        <!-- Link back to HTML Dog: -->
        <p id="shadowBoxer"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Transitions with border-radius and RGBa</title>
        <style>
                * {
                        margin: 0;
                }
                body {
                        font: 36px arial, helvetica, sans-serif;
                        color: #000;
                        background: #06c;
                }
                h1 {
                        margin: 40px;
                        font-size: 24px;
                        color: white;
                        text-align: center;
                }
                p {
                        height: 360px;
                        padding: 40px;
                }
                a, img {
                        display: block;
                        margin: 0 auto;
                        color: rgba(255,255,255,.5);
                        text-align: center;
                        text-decoration: none;
                        -webkit-transition: .5s;
                        transition: .5s;
                }
                a:hover {
                        background: rgba(255,255,255,.5);
                        color: white;
                        border-color: white;
                }

                #daddy {
                        background: rgba(0,0,0,.3);
                }
                #daddy a {
                        width: 120px;
                        height: 120px;
                        padding: 60px;
                        border-radius: 80px;
                        border: 60px solid rgba(255,255,255,.5);
                }
                #daddy a:hover {
                        border-radius: 180px;
                }

                #spurt {
                        background: rgba(0,0,0,.2);
                }
                #spurt a, #baby a {
```

```
                        width: 80px;
                        height: 80px;
                        padding: 40px;
                        border-radius: 60px;
                        border: 40px solid rgba(255,255,255,.5);
                        margin: 60px auto;
                        font-size: 24px;
                }
                #spurt a:hover {
                        width: 120px;
                        height: 120px;
                        padding: 60px;
                        border-width: 60px;
                        border-radius: 180px;
                        margin: 0 auto;
                        font-size: 36px;
                }

                #baby {
                        background: rgba(0,0,0,.1);
                }
                #baby a:hover {
                        width: 40px;
                        height: 40px;
                        padding: 20px;
                        border-width: 20px;
                        border-radius: 60px;
                        margin: 120px auto;
                        font-size: 12px;
                }



                #pet a {
                        width: 120px;
                        margin-top: 125px;
                        background: white;
                        border: 10px solid white;
                        border-radius: 10px;
                        opacity: .7;
                }
                #pet a:hover {
                        background: white;
                        box-shadow: 0 0 100px 50px rgba(255,255,255,.5);
                        opacity: 1;
                }
        </style>
</head>
<body>
        <h1>CSS transitions, using <code>border-radius</code> and RGBa
colors.</h1>
        <p id="daddy"><a href="">Big daddy link!</a></p>
        <p id="spurt"><a href="">Growth spurt link!</a></p>
        <p id="baby"><a href="">Shy baby link!</a></p>

        <!-- Link back to HTML Dog: -->
        <p id="pet"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

At-rules

At-Rules: @import, @media, and @font-face

At-rules são pequenos abraços inteligentes e poderosos que encapsulam um monte de regras CSS e as aplicam a algo específico. Eles podem ser usados para importar outros arquivos CSS, aplicar CSS a uma mídia específica ou incorporar fontes incomuns.

Cada regra começa com um arroba @.

Importando

Vamos começar com a regra @import simples. Isso é usado para anexar outra folha de estilo em sua existente.

@import url(morestyles.css);

Isso pode ser usado se um site requer folhas de estilo longas e complexas que podem ser mais fáceis de gerenciar se forem divididas em arquivos menores.

Importante

As regras @import devem ser colocadas no topo de uma folha de estilo, antes de qualquer outra regra.

Targeting media types/Tipos de mídia de segmentação

```
@media pode ser usado para aplicar estilos a uma mídia específica, como impressão.

@media print {
    body {
        font-size: 10pt;
        font-family: times, serif;
    }

    #navigation {
        display: none;
    }
}
```

Os valores que seguem @media podem incluir screen, print, projection, handheld e all ou uma lista separada por vírgulas de mais de um, como: `css @media screen, projection { /* ... */ }` Não para por aí. O CSS 3 permite que você segmente não apenas uma mídia específica, mas também variáveis relacionadas a essa mídia, como o tamanho da tela (particularmente útil na segmentação de celulares). Dê uma olhada na página Media

Queries (https://www.htmldog.com/guides/css/advanced/mediaqueries/) para mais detalhes.

Embedding fonts

@font-face existe há muito tempo, mas foi quase inútil durante grande parte de sua vida. CSS 3 o aprimorou e agora é amplamente usado como uma técnica para incorporar fontes em uma página da web para que um tipo de letra possa ser usado mesmo que não esteja no computador do usuário. Assim, você não precisa mais depender de fontes seguras para a web, como Arial ou Verdana. Tempos emocionantes. `css @font-face { font-family: "font of all knowledge"; src: url(fontofallknowledge.woff); }` O que isso faz é criar uma fonte chamada font of all knowledge usando o descritor da família de fontes e vincular o arquivo de fonte fontofallknowledge.woff a esse nome usando o descritor src. font of all knowledge pode então ser usada em uma regra de fonte padrão, como: `css p { font-family: "font of all knowledge", arial, sans-serif; }` A fonte será baixada (neste caso do mesmo diretório do arquivo CSS) e aplicada aos parágrafos. Se o navegador estiver muito decrépito para lidar com novas fontes brilhantes, ele simplesmente reverterá para a próxima fonte padrão da lista. Magia!

Você também pode procurar várias fontes para aplicar à regra com uma lista separada por vírgulas. Verificar se uma fonte já está presente no computador de um usuário, eliminando a necessidade de baixá-la, também pode ser feito substituindo url no valor src por local.

E como um arquivo de fonte pode conter uma grande quantidade de pesos ou estilos, você também pode especificar em qual deles está interessado. Portanto, a regra @font-face pode ter a seguinte aparência: `css @font-face { font-family: "font of all knowledge"; src: local("font of all knowledge"), local(fontofallknowledge), url(fontofallknowledge.woff); font-weight: 400; font-style: normal; }` Legalmente falando, você não pode simplesmente jogar qualquer fonte antiga que você deseja na Internet porque existem direitos autorais e termos de uso a serem considerados, sem mencionar problemas de compatibilidade e otimização.

No entanto, existem fontes da web gratuitas que você pode encontrar, baixar, fazer upload e usar. Você mesmo pode criar uma se você for um cientista maluco inteligente. Também existem fornecedores de fontes da web, como o Typekit da Adobe, que têm uma grande seleção de fontes para escolher a um preço.

O Google Web Fonts tem uma seleção mais limitada, mas é de uso gratuito e torna as coisas super fáceis. Tudo que você precisa fazer é criar um link para um de seus arquivos CSS externos, que nada mais é do que uma regra @font-face, e besteira - você tem uma fonte nova e adorável para brincar.

Attribute Selectors

Antes de adicionar um atributo de classe a uma tag, veja se você pode simplesmente direcioná-lo por um atributo que pode já estar lá.

Os seletores de atributo permitem definir o estilo da box de um elemento com base na presença de um atributo HTML ou do valor de um atributo.

Com o atributo ...

Anexando um nome de atributo entre colchetes, para estilizar algo que simplesmente contém um determinado atributo, você pode fazer algo assim:

abbr[title] { border-bottom: 1px dotted #ccc }

Isso basicamente diz insira uma linha pontilhada abaixo de todas as abreviações com um atributo title.

Com atributo e seu valor…

Você pode ainda especificar que deseja estilizar algo com um valor de atributo específico.

input[type=text] { width: 200px; }

Este exemplo aplicará uma largura de 200 pixels apenas a elementos input que são do tipo text ().

Essa abordagem pode certamente ser útil em formulários em que os elementos de entrada podem assumir muitas formas usando o atributo type.

Você também pode direcionar mais de um atributo por vez de uma forma semelhante a esta:

input[type=text][disabled] { border: 1px solid #ccc }

Isso só aplicará uma borda cinza aos inputs do tipo text desabilitados.

O CSS 3 permite ainda que você combine um atributo sem ser exato: • [attribute ^ = algo] irá corresponder ao valor de um atributo que começa com algo. • [attribute $ = algo] irá

corresponder ao valor de um atributo que termina com algo. • [attribute * = algo] irá corresponder ao valor de um atributo que contém algo, seja no início, meio ou fim.

Portanto, como exemplo, você pode estilizar links externos (por exemplo, http://www.htmldog.com) de forma diferente de links internos (por exemplo, overhere.html) da seguinte maneira: `css a[href^=http] { padding-right: 10px; background: url(arrow.png) right no-repeat; }`

CSS Transitions

As transições permitem que você anime facilmente partes do seu design sem a necessidade de JavaScript.

No nível mais simplista, pense em um foco tradicional, onde você pode alterar a aparência de um link quando um cursor o acaricia: `css a:link { color: hsl(36,50%,50%); } a:hover { color: hsl(36,100%,50%); }` Isso cria uma animação binária; um link muda de um laranja suave para um laranja intenso quando passa o mouse sobre ele.

A propriedade de transição, no entanto, é muito mais poderosa, permitindo uma animação suave (ao invés de um salto de um estado para outro). É uma propriedade abreviada que combina as seguintes propriedades (que podem ser usadas individualmente se você quiser):

```
 • transition-property/propriedade de transição: qual propriedade (ou propriedades) fará a transição.
 • transition-durationduração da transição: quanto tempo leva a transição.
 • transition-timing-function/função de tempo de transição: se a transição ocorre a uma velocidade constante ou se acelera e desacelera.
 • transition-delay/atraso de transição: quanto tempo esperar até que a transição ocorra.
```

Voltando à propriedade abreviada, se uma transição for aplicada assim ... `css a:link { transition: all .5s linear 0; color: hsl(36,50%,50%); } a:hover { color: hsl(36,100%,50%); }` … Quando o ponteiro do mouse passa sobre o link, a cor muda gradualmente em vez de mudar bruscamente. A propriedade de transição aqui está dizendo que deseja que todas as propriedades tenham a transição de meio segundo de maneira linear sem atraso.

Para que uma transição ocorra, apenas a duração da transição é necessária, o resto padronizando para a propriedade da transição: all; transition-timing-function: ease; transition-delay: 0 ;. Portanto, você poderia, por exemplo, simplesmente declarar a transição: .5s.

Targeting specific properties

Embora “all” possa ser usado na propriedade transition-property (ou transição), você pode dizer a um navegador apenas para fazer a transição de certas propriedades,

simplesmente inserindo o nome da propriedade que deseja alterar. Portanto, o exemplo anterior poderia realmente incluir transição: cor .5s atenuação 0, considerando apenas as alterações de cor.

Se você deseja segmentar mais de uma propriedade (sem usar “all”), você pode oferecer uma lista separada por vírgulas com a propriedade de transição: `css a:link { transition: .5s; transition-property: color, font-size; ...` Ou você pode oferecer uma série de regras para a transição de cada propriedade, como: `css a:link { transition: color .5s, font-size 2s; ...`

Easing/Relaxamento

OK, então a função de tempo de transição (catchy/cativante!) é o cara menos óbvio. Ele efetivamente afirma como o navegador deve lidar com os estados intermediários da transição.

Ele segue uma curva de Bézier cúbica. Sim. Obviamente, sabemos tudo sobre eles desde a escola infantil, mas, para chegar ao fundo, no nível mais básico você pode usar ease ou linear.

ease produz uma aceleração gradual no início da transição e uma desaceleração gradual no final.

linear mantém uma velocidade constante durante todo o tempo. Outros valores incluem ease-in e ease-out.

As transições CSS não funcionam nas versões 9 e anteriores do Internet Explorer. Mas isso não importa nos casos (que serão na maioria dos casos) em que as transições não são essenciais para um design funcionar bem. É apenas algo para se manter em mente.

Exemplos

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Simple CSS transitions: Text links</title>
        <style>
                * {
                        margin: 0;
                }
                body {
                        font: 20px/1.5 arial, helvetica, sans-serif;
                        color: #000;
                        margin: 20px;
                }
                p {
```

```
                        margin: 20px 0;
                }

                a {
                        -webkit-transition: .5s;
                        transition: .5s;
                }

                #elephant {
                        color: #999;
                }
                #elephant:hover {
                        color: #f66;
                }

                #plesiosaur {
                        color: #06c;
                        text-decoration: none;
                        border-bottom: 3px solid #ddd;
                }
                #plesiosaur:hover {
                        border-color: #06c;
                }

                #tourist {
                        color: #f66;
                }
                #tourist:hover {
                        color: #c00;
                        background: #fcc
                }


                #htmldog a:hover {
                        transform: rotate(360deg);
                }
        </style>
</head>
<body>
        <h1>Simple <a
href="http://www.htmldog.com/guides/css/advanced/transitions/">CSS
transitions</a></h1>
        <p>Using <a
href="http://www.htmldog.com/references/css/properties/transition/">the
<code>transition</code> property</a> for basic animated effects when links
are hovered over.</p>
        <ul>
                <li><a href="" id="elephant">Changing color</a></li>
                <li><a href="" id="plesiosaur">Changing border color</a></li>
                <li><a href="" id="tourist">Changing color and background
color</a></li>
        </ul>

        <!-- Link back to HTML Dog: -->
        <p id="htmldog"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
```

```
        <title>Transitions with border-radius and RGBa</title>
        <style>
                * {
                        margin: 0;
                }
                body {
                        font: 36px arial, helvetica, sans-serif;
                        color: #000;
                        background: #06c;
                }
                h1 {
                        margin: 40px;
                        font-size: 24px;
                        color: white;
                        text-align: center;
                }
                p {
                        height: 360px;
                        padding: 40px;
                }
                a, img {
                        display: block;
                        margin: 0 auto;
                        color: rgba(255,255,255,.5);
                        text-align: center;
                        text-decoration: none;
                        -webkit-transition: .5s;
                        transition: .5s;
                }
                a:hover {
                        background: rgba(255,255,255,.5);
                        color: white;
                        border-color: white;
                }

                #daddy {
                        background: rgba(0,0,0,.3);
                }
                #daddy a {
                        width: 120px;
                        height: 120px;
                        padding: 60px;
                        border-radius: 80px;
                        border: 60px solid rgba(255,255,255,.5);
                }
                #daddy a:hover {
                        border-radius: 180px;
                }

                #spurt {
                        background: rgba(0,0,0,.2);
                }
                #spurt a, #baby a {
                        width: 80px;
                        height: 80px;
                        padding: 40px;
                        border-radius: 60px;
                        border: 40px solid rgba(255,255,255,.5);
                        margin: 60px auto;
                        font-size: 24px;
                }
                #spurt a:hover {
                        width: 120px;
```

```
                        height: 120px;
                        padding: 60px;
                        border-width: 60px;
                        border-radius: 180px;
                        margin: 0 auto;
                        font-size: 36px;
                }

                #baby {
                        background: rgba(0,0,0,.1);
                }
                #baby a:hover {
                        width: 40px;
                        height: 40px;
                        padding: 20px;
                        border-width: 20px;
                        border-radius: 60px;
                        margin: 120px auto;
                        font-size: 12px;
                }



                #pet a {
                        width: 120px;
                        margin-top: 125px;
                        background: white;
                        border: 10px solid white;
                        border-radius: 10px;
                        opacity: .7;
                }
                #pet a:hover {
                        background: white;
                        box-shadow: 0 0 100px 50px rgba(255,255,255,.5);
                        opacity: 1;
                }
        </style>
</head>
<body>
        <h1>CSS transitions, using <code>border-radius</code> and RGBa
colors.</h1>
        <p id="daddy"><a href="">Big daddy link!</a></p>
        <p id="spurt"><a href="">Growth spurt link!</a></p>
        <p id="baby"><a href="">Shy baby link!</a></p>

        <!-- Link back to HTML Dog: -->
        <p id="pet"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>CSS transitions: Timing functions</title>
        <style>
                * {
                        padding: 0;
                        margin: 0;
                }
                body {
                        font: 15px arial, helvetica, sans-serif;
```

```
                        text-align: center;
                }

                h1 {
                        margin-top: 20px;
                        color: #06c;
                }

                li {
                        list-style: none;
                        margin: 20px 0;
                        background: #def;
                }
                li a {
                        display: block;
                        width: 300px;
                        padding: 20px 0;
                        background: #06c;
                        color: white;
                }

                #timing1 {
                        -webkit-transition: 1s;
                        transition: 1s;
                }
                #timing2 {
                        -webkit-transition: 1s linear;
                        transition: 1s linear;
                }
                #timing3 {
                        -webkit-transition: 1s ease-in;
                        transition: 1s ease-in;
                }
                #timing4 {
                        -webkit-transition: 1s ease-out;
                        transition: 1s ease-out;
                }
                #timing5 {
                        -webkit-transition: 1s ease-in-out;
                        transition: 1s ease-in-out;
                }
                #timing6 {
                        -webkit-transition: 1s cubic-bezier(0.5,0.25,0,1);
                        transition: 1s cubic-bezier(0.5,0.25,0,1);
                }
                #timing7 {
                        -webkit-transition: 1s cubic-bezier(0.5,1.5,0.5,-
0.5);
                        transition: 1s cubic-bezier(0.5,1.5,0.5,-0.5);
                }
                #timing8 {
                        -webkit-transition: 1s steps(4);
                        transition: 1s steps(4);
                }

                li a:hover {
                        width: 100%;
                }
        </style>
</head>
<body>
        <h1>CSS transition timing functions</h1>
        <ul>
```

```
                <li><a href="" id="timing1"><code>ease</code>
(default)</a></li>
                <li><a href="" id="timing2"><code>linear</code></a></li>
                <li><a href="" id="timing3"><code>ease-in</code></a></li>
                <li><a href="" id="timing4"><code>ease-out</code></a></li>
                <li><a href="" id="timing5"><code>ease-in-out</code></a></li>
                <li><a href="" id="timing6"><code>cubic-
bezier(0.5,0.25,0,1)</code></a></li>
                <li><a href="" id="timing7"><code>cubic-bezier(0.5,1.5,0.5,-
0.5)</code></a></li>
                <li><a href="" id="timing8"><code>steps(4)</code></a></li>
        </ul>

        <!-- Link back to HTML Dog: -->
        <p><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

CSS Backgrounds

CSS Backgrounds: Multiples, Size, and Origin

Além de aplicar uma imagem de fundo única ou ladrilhada em partes da página, você também pode aplicar fundos múltiplos, ajustar o tamanho das imagens de fundo e definir a origem de um fundo com base nos níveis do modelo da box.

Multiple backgrounds/Vários fundos

CSS3 permite que você aplique várias imagens de fundo a uma única box, simplesmente colocando os locais das imagens em uma lista separada por vírgulas:

background-image: url(this.jpg), url(that.gif), url(theother.png);

Mais útil, você também pode definir todos os outros aspectos de fundo. Se você quiser estilizar um link em formato de botão com fundo, marcador e seta separados, por exemplo, pode usar algo como:

background: url(bg.png), url(bullet.png) 0 50% no-repeat, url(arrow.png) right no-repeat;

Fácil, certo? É o mesmo que usar background-image, background-position, background-repeat, background-attachment e background, exceto que você pode especificar mais de um fundo com a ajuda dessa vírgula útil.

Background size

A propriedade background-size permite esticar ou comprimir uma imagem de fundo.

background-size: contain e background-size: cover Os valores podem ser: • automático, que mantém o tamanho original da imagem de fundo e a proporção largura / altura. • comprimentos, largura e altura, como 100px 50px (100px de largura, 50px de altura). Especificar um único comprimento, como 100px, resultará no equivalente a 100px automático. • porcentagens, uma largura e uma altura, como 50% 25% (50% da largura da área de fundo, 25% da altura da área de fundo). Especificar uma única porcentagem, como 50%, resultará no equivalente a 50% automático. • Uma combinação de comprimentos, porcentagens e automático, como 80px automático (80px de largura, altura automática para manter a proporção original da imagem) • conter, que mantém a proporção original da imagem de fundo e a torna o maior possível, enquanto se ajusta inteiramente à área de fundo da caixa. • capa, que mantém a proporção original da imagem de fundo e a torna grande o suficiente para preencher toda a área de fundo, o que pode resultar no corte da altura ou da largura.

Background origin

background-origin é o garoto notavelmente chato do grupo. A propriedade define onde a área de fundo de uma box realmente começa. Se você pensar no modelo da box, ao definir um fundo, ele deve, por padrão, começar no canto superior esquerdo da box de preenchimento. Então, se você tivesse ... `css #burrito { width: 400px; height: 200px; border: 10px solid rgba(0,255,0,.5); padding: 20px; background: url(chilli.png) 0 0 no-repeat; }` … A imagem de fundo deve aparecer no canto superior esquerdo, na box de preenchimento, imediatamente após as bordas internas de uma borda verde. Você pode alterar esse comportamento, no entanto, com a origem do fundo. Seus valores autodescritivos podem ser padding-box (padrão, conforme descrito acima), border-box e content-box. Portanto, adicionar background-origin: border-box ao exemplo anterior fará com que a origem da imagem de fundo seja dobrada dentro da borda.

Exemplos

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Multiple backgrounds</title>
        <style>
                body {
                        font: 14px/1.5 courier, monospace;
                        background: white;
                        color: black;
                        margin: 20px;
                }

                pre, #htmldog {
                        background-color: #c72;
                        height: 200px;
                        margin: 20px 0;
                        overflow: auto;
                }
                pre code {
                        background: white;
```

```
                        padding: 2px 10px 4px 0;
                }

                #p0, #htmldog {
                        background-image:
url("http://www.htmldog.com/examples/images/bg.gif");
                }
                #p1 {
                        background-image:
url("http://www.htmldog.com/examples/images/circle.png"),
url("http://www.htmldog.com/examples/images/bg.gif");
                }
                #p2 {
                        background-image:
url("http://www.htmldog.com/examples/images/circle.png"),
url("http://www.htmldog.com/examples/images/bg.gif");
                        background-repeat: no-repeat, repeat;
                }
                #p3 {
                        background-image:
url("http://www.htmldog.com/examples/images/circle.png"),
url("http://www.htmldog.com/examples/images/bg.gif");
                        background-repeat: no-repeat, repeat;
                        background-position: center;
                }
                #p4 {
                        background:
url("http://www.htmldog.com/examples/images/circle.png") center no-repeat,
url("http://www.htmldog.com/examples/images/bg.gif");
                }

                #htmldog a {
                        display: block;
                        height: 200px;
                        background:
url("http://www.htmldog.com/examples/images/cornerbg.gif") top -15px left -
15px, url("http://www.htmldog.com/examples/images/cornerbg.gif") top -15px
right -15px, url("http://www.htmldog.com/examples/images/cornerbg.gif")
bottom -15px right -15px,
url("http://www.htmldog.com/examples/images/cornerbg.gif") bottom -15px left
-15px, url("http://www.htmldog.com//badge1.gif") center;
                        background-repeat: no-repeat;
                        text-indent: -999em;
                        transition: background-position .5s steps(2);
                }
                #htmldog a:hover {
                        background-position: top -25px left -25px, top -25px
right -25px, bottom -25px right -25px, bottom -25px left -25px, center;
                }
        </style>
</head>
<body>
        <h1><a
href="http://www.htmldog.com/guides/css/advanced/backgrounds/">Multiple
backgrounds</a></h1>
        <p>Using the <a
href="http://www.htmldog.com/references/css/properties/background-
image/"><code>background-image</code></a> and <a
href="http://www.htmldog.com/references/css/properties/background/"><code>bac
kground</code></a> CSS properties.</p>

        <pre id="p0"><code>background-image: url("bg.gif");</code></pre>
```

```
        <pre id="p1"><code>background-image: url("circle.png"),
url("bg.gif");</code></pre>
        <pre id="p2"><code>background-image: url("circle.png"),
url("bg.gif");</code>
<code>background-repeat: no-repeat, repeat;</code></pre>
        <pre id="p3"><code>background-image: url("circle.png"),
url("bg.gif");</code>
<code>background-repeat: no-repeat, repeat;</code>
<code>background-position: center;</code></pre>
        <pre id="p4"><code>background: url("circle.png") center no-repeat,
url("bg.gif");</code></pre>

        <!-- Link back to HTML Dog: -->
        <p id="htmldog"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>background-size</title>
        <style>
                body {
                        font: 14px courier;
                        background: #c72
url("/examples/images/opacityBg.gif");
                        color: #000;
                        margin: 0;
                        padding: 10px;
                }

                p {
                        background: #7a796b
url("/examples/images/tikal.jpg");
                        margin: 10px;
                        width: 300px;
                        height: 300px;
                        float: left;
                        text-align: center;
                }
                code {
                        background: white;
                        padding: 0 15px 2px;
                }

                #p1 {
                        background-size: auto;
                }
                #p2 {
                        background-size: 50%;
                }
                #p3 {
                        background-size: 100px;
                }
                #p4 {
                        background-size: 100px 100px;
                }
                #p5 {
                        background-size: contain;
                }
                #p6 {
```

```
                        background-size: cover;
                }

                #borderCollie a {
                        display: block;
                        height: 300px;
                        background: white
url("http://www.htmldog.com//badge1.gif") center no-repeat;
                        background-size: 60px 45px;
                        transition: background-size .5s;
                        text-indent: -999em;
                }
                #borderCollie a:hover {
                        background-size: 120px 90px;
                }
        </style>
</head>
<body>
        <p id="p1"><code>background-size: auto;</code></p>
        <p id="p2"><code>background-size: 50%;</code></p>
        <p id="p3"><code>background-size: 100px;</code></p>
        <p id="p4"><code>background-size: 100px 100px;</code></p>
        <p id="p5"><code>background-size: contain;</code></p>
        <p id="p6"><code>background-size: cover;</code></p>

        <!-- Link back to HTML Dog: -->
        <p id="borderCollie"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

Transformations

A poderosa manipulação da forma, tamanho e posição de uma caixa e seu conteúdo usando a propriedade transform.

Por padrão, as boxes CSS, aquelas representações visuais de elementos HTML, são tão quadradas. Retangular, pelo menos; nível, com quatro lados retos e quatro ângulos retos. Mas com o uso de transform você pode esticar e moldar essas boxes em todos os tipos de formas.

Esta página mencionará apenas as propriedades transform e transform-origin, mas, na prática, você provavelmente desejará duplicá-las com -webkit-transform e -webkit-transform-origin para obter os mesmos resultados em Safari e Chrome também as -ms-transform e -ms-transform-origin para o Internet Explorer 9, que é a versão mais antiga do IE que oferece suporte a transformações.

Manipulando uma box em duas dimensões, a transformação pode ser usada para girar, inclinar, dimensionar e traduzir uma caixa e seu conteúdo.

Rotating

O seguinte resultaria em uma caixa quadrada laranja com todo o seu conteúdo - texto, imagens, o que quer que seja - obedientemente em posição de atenção: `css .note { width: 300px; height: 300px; background: hsl(36,100%,50%); }` Mas você pode girar esses soldados adicionando outra declaração:

transform: rotate(-10deg);

Isso girará a box e, o mais importante, tudo nela, dez graus no sentido anti-horário.

Skewing

Skewing/inclinação permite que você incline a horizontal e a vertical para que o seguinte ...

transform: skew(20deg,10deg);

… Irá inclinar sobre o eixo x em 20 graus e no eixo y em 10 graus.

Você também pode especificar um ângulo, como inclinação (20 graus), que é equivalente a inclinação (20 graus, 0).

Scaling

Obviamente, você pode alterar as propriedades width/largura e height/altura em uma box, mas isso não afetará o tamanho de nada dentro dela. O dimensionamento, no entanto, multiplicará não apenas a largura e a altura, mas também o tamanho de tudo o que está contido na caixa:

transform: scale(2);

Isso multiplicará o tamanho por dois. Você pode usar qualquer número positivo, incluindo um valor menor que 1, como 0,5, se quiser usar um raio de redução. Você também pode dimensionar as dimensões horizontal e vertical separadamente:

transform: scale(1,2);

Isso deixará a horizontal como está (porque é uma escala de 1) e multiplicará a vertical por dois.

Translating

Você pode deslocar uma box horizontal e verticalmente usando transform: translate:

transform: translate(100px,200px);

Semelhante à posição: relative; left: 100px; top: 200px ;, isso moverá uma box 100 pixels para a direita e 200 pixels para baixo.

Assim como os valores mencionados, se você deseja atingir um eixo individual, também pode usar skewX, skewY, scaleX, scaleY, translateX e translateY.

Combining transformations

Quer girar e dimensionar ao mesmo tempo? Você pode fazer isso simplesmente separando os valores com espaços, como:

transform: rotate(-10deg) scale(2);

A ordem dos valores é importante - o último será executado antes do primeiro. No exemplo anterior, a caixa será dimensionada e girada. É, portanto, diferente transformar: scale (2) rotate (-10deg) ;, que será girado e depois escalado.

Alternativamente, você pode usar a função de matriz. o matrix faz tudo - girar, inclinar, dimensionar e traduzir. São seis valores:

transform: matrix(2,-0.35,0.35,2,0,0);

Esses valores não são totalmente diretos e envolvem matemática, se você realmente quiser abordar (há benefícios não apenas na brevidade, mas também na precisão), pode valer a pena dando uma olhada nas especificações.

Origin

Há um aspecto importante faltando. Se você está transformando uma caixa, há um outro parâmetro envolvido: a origem da transformação. Se você estiver girando, por exemplo, ele irá, por padrão, ligar o ponto que é o centro da caixa; se você tivesse um pedaço de cartão, enfiasse um alfinete bem no meio dele e, em seguida, colasse na sua testa, o cartão giraria do meio. Você pode alterar onde no cartão o pino está preso com a origem da transformação, no entanto:

transform-origin: 0 0;

Este exemplo dirá à caixa para se transformar (girar, no exemplo anterior) do canto superior esquerdo, o primeiro 0 sendo horizontal, o segundo sendo vertical. Esses valores podem ser diferentes, é claro - como todos os outros x-y, e você pode usar os valores normais de centro, superior, direita, inferior, esquerda, comprimento e porcentagem, incluindo negativos.

E tudo isso com duas dimensões apenas. transform tem um poder ainda maior que também pode ser usado para magia tridimensional. No nível mais básico, você pode usar rotateX e rotateY, que irão girar na direção/towards ou longe/away de você nos eixos x e y, e existem os gostos de translate3d, scale3d e a intimidante matrix3d, todos dos quais têm dificuldades de navegador ainda maiores do que suas contrapartes 2D.

Exemplos

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>2D Transforms</title>
        <style>
                body {
                        font: 15px/1.5 helvetica, arial, sans-serif;
                        background: #333;
                        color: #ccc;
                        margin: 50px 0 50px 50px;
                }
                a, code { color: #fff }
                p { margin-bottom: 50px }

                pre, #htmldog {
                        float: left;
                        height: 300px;
                        width: 300px;
                        background-color: #666;
                        margin: 0 50px 50px 0;
                        white-space: normal;
                }
                pre code {
                        display: block;
                        height: 260px;
                        background: rgba(255,204,000,.7);
                        padding: 20px;
                }

                #c0 {
                        -webkit-transform: none; /* for some older browsers
*/
                        -ms-transform: none; /* for IE9 */
                        transform: none; /* default, obvs */
                }
                #c1 {
                        -webkit-transform: rotate(10deg);
                        -ms-transform: rotate(10deg);
```

```
                        transform: rotate(10deg);
                }
                #c2 {
                        -webkit-transform: skewX(10deg);
                        -ms-transform: skewX(10deg);
                        transform: skewX(10deg);
                }
                #c3 {
                        -webkit-transform: skewY(-10deg);
                        -ms-transform: skewY(-10deg);
                        transform: skewY(-10deg);
                }
                #c4 {
                        -webkit-transform: skew(10deg, -10deg);
                        -ms-transform: skew(10deg, -10deg);
                        transform: skew(10deg, -10deg);
                }
                #c5 {
                        -webkit-transform: scale(0.8);
                        -ms-transform: scale(0.8);
                        transform: scale(0.8);
                }
                #c6 {
                        -webkit-transform: scale(0.8, 1.2);
                        -ms-transform: scale(0.8, 1.2);
                        transform: scale(0.8, 1.2);
                }
                #c7 {
                        -webkit-transform: translate(25px, 10px);
                        -ms-transform: translate(25px, 10px);
                        transform: translate(25px, 10px);
                }
                #c8 {
                        -webkit-transform: scale(0.8) rotate(10deg)
translate(25px, 10px);
                        -ms-transform: scale(0.8) rotate(10deg)
translate(25px, 10px);
                        transform: scale(0.8) rotate(10deg) translate(25px,
10px);
                }
                #c9 {
                        -webkit-transform: matrix(0.787846, 0.138919, -
0.138919, 0.787846, 18.307, 11.3514);
                        -ms-transform: matrix(0.787846, 0.138919, -0.138919,
0.787846, 18.307, 11.3514);
                        transform: matrix(0.787846, 0.138919, -0.138919,
0.787846, 18.307, 11.3514);
                }

                #htmldog a {
                        display: block;
                        padding: 105px 90px;
                        background: rgba(255,204,000,.7);
                        -webkit-transform: scale(.8);
                        -ms-transform: scale(.8);
                        transform: scale(.8);
                        transition: .7s transform cubic-bezier(0.5,1.5,0.5,-
0.5);
                }
                #htmldog a:hover {
                        -webkit-transform: scale(1);
                        -ms-transform: scale(1);
                        transform: scale(1);
```

```
                }
        </style>
</head>
<body>
        <h1><a
href="http://www.htmldog.com/guides/css/advanced/transformations/">2D
Transforms</a></h1>
        <p>Using the <a
href="http://www.htmldog.com/references/css/properties/transform/"><code>tran
sform</code></a> CSS property.</p>

        <pre><code id="c0">transform: none;</code></pre>

        <pre><code id="c1">transform: rotate(10deg);</code></pre>

        <pre><code id="c2">transform: skewX(10deg);</code></pre>

        <pre><code id="c3">transform: skewY(-10deg);</code></pre>

        <pre><code id="c4">transform: skew(10deg, -10deg);</code></pre>

        <pre><code id="c5">transform: scale(0.8);</code></pre>

        <pre><code id="c6">transform: scale(0.8, 1.2);</code></pre>

        <pre><code id="c7">transform: translate(25px, 10px);</code></pre>

        <pre><code id="c8">transform: scale(0.8) rotate(10deg)
translate(25px, 10px);</code></pre>

        <pre><code id="c9">transform: matrix(0.787846, 0.138919, -0.138919,
0.787846, 18.307, 11.3514);</code></pre>

        <!-- Link back to HTML Dog: -->
        <p id="htmldog"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog" width="120"
height="90"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Transform origin</title>
        <style>
                body {
                        font: 15px/1.5 helvetica, arial, sans-serif;
                        background: #333;
                        color: #ccc;
                        margin: 50px 0 50px 50px;
                }
                a, code { color: white }
                p { margin-bottom: 50px }

                pre, #htmldog {
                        float: left;
                        height: 300px;
                        width: 300px;
                        background-color: #666;
                        margin: 0 50px 50px 0;
                        white-space: normal;
                        position: relative;
```

```
                }
                pre code {
                        display: block;
                        height: 260px;
                        background: rgba(255,204,000,.7);
                        padding: 20px;
                }
                pre:after {
                        content: "";
                        position: absolute;
                        width: 10px;
                        height: 10px;
                        background: red;
                        border: 5px solid white;
                        border-radius: 10px;
                        margin: -10px 0 0 -10px;
                }

                #p0 code {
                        -webkit-transform: rotate(10deg); /* for some older
browsers */
                        -webkit-transform-origin: 50% 50%;
                        -ms-transform: rotate(10deg); /* for IE9 */
                        -ms-transform-origin: 50% 50%;
                        transform: rotate(10deg);
                        transform-origin: 50% 50%; /* default */
                }
                #p0:after {
                        top: 50%;
                        left: 50%;
                }
                #p1 code {
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: top;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: top;
                        transform: rotate(10deg);
                        transform-origin: top;
                }
                #p1:after {
                        top: 0;
                        left: 50%;
                }
                #p2 code {
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: right;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: right;
                        transform: rotate(10deg);
                        transform-origin: right;
                }
                #p2:after {
                        top: 50%;
                        right: 0;
                        margin: -10px -10px 0 0;
                }
                #p3 code {
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: right top;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: right top;
                        transform: rotate(10deg);
                        transform-origin: right top;
```

```
                }
                #p3:after {
                        top: 0;
                        right: 0;
                        margin: -10px -10px 0 0;
                }
                #p4 code {
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: 25% 25%;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: 25% 25%;
                        transform: rotate(10deg);
                        transform-origin: 25% 25%;
                }
                #p4:after {
                        top: 25%;
                        left: 25%;
                }
                #p5 code {
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: -20px 20px;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: -20px 20px;
                        transform: rotate(10deg);
                        transform-origin: -20px 20px;
                }
                #p5:after {
                        top: 20px;
                        left: -20px;
                }

                #htmldog a {
                        display: block;
                        padding: 105px 90px;
                        background: rgba(255,204,000,.7);
                        -webkit-transform: rotate(10deg);
                        -webkit-transform-origin: 0 0;
                        -ms-transform: rotate(10deg);
                        -ms-transform-origin: 0 0;
                        transform: rotate(10deg);
                        transform-origin: 0 0;
                        transition: 1s;
                }
                #htmldog a:hover {
                        -webkit-transform: rotate(-10deg);
                        -webkit-transform-origin: right top;
                        -ms-transform: rotate(-10deg);
                        -ms-transform-origin: right top;
                        transform: rotate(-10deg);
                        transform-origin: right top;
                }
        </style>
</head>
<body>
        <h1><a
href="http://www.htmldog.com/guides/css/advanced/transformations/">Transform
origin</a></h1>
        <p>Using the <a
href="http://www.htmldog.com/references/css/properties/transform-
origin/"><code>transform-origin</code></a> CSS property to alter the point at
which transformations are measured from. Using a rotated 2D transformation as
an example.</p>
```

```
        <pre id="p0"><code id="c0">transform: rotate(10deg);
transform-origin: 50% 50%;
/* default */</code></pre>

        <pre id="p1"><code id="c1">transform: rotate(10deg);
transform-origin: top;</code></pre>

        <pre id="p2"><code id="c2">transform: rotate(10deg);
transform-origin: right;</code></pre>

        <pre id="p3"><code id="c3">transform: rotate(10deg);
transform-origin: right top;</code></pre>

        <pre id="p4"><code id="c4">transform: rotate(10deg);
transform-origin: 25% 25%;</code></pre>

        <pre id="p5"><code id="c5">transform: rotate(10deg);
transform-origin: -20px 20px;</code></pre>

        <!-- Link back to HTML Dog: -->
        <p id="htmldog"><a href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog" width="120"
height="90"></a></p>
</body>
</html>
```

Gradients

Imagens que mostram uma dissolução suave de uma cor para outra estão espalhadas por toda a web. No entanto, o CSS 3 permite que você os coloque em seus designs sem ter que criar um arquivo de imagem real.

Não há propriedade especial para isso; você simplesmente usa a propriedade background ou background-image e define seu gradiente em seu valor. Você pode criar gradientes lineares e radiais dessa maneira.

Linear gradients

Para uma distribuição uniforme de duas cores, esmaecendo de uma na parte superior para outra na parte inferior, uma declaração pode ser simplesmente algo como:

background: linear-gradient(yellow, red);

Para manipular o ângulo do desvanecimento, você posiciona to e o destino para o qual deseja que a transição siga. Você pode ir para um lado:

background: linear-gradient(to right, orange, red);

Ou um canto

background: linear-gradient(to bottom right, orange, red);

Ou qualquer ângulo que agrade a sua imaginação:

background: linear-gradient(20deg, orange, red);

Observe que o to é excluído com ângulos porque não há um destino explícito.

E por que parar em duas cores? Especifique quantos você ousar:

background: linear-gradient(hsl(0,100%,50%),hsl(60,100%,50%),hsl(120,100%,50%),hsl(180,100%,50%),hsl(240,100%,50%),hsl(300,100%,50%));

Para fazer os gradientes funcionarem no maior número possível de navegadores, você provavelmente também desejará usar -webkit-linear-gradient para cobrir o Safari e versões mais antigas do Chrome. Os valores destes também devem excluir a parte to, se usada.

Os gradientes CSS não serão reproduzidos com o IE 9 e anteriores, mas você ainda pode fazê-los, e qualquer outro navegador incapaz, usar o método tradicional de uma boa imagem à moda antiga, especificando-o primeiro, como um fallback (declarações posteriores apenas ser ignorado).

Então, tudo incluído, seus gradientes podem ser parecidos com isto: `css body { background: #666 url(fade.png) 0 0 repeat-y; background: -webkit-linear-gradient(right, #000, #666); background: linear-gradient(to right, #000, #666); }`

Radial gradients

Gradientes radiais, com uma cor começando de um ponto central e desbotando para outra cor, usam uma sintaxe semelhante:

background: radial-gradient(yellow, green);

Você também pode especificar a forma do fade. Por padrão, é uma elipse, espalhando-se para preencher a caixa de fundo, mas você pode forçá-la a ser circular, independentemente do formato da caixa:

background: radial-gradient(circle, yellow, green);

Usando closest-side, closest-corner, farthest-side e farthest-corner, você também pode especificar se o gradiente está contido pelos lados ou cantos da caixa mais próximos ou mais distantes da origem:

background: radial-gradient(circle closest-side, yellow, green);

E se você quiser colocar a origem do gradiente em algum lugar específico, você também pode usar at:

background: radial-gradient(at top left, yellow, green);

Mais notas de compatibilidade: o uso adicional de -webkit-radial-gradiente é sábio. A ordem dos parâmetros precisa ser alterada aqui e, como to, at é excluído.

Então:

background: radial-gradient(circle closest-side at 100px 200px , black, white);

Estaria acompanhado por:

background: -webkit-radial-gradient(100px 200px, circle closest-side, black, white);

Color stops

Se você não quiser uma mistura uniforme em seu gradiente, você pode especificar exatamente onde no gradiente cada cor entra em ação, logo após cada cor, começando em 0 e terminando em 100% (embora comprimentos também possam ser usados)

Então, só para deixar claro antes de mexer: • linear-gradient (black 0, white 100%) é o equivalente ao linear-gradient(black, white) • radial-gradient(??) é igual ao radial-gradient(??) • linear-gradient(red 0%, green 33.3%, blue 66.7%, black 100%) terá o mesmo resultado que linear-gradient(red, green, blue, black)

Isso porque, quando as posições são declaradas nesses exemplos, elas espaçam uniformemente as cores, que é o padrão quando nenhuma interrupção de cor é explicitamente definida.

Então, para continuar com os ajustes, você pode puxar e alongar com essas paradas:

background: linear-gradient(135deg, hsl(36,100%,50%) 10%, hsl(72,100%,50%) 60%, white 90%);

Repeating gradients

Um único gradiente preencherá uma box com os métodos anteriores, mas você pode usar repeating-linear-gradient e repeating-linear-gradient para construir sobre os pontos de cor e, bem, repetir o gradiente.

Para barras básicas de barras pretas e brancas, por exemplo:

background: repeating-linear-gradient(white, black 15px, white 30px);

Ou algo um pouco mais sólido:

background: repeating-radial-gradient(black, black 15px, white 15px, white 30px);

Exemplos

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Linear gradients</title>
	<style>
		html {
			background: -webkit-linear-gradient(left, yellow, red);
			background: linear-gradient(to left, yellow, red);
			height: 100%;
		}

		body {
			font: 14px/1.5 courier;
			color: #000;
		}

		p {
			width: 200px;
			height: 200px;
			padding: 20px;
			margin: 20px 0 0 20px;
			float: left;
			border: 1px solid yellow;
		}

		#gradient1 {
			background: #888 url(images/gradientLinear.jpg) repeat-x; /* Background images can be used for browsers that aren't capable of producing gradients */
			background: -webkit-linear-gradient(yellow, red); /* Backup for major browsers that can still handle gradients */
```

```
                        background: linear-gradient(yellow, red); /* The CSS3
standard */
                }

                #gradient2 {
                        background: -webkit-linear-gradient(right, yellow,
red);
                        background: linear-gradient(to right, yellow, red);
                }

                #gradient3 {
                        background: -webkit-linear-gradient(bottom right,
yellow, red);
                        background: linear-gradient(to bottom right, yellow,
red);
                }

                #gradient4 {
                        background: -webkit-linear-gradient(20deg, yellow,
red);
                        background: linear-gradient(20deg, yellow, red);
                }

                #gradient5 {
                        background: -webkit-linear-gradient(hsl(0,100%,50%),
hsl(60,100%,50%), hsl(120,100%,50%), hsl(180,100%,50%), hsl(240,100%,50%),
hsl(300,100%,50%));
                        background: linear-gradient(hsl(0,100%,50%),
hsl(60,100%,50%), hsl(120,100%,50%), hsl(180,100%,50%), hsl(240,100%,50%),
hsl(300,100%,50%));
                }

                #gradient6 {
                        background: -webkit-linear-gradient(135deg,
hsl(36,100%,50%) 10%, hsl(72,100%,50%) 60%, white 90%);
                        background: linear-gradient(135deg, hsl(36,100%,50%)
10%, hsl(72,100%,50%) 60%, white 90%);
                }

                #gradientCollie {
                        width: 120px;
                        height: 90px;
                        padding: 75px 60px;
                        background: -webkit-linear-gradient(white, #06c);
                        background: linear-gradient(white, #06c);
                }
        </style>
</head>
<body>
        <p id="gradient1"><code>background: linear-gradient(yellow,
red);</code></p>
        <p id="gradient2"><code>background: linear-gradient(to right, yellow,
red);</code></p>
        <p id="gradient3"><code>background: linear-gradient(to bottom right,
yellow, red);</code></p>
        <p id="gradient4"><code>background: linear-gradient(20deg, yellow,
red);</code></p>
        <p id="gradient5"><code>background: linear-gradient(hsl(0,100%,50%),
hsl(60,100%,50%), hsl(120,100%,50%), hsl(180,100%,50%), hsl(240,100%,50%),
hsl(300,100%,50%));</code></p>
        <p id="gradient6"><code>background: linear-gradient(135deg,
hsl(36,100%,50%) 10%, hsl(72,100%,50%) 60%, white 90%);</code></p>
```

```
        <!-- Link back to HTML Dog: -->
        <p id="gradientCollie"><a
href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Radial gradients</title>
        <style>
                html {
                        background: -webkit-radial-gradient(green, yellow);
                        background: radial-gradient(green, yellow);
                        height: 100%;
                }

                body {
                        font: 14px/1.5 courier;
                        color: #000;
                }

                p {
                        width: 300px;
                        height: 150px;
                        padding: 20px;
                        margin: 20px 0 0 20px;
                        float: left;
                        border: 1px solid green;
                }

                #gradient1 {
                        background: #888 url(images/gradientRadial.jpg); /*
Background images can be used for browsers that aren't capable of producing
gradients */
                        background: -webkit-radial-gradient(yellow, green);
/* Backup for major browsers that can still handle gradients */
                        background: radial-gradient(yellow, green); /* The
CSS3 standard */
                }

                #gradient2 {
                        background: -webkit-radial-gradient(circle, yellow,
green);
                        background: radial-gradient(circle, yellow, green);
                }

                #gradient3 {
                        background: -webkit-radial-gradient(circle closest-
side, yellow, green);
                        background: radial-gradient(circle closest-side,
yellow, green);
                }

                #gradient4 {
                        background: -webkit-radial-gradient(top left, yellow,
green);
                        background: radial-gradient(at top left, yellow,
green);
                }
```

```
                #gradient5 {
                        background: -webkit-repeating-radial-gradient(#8d0,
#0d0 10px, #9f0 10px, #0f0 20px);
                        background: repeating-radial-gradient(#8d0, #0d0
10px, #9f0 10px, #0f0 20px);
                }

                #gradientCollie {
                        width: 120px;
                        height: 90px;
                        padding: 50px 60px;
                        border-radius: 120px/95px;
                        background: -webkit-radial-gradient(white 50%, #06c
75%);
                        background: radial-gradient(white 50%, #06c 75%);
                }
        </style>
</head>
<body>
        <p id="gradient1"><code>background: radial-gradient(yellow,
green);</code></p>
        <p id="gradient2"><code>background: radial-gradient(circle, yellow,
green);</code></p>
        <p id="gradient3"><code>background: radial-gradient(circle closest-
side, yellow, green);</code></p>
        <p id="gradient4"><code>background: radial-gradient(at top left,
yellow, green);</code></p>
        <p id="gradient5"><code>background: repeating-radial-gradient(#8d0,
#0d0 10px, #9f0 10px, #0f0 20px);</code></p>

        <!-- Link back to HTML Dog: -->
        <p id="gradientCollie"><a
href="http://www.htmldog.com/examples/"><img
src="http://www.htmldog.com/badge1.gif" alt="HTML Dog"></a></p>
</body>
</html>
```

Media Queries

@media at-rules, usado para direcionar estilos em mídias específicas, como tela ou impressão, já foram abordadas. Mas isso pode ser levado a um nível ainda maior de sofisticação, permitindo que você especifique diferentes opções de design dependendo do tamanho da tela. Uma página pode então ser otimizada e disposta de maneira completamente diferente para telefones celulares, tablets e vários tamanhos de janela do navegador.

Para recapitular, se quisermos aplicar algum CSS apenas à mídia baseada em tela, por exemplo, uma opção seria inserir algo como o seguinte na parte inferior de uma folha de estilo: ```css @media screen {

```
body { font: 12px arial, sans-serif }
#nav { display: block }
```

} ```

Browser-size specific CSS

Para estender o exemplo anterior, não apenas a mídia baseada em tela pode ser direcionada, mas também a mídia baseada em tela de um determinado tamanho: `css @media screen and (max-width: 1000px) { #content { width: 100% } }` Isso diz a um navegador para aplicar um bloco de CSS quando sua janela de visualização tiver 1000 pixels de largura ou menos. Você pode usar isso para fazer algo tão simples como tornar uma área de conteúdo ou navegação mais restrita ou pode reorganizar completamente o layout de uma página (como empilhar seções de página umas sobre as outras em vez de exibi-las em colunas).

Você pode aplicar mais de uma regra @media, então você pode ter vários designs diferentes que dependem do tamanho do navegador: ```css @media screen and (max-width: 1000px) { #content { width: 100% } }

@media screen and (max-width: 800px) { #nav { float: none } }

@media screen and (max-width: 600px) { #content aside { float: none; display: block; } } ``` Observe que se, por exemplo, uma área de layout tivesse 600 pixels de largura ou menos, todos os três seriam aplicados (porque seria menor ou igual a 1000px, 800px e 600px).

Além de usar uma largura máxima geral na área de conteúdo principal (os elementos do artigo), este site também possui várias regras CSS que dependem do tamanho. Se você puder fazer isso, tente redimensionar seu navegador para ver as alterações entrarem em vigor.

Você também pode usar min-width no lugar de max-width se quiser fazer as coisas ao contrário e aplicar CSS a tamanhos iguais ou acima de um determinado comprimento.

Orientation-specific CSS

Se você deseja aplicar CSS dependendo da orientação do navegador, faça o seguinte: ```css @media screen and (orientation: landscape) { #nav { float: left } }

@media screen and (orientation: portrait) { #nav { float: none } } ``` Isso pode ser especialmente útil com dispositivos móveis ...

Device-specific CSS

Não estamos falando de CSS diferente para cada marca e modelo de laptop e telefone, mas podemos, com a consciência relativamente limpa (contanto que sejamos sensatos), especificar os gostos e as dimensões da tela do dispositivo e sua proporção de pixels.

Um tipo de mídia handheld existe e poderia ser usado como portátil @mídia handheld mas não é amplamente suportado e a distinção do que é handheld tornou-se obscura devido à proliferação de todos os tipos de dispositivos com todos os tipos de tamanhos de tela.

Width and height

device-width, device-height, min-device-width, max-device-width, min-device-height e max-device-height podem ser usados para atingir a resolução computada de um dispositivo: `css @media screen and (min-device-height: 768px) and (max-device-width: 1024px) { /* You can apply numerous conditions separated by "and" */ }`

Pixel ratio

É importante ter em mente que um pixel CSS não precisa ser igual a um pixel físico do monitor. Embora uma tela possa ter fisicamente 720 pixels de largura, um navegador pode realmente aplicar CSS assumindo que ela tenha 480 pixels de largura, por exemplo. Isso é feito para que uma página da web com design padrão caiba mais provavelmente na tela. Neste exemplo, há uma proporção de pixel de 1,5: 1: Há 1½ pixels físicos para cada pixel CSS. Um monitor de desktop padrão bog terá uma proporção de pixel de 1:1:Um pixel CSS para cada pixel físico.

Se você deseja aplicar estilos apenas a esses dispositivos, pode usar algo como: `css @media (device-pixel-ratio: 2) { body { background: url(twiceasbig.png) } }` Isso se aplica a aparelhos como o iPhone (4 e superior), com uma proporção de pixels de 2:1. Você também pode usar a proporção de pixel de dispositivo mínimo e proporção de pixel de dispositivo máximo.

Por falar em iPhones também use -webkit-device-pixel-ratio, etc, etc ...

Você também pode mexer na janela de visualização de um dispositivo para que toque como você deseja. Voltando ao HTML, o seguinte metaelemento forçará um dispositivo a renderizar uma página na largura da janela de visualização (em vez de tentar mostrar um design de largura maior e diminuir o zoom, o que fará por padrão se a página puder ser mais larga do que a largura da janela de visualização) e também evita que o usuário aumente e diminua o zoom: `css <meta name = "viewport" content = "width = device-width, initial-scale = 1, maximum-scale = 1, user-scalable = no">` A vantagem disso é que você pode controlar exatamente o que é aplicado ao tamanho físico da tela. E embora seja doloroso remover os controles do usuário, se o design for deliciosamente grande e feito apenas para aquela tela pequena, não deve haver necessidade de zoom.

O site HTML Dog segue esta abordagem: em vez de um pequeno dispositivo tentar renderizar uma página maior e mais gorda reduzindo-a, o CSS a transforma em um design de coluna única feito especificamente para tal dispositivo.

Others

Você também pode aplicar estilos dependendo da resolução do dispositivo:

@media screen and (resolution: 326dpi) { /* */ }

@media screen and (min-resolution: 96dpi) { /* */ }

Ou em sua proporção:

@media screen and (device-aspect-ratio: 16/9) { /* */ }

Original em inglês

https://www.htmldog.com/guides/css/

Créditos pelo original em inglês

https://www.htmldog.com/guides/html/advanced/