© 2012, Casa do Código

Todos os direitos reservados e protegidos pela Lei nº9.610, de 10/02/1998. Nenhuma parte deste livro poderá ser reproduzida, nem transmitida, sem auto rização prévia por escrito da editora, sejam quais forem os meios: fotográficos, eletrônicos, mecânicos, gravação ou quaisquer outros.

Casa do Código

Livros para o programador

Rua Vergueiro, 3185 - 8º andar

04101-300 – Vila Mariana – São Paulo – SP – Brasil

Agradecimentos

*“Nothing of me is original. I am the combined eort of everybody I’ve ever known.”* – Chuck Palahniuk, Invisible Monsters (1999)

Tive a ajuda de muitas pessoas durante a criação deste livro, e sou extremamente grato a elas pela sua participação durante os meses em que trabalhei nele: Ao Adriano Almeida e ao Paulo Silveira pela oportunidade única de escrever este livro, e a ajuda de todos da Casa do Código e da Caelum.

A minha família e a minha namorada, pela paciência e ajuda durante estes meses, por todas as noites e nais de semana dedicados a este projeto. A incrível equipe da Plataformatec, que é muito boa no que faz. Por todos os projetos, apresentações, cafés e chopps compartilhados até então - que muitos outros venham no futuro!

A todos os meus colegas e amigos de projetos e empresas passadas - devo muito a todos pelo conhecimento compartilhado ao longo dos anos, que sem dúvida foi muito importante para eu chegar até aqui.

E a você, leitor, que está prestes a ler o meu livro. Muito obrigado por dedicar o seu tempo e a sua atenção a ele. Espero contribuir para que você crie ótimos projetos e faça a sua parte para criar uma web melhor.

Sumário

**1 O desenvolvimento web hoje 1** 1.1 Por quê você deve aprender HTML e CSS . . . . . . . . . . . . . . . . 2 1.2 O estado dos navegadores . . . . . . . . . . . . . . . . . . . . . . . . . 2 1.3 A complicação dos prexos proprietários . . . . . . . . . . . . . . . . 3 1.4 A longa e sinuosa estrada deste livro . . . . . . . . . . . . . . . . . . . 4

**2 Os primeiros passos com o nosso site 7** 2.1 Escrevendo HTML, de dentro para fora . . . . . . . . . . . . . . . . . 8 2.2 Adicionando formatações básicas . . . . . . . . . . . . . . . . . . . . . 10 2.3 Bordas e margens . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12 2.4 Um pouco de cor sempre é bom . . . . . . . . . . . . . . . . . . . . . . 14 2.5 Primeiro contato com imagens . . . . . . . . . . . . . . . . . . . . . . . 18 2.6 Adicionando elementos secundários . . . . . . . . . . . . . . . . . . . 22 2.7 Faça para sua cidade também! . . . . . . . . . . . . . . . . . . . . . . . 25

**3 HTML5: o que mudou? 27** 3.1 Escrevendo menos e fazendo mais . . . . . . . . . . . . . . . . . . . . . 27 3.2 Atributos personalizados . . . . . . . . . . . . . . . . . . . . . . . . . . 29 3.3 Tags novas para elementos antigos . . . . . . . . . . . . . . . . . . . . 29 3.4 Refatoração da página de São Paulo . . . . . . . . . . . . . . . . . . . . 31 3.5 Seja pragmático . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

**4 O que todo desenvolvedor precisa saber sobre CSS 35** 4.1 A incompatibilidade dos browsers e a razão dos resets de CSS . . . . 35 4.2 Compreendendo o Box model . . . . . . . . . . . . . . . . . . . . . . . 38 4.3 Utilizando pseudo elementos . . . . . . . . . . . . . . . . . . . . . . . . 41 4.4 Desenhando uma faixa com “*before” e “*aer” . . . . . . . . . . . . . . 41 4.5 Decorando mensagens . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

Sumário Casa do Código

4.6 Criando conteúdo através de CSS . . . . . . . . . . . . . . . . . . . . . 47 4.7 Arquitete o seu CSS para o futuro . . . . . . . . . . . . . . . . . . . . . 49 4.8 Gere relatórios inteligentes e simples com os estilos de impressão . . 52

**5 O que você consegue fazer com CSS 3 61** 5.1 A regra @font-face . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62 5.2 Como utilizar serviços de distribuição de fontes . . . . . . . . . . . . . 63 5.3 Substituição de ícones por fontes . . . . . . . . . . . . . . . . . . . . . 64 5.4 Explore novas possibilidades com bordas . . . . . . . . . . . . . . . . 67 5.5 Manipulação de cores com rgba e gradientes . . . . . . . . . . . . . . . 71 5.6 Trabalhe com sombras e crie menus elegantes . . . . . . . . . . . . . . 79 5.7 Combinando tudo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86

**6 Tomando controle da estrutura visual 95** 6.1 A propriedade ‘display’ . . . . . . . . . . . . . . . . . . . . . . . . . . . 96 6.2 Flutue elementos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97 6.3 O clearx, uma classe obrigatória em seus projetos . . . . . . . . . . . 106 6.4 Compreenda o uso de position . . . . . . . . . . . . . . . . . . . . . . . 107 6.5 Crie a sua própria janela modal . . . . . . . . . . . . . . . . . . . . . . 110 6.6 Como escolher os métodos para posicionar os seus elementos . . . . 116 6.7 Grids - um padrão de estrutura para as suas páginas . . . . . . . . . . 117 6.8 Posicionando elementos com CSS 3 . . . . . . . . . . . . . . . . . . . . 118

**7 Melhorando os seus formulários 123** 7.1 O que temos no HTML 5 . . . . . . . . . . . . . . . . . . . . . . . . . . 123 7.2 Formulários HTML 5 nos dispositivos móveis . . . . . . . . . . . . . . 124 7.3 Criação do primeiro formulário . . . . . . . . . . . . . . . . . . . . . . 125 7.4 Alinhamento e estilos visuais nos campos e formulários . . . . . . . . 127 7.5 Exibição de mensagens de ajuda . . . . . . . . . . . . . . . . . . . . . . 129 7.6 Mostre mensagens de erro . . . . . . . . . . . . . . . . . . . . . . . . . 132 7.7 Levando o usuário direto ao que importa com o autofocus . . . . . . 134 7.8 A exibilidade do atributo placeholder . . . . . . . . . . . . . . . . . . 134 7.9 Aplicando CSS3 em botões . . . . . . . . . . . . . . . . . . . . . . . . . 140

iv

Casa do Código Sumário

**8 Efeitos 101: Trabalhando com animações e transições 149** 8.1 Transformando elementos . . . . . . . . . . . . . . . . . . . . . . . . . 150 8.2 Os efeitos rotate, scale, skew e translate em uma galeria de fotos . . . 150 8.3 Transições de estilos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155 8.4 Transições na galeria de fotos . . . . . . . . . . . . . . . . . . . . . . . 156 8.5 Um detalhe importante sobre transições e JavaScript . . . . . . . . . . 160 8.6 Transformações em 3D . . . . . . . . . . . . . . . . . . . . . . . . . . . 161 8.7 Girar formulários com apenas um clique . . . . . . . . . . . . . . . . . 161 8.8 Utilizando animações . . . . . . . . . . . . . . . . . . . . . . . . . . . . 170 8.9 Começando com keyframes . . . . . . . . . . . . . . . . . . . . . . . . 170

**9 O universo fora dos desktops e notebooks 175** 9.1 O que é “Responsive Web Design” e porquê você deve se preocupar . 176 9.2 O funcionamento dos media queries . . . . . . . . . . . . . . . . . . . 177 9.3 Não é uma questão de aparelhos . . . . . . . . . . . . . . . . . . . . . . 178 9.4 Por um futuro melhor . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

**10 Ferramentas - Frameworks, Plugins e Pré-processadores 181** 10.1 Twitter Bootstrap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182 10.2 HTML5 Boilerplate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183 10.3 Plugins em JavaScript . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184 10.4 Modernizr . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184 10.5 Polylls . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185 10.6 Pré-processadores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 186 10.7 É tudo CSS e HTML . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189

**11 Não pare por aqui 191 Índice Remissivo 193 Bibliograa 196**

v

Capítulo 1

O desenvolvimento web hoje

Talvez eu seja um otimista, mas acredito que nos encontramos em uma época revo lucionária para o desenvolvimento web. A internet se proliferou pelo mundo graças aos avanços de banda larga e os diversos dispositivos capazes de navegar pela rede - celulares, tablets, televisões e videogames. Diversas empresas do mundo digital impulsionam a evolução das tecnologias que usamos para criar uma web melhor. Prossionais em empresas como Google, Microso, Apple e Facebook trabalham exclusivamente em melhorias para os navegadores mais utilizados, além de parti cipar da denição de novos padrões e disseminam bastante conhecimento junto a comunidade de desenvolvedores.

E tudo irá continuar a se expandir. Novas empresas, novos padrões, novos dispo sitivos. A indústria da internet vai continuar a crescer e precisamos de bastante ajuda para continuar esse trabalho. E é muito fácil garantir o seu espaço neste universo que se move como um trem bala. Basta participar.

1.1. Por quê você deve aprender HTML e CSS Casa do Código

**1.1 Por quê você deve aprender HTML e CSS**

Caso você seja um designer sem muita experiência com desenvolvimento, ou um programador mais acostumado a trabalhar com linhas de comando e compilado res do que navegadores, me deixe explicar um pouco a importância de se aprender HTML e CSS. Se você estiver trabalhando ou pretende trabalhar com tecnologia, acredito que boa parte - ou tudo - dos seus projetos será utilizado através de um na vegador. Seja um sistema interno de um banco, uma rede social, um grande portal de notícias ou sites para campanhas de publicidade, o meio comum hoje em dia é a web, e é bastante interessante ter uma ótima base de conhecimento sobre desen volvimento *front-end* (um dos termos usados para se referenciar a interface de uma aplicação) para contribuir para o sucesso dos projetos que você estiver participando.

Por isso, recomendo muito que você aprenda sobre o que faz a web funcionar. Além de livros como este, existem diversos cursos e sites para você aprender mais a respeito ou aprender outras tecnologias relacionadas. Imagine cuidar de um parque de servidores sem ter bons conhecimentos de sistemas operacionais e de topologia de redes, ou trabalhar em propaganda impressa sem entender sobre as cores ou o papel usado.

**1.2 O estado dos navegadores**

Todo o HTML e o CSS que escrevemos ganha vida dentro dos navegadores utilizados por quem acessa nossas páginas e sites, por isso é bastante importante ter um bom entendimento de como eles funcionam e, principalmente, dos buracos no meio do caminho. Desde os tempos do Mosaic (que eventualmente se tornou o Netscape) e das primeiras versões do Internet Explorer em 1995 sempre enfrentamos uma guerra de incompatibilidade entre os nevagadores e a necessidade de padrões para garantir a interoperabilidade da web entre clientes diferentes. Enquanto essa briga pode custar alguns cabelos nossos, ela também impulsiona a evolução das tecnologias que fazem os navegadores funcionar. O Google Chrome mudou a perspectiva sobre o processo de atualização contínua, garantindo que a maioria dos seus usuários possuam a úl tima versão do navegador, e o Firefox começou a adotar um processo similar. E o componente de debug e inspeção do Firefox, o Firebug, deniu o modelo básico para que os outros navegadores implementassem ferramentas similares.

Atualmente, o Google Chrome costuma ser o campeão dos testes de compatibi lidade com as últimas especicações, sempre seguido de perto das últimas versões do Firefox, Safari e Opera. Boa parte das inclusões do HTML5 e CSS3 já estão dis

2

Casa do Código Capítulo 1. O desenvolvimento web hoje

poníveis nesses navegadores por completo ou através de prexos proprietários. O Internet Explorer, atualmente na versão 9 (no momento da escrita desse livro a ver são 10 está em desenvolvimento), ainda se encontra bem atrás dos demais, mas as promessas para a sua próxima versão são interessantes.

Os perigos moram nos casos onde precisamos ir além das últimas versões e tra balhar com mais antigas, como versões do Firefox anteriores a 8 e os famigerados Internet Explorer 6 e 7. A diferença entre os navegadores e a performance do JavaS cript em comparação com as suas últimas versões é assombrosa, mas em alguns casos a audiência desejada se mantém presa a esses navegadores, por complicações para se atualizar o navegador ou até o sistema operacional - As últimas versões do Internet Explorer não são compatíveis com o Windows XP por exemplo, o que atrapalha a atualização em alguns ambientes corporativos.

Por isso, é importante denir as fronteiras dos navegadores que você pretende trabalhar, e se aproveitar das técnicas adequeadas ou das soluções existentes para problemas relacionados a eles.

**1.3 A complicação dos prefixos proprietários**

Para quem está acompanhando a crista da onda, diversas propriedades e funciona lidades novas não se encontram completamente denidas e implementadas nos na vegadores, mas isso não nos impede de utilizar versões “experimentais” delas. Para isso, cada navegador costuma expor essas novidades com prexos especícos para diferenciar da implementação nal. O que a princípio pode soar como uma ótima idéia, se tornou um fardo para todos os desenvolvedores: escrever as mesmas linhas de CSS (até) 4 vezes, assim:

.button {

/\* Prefixo para o Firefox. \*/

-moz-transition: all 0.2s linear;

/\* Prefixo para o Chrome, Safari, Safari Mobile e Android. \*/ -webkit-transition: all 0.2s linear;

/\* Prefixo para o Opera. \*/

-o-transition: all 0.2s linear;

/\* Versão final, ainda nao suportada em todos os browsers \*/ transition: all 0.2s linear;

}

Enquanto *hacks* como esse podem parecer desnecessários e culpados por sujar o código, é preciso entender a importância deles. Já estamos desfrutando e testando

3

1.4. A longa e sinuosa estrada deste livro Casa do Código

essas propriedades enquanto elas são denidas e renadas. O objetivo é, eventual mente, não precisar mais das versões prexadas e se utilizar apenas a versão canônica das propriedades.

Mas nem todos param para acompanhar o progresso desses itens. Propriedades como border-radius e box-shadow não precisam mais de prexos para os principais navegadores de desktops, e até o nal de 2012 outras propriedades como animation, transition e transform estarão nalizadas e devidamente implementadas.

Para isso, se precisa levar em consideração dois itens: Mantenha-se atualizado sobre o suporte dos navegadores para novas funcionalidades - minha referência fa vorita é o http://caniuse.com, que possui a relação de navegadores com suporte com pleto, parcial ou inexistente, e que eu costumo visitar de tempos em tempos - prin cipalmente ao começar novos projetos - para me atualizar sobre esse assunto. Outro ponto importante é que você deve mitigar ou automatizar o ruído e retrabalho ge rado pelo uso de prexos, seja com soluções como pré-processadores ou ferramentas em JavaScript, e focar o seu trabalho no que é tido como padrão, pois em algum mo mento os prexos não serão mais necessários.

Para isso vamos trabalhar com os padrões denidos e eventualmente citando os prexos quando necessários.

**1.4 A longa e sinuosa estrada deste livro**

Neste livro, vamos passar por diversas receitas e técnicas para você utilizar em seus projetos, e vamos ler e escrever **muito** código - muito mesmo. Usaremos tags e pro priedades que estão em uso a anos, além de passar por adições recentes do HTML5 e do CSS3. A minha intenção é mostrar exemplos práticos que provavelmente você chegou a ver ou irá usar no seu dia-a-dia, mas deixo essa avaliação a você. Eu sei que vários deles já me ajudaram uma ou duas vezes por aí.

Para tudo isso, precisamos de um ponto de saída, uma tela em branco para o nosso trabalho. Além da estrutura básica de HTML abaixo, vamos utilizar algu mas ferramentas para ajudar o nosso trabalho e deixar algumas partes não tão inte ressantes do desenvolvimento web de lado, assim podemos focar no que realmente importa. Utilizaremos o **Normalize.css** (http://necolas.github.com/normalize.css/) como um estilo de base, para garantir uma consistência melhor entre navegadores diferentes e o **-prex-free** (http://leaverou.github.com/prexfree/), que será respon sável por tratar os prexos proprietários para as propriedades de CSS3 que ainda não se tornaram um padrão entre navegadores. Além deles, iremos precisar de

4

Casa do Código Capítulo 1. O desenvolvimento web hoje

um pouco de JavaScript para nalizar alguns dos nossos exemplos, e ai o **jQuery** (http://jquery.com/) entrará em ação.

Mas não tema! Caso tudo isso pareça demais para você, que seguro que iremos entrar em detalhes sobre resets de CSS e sobre o -prex-free. Por enquanto, posso te garantir que eles irão ajudar bastante o nosso trabalho ao decorrer deste livro. Abaixo, segue o mínimo necessário de código HTML que iremos precisar.

Você pode baixar este template em http://lucasmazza.github.com/template.zip.

*<!doctype html>*

<html lang='pt-BR'>

<head>

<meta charset="UTF-8">

<title></title>

<link rel="stylesheet" href="normalize.css">

<script src="prefixfree.min.js"></script>

</head>

<body>

<!-- O seu HTML vem aqui! -->

<script src="jquery.min.js"></script>

</body>

</html>

Para o CSS especíco de cada exemplo, você pode criar um arquivo com um nome relacionado ao que estiver fazendo e adicionar uma referência a ele, logo abaixo do normalize.css. Assim, no exemplo de animações, você pode criar um arquivo animacoes.css e colocar uma tag link dentro da tag head.

*<!doctype html>*

<html lang='pt-BR'>

<head>

<meta charset="UTF-8">

<title>Trabalhando com animações!</title>

<link rel="stylesheet" href="normalize.css">

<link rel="stylesheet" href="animacoes.css">

<script src="prefixfree.min.js"></script>

</head>

<body>

<!-- O seu HTML vem aqui! -->

<script src="jquery.min.js"></script>

</body>

</html>

5

1.4. A longa e sinuosa estrada deste livro Casa do Código Com isso em mãos, hora de começar a trabalhar no que realmente interessa!

6

Capítulo 2

Os primeiros passos com o nosso site

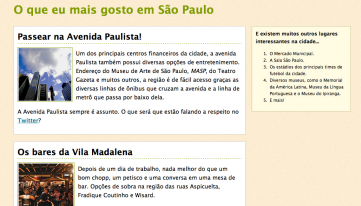
Antes de mergulharmos de cabeça em técnicas especícas, propriedades de CSS e combinações de tags e estilos, vamos praticar um pouco de *front-end* arroz com fei jão para garantir que estamos prontos para os demais assuntos e que as bases de desenvolvimento web estejam bem claras para você.

**São Paulo, São Paulo**

Quando você mora bastante tempo em uma mesma cidade você sempre dene uma lista de lugares e regiões da sua preferência, e sempre os indica para os seus ami gos de fora quando eles chegam para passar alguns dias. Para deixar a minha lista de paradas obrigatórias de São Paulo um pouco mais interessante de se ver vamos criar o **O que eu mais gosto em São Paulo...**, um site de uma página só para falar sobre alguns lugares da cidade. A página terá uma lista de 3 lugares de maior importância, com uma breve descrição, uma foto, e uma lista adicional de mais coisas a se ver na

2.1. Escrevendo HTML, de dentro para fora Casa do Código

cidade. Nós vamos criá-la do zero, e a versão nal cará assim:

Figura 2.1: O nosso site sobre lugares de São Paulo que não se pode deixar de visitar.

A versão nal da nossa página também está disponível online em http:// saopaulo.herokuapp.com.

**2.1 Escrevendo HTML, de dentro para fora**

Vamos começar pela parte mais importante da página, o conteúdo. Que no caso, são os 3 lugares que você não pode deixar de visitar em São Paulo. O primeiro, a Avenida Paulista, cará assim:

<h2>Passear na Avenida Paulista!</h2>

<p>

Um dos principais centros financeiros da cidade, a avenida Paulista também possui diversas opções de entretenimento. Endereço do Museu de Arte de São Paulo, <em>MASP</em>, do Teatro Gazeta e muitos outros, a região é de fácil acesso graças as diversas linhas de ônibus que cruzam a avenida e a linha de metrô que passa por baixo dela. </p>

<p>

A Avenida Paulista sempre é assunto. O que será que estão falando a 8

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

respeito no <a href='https://twitter.com/#!/search/Avenida Paulista' target='\_blank'>Twitter</a>?

</p>

O que temos aqui? Primeiro, um título secundário utilizando a tag h2, acompa nhado de dois parágrafos com texto, delimitados pela tag p. Dentro dos parágrafos, além do próprio texto, temos outras tags para denir mais elementos. Uma delas é a tag em, utilizada para indicar ênfase, que terá o seu texto exibido em itálico na página. O outro elemento, é um link para se navegar aos resultados de busca do Twitter para o termo “Av. Paulista”, é criado pela tag a (derivada da palavra *âncora* em inglês, *an chor*). O atributo href indica qual o endereço que o link aponta, e o target indica que o link deve ser aberto em uma nova página. Estas são algumas das principais tags utilizadas para se denir elementos de conteúdo, então você irá vê-las bastante por aqui.

Seguindo em frente, podemos adicionar o resto do nosso conteúdo, outros 2 lu gares. Um deles é a região da Vila Madalena, e o último é o Parque do Ibirapuera.

<h2>Os bares da Vila Madalena</h2>

<p>

Depois de um dia de trabalho, nada melhor do que um bom chopp, um petisco e uma conversa em uma mesa de bar. Opções de sobra na região das ruas Aspicuelta, Fradique Coutinho e Wisard.

</p>

<p>

Veja quais os melhores bares e restaurantes da região no <a href='http://vejasp.abril.com.br/busca?chn=bares&qu= &nbh=Pinheiros%20/%20Vila%20Madalena' target='\_blank'>

Guia da VEJA São Paulo.

</a>

</p>

<h2>O Parque do Ibirapuera</h2>

<p>

Um dos cartões postais da cidade, o parque dispõe de mais de 1,5 km{^2} de área verde, lagos artificiais e pistas de cooper e ciclismo. E se isso não fosse o suficiente, o parque costuma ser palco de diversos eventos culturais ao longo do ano.

</p>

<p>

Veja no <a href='http://goo.gl/maps/HoH8' target='\_blank'>mapa</a> 9

2.2. Adicionando formatações básicas Casa do Código

como chegar ao parque.

</p>

Nada de novo até o momento. Novos parágrafos, títulos e links para outros luga res - a busca de bares do site da Veja São Paulo e a localização do Parque no Google Maps. Com a parte principal no seu lugar, vamos continuar na lista de melhorias a fazer para a nossa página. Todo site precisa de um título, certo? Para isso vamos utilizar a tag h1, que representa um título principal. Além disso, dentro do elemento head, podemos preencher a tag title com um título para a janela do navegador.

*<!doctype html>*

<html lang='pt-BR'>

<head>

<!-- outras tags do head aqui -->

<title>O que eu mais gosto em São Paulo</h1>

</head>

<body>

<h1>O que eu mais gosto em São Paulo</h1>

<!-- o nosso conteúdo aqui... -->

</body>

</html>



Figura 2.2: Todo o nosso conteúdo, sem estilos.

Ótimo! Já podemos dizer que estamos com uma página em mãos, certo? Bem, o HTML está no seu lugar, mas precisamos de um pouco de CSS para melhorar a apresentação do nosso conteúdo.

**2.2 Adicionando formatações básicas**

Para começar o CSS da nossa página vamos xar a largura disponível para o conteúdo e centralizar o mesmo na página, deixando um espaço em branco nas laterais. Para

10

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

isso, adicionamos uma tag div em volta de todo o HTML que criamos dentro do body, com uma classe chamada container

<div class='container'>

<!-- Todo o corpo da página vem aqui... -->

</div>

E o CSS para centralizar esta div na página:

.container {

margin: 0 auto;

width: 960px;

}

A propriedade width xa a largura para 960px, adequando o conteúdo a telas de pelo menos 1024px de largura, como tablets e netbooks. A margem superior e inferior está com 0, e as margens laterais em auto cuidam de centralizar o elemento na página. O próximo passo é melhorar a exibição dos textos, ajustando um pouco o estilo das fontes com o seguinte código, que deve ser adicionado acima do CSS anterior:

body {

font-family: "Lucida Grande", "Lucida Sans Unicode", Verdana, sans-serif;

line-height: 1.6;

}

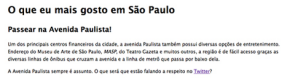
Aqui, estamos trocando a *família* de fontes utilizadas para uma lista de 4 fon tes diferentes que são utilizadas da seguinte maneira: caso a primeira fonte listada não esteja disponível no sistema operacional do usuário, o navegador tentará utili zar a próxima e assim sucessivamente. A Lucida Grande será a fonte utilizada pelo Mac OS X ao exibir a nossa página, enquanto a Lucida Sans Unicode será vista por usuários do Windows, já que este sistema operacional não possui a primeira fonte da nossa lista. Em casos mais extremos, Verdana será utilizado ou o navegador irá exibir uma fonte sem serifa por padrão. Uma forma de se contornar isso é instalando diversas fontes diferentes por conta própria, mas não podemos contar com isso. En quanto o line-height de 1.6, sem nenhuma medida especíca - como px ou em - indica para a altura das linhas carem com 160% da altura da fonte. Caso ela esteja em 16px, o line-height será 25px. Caso a altura da fonte mude, a altura da linha de

11

2.3. Bordas e margens Casa do Código

texto irá acompanhar essa mudança, respeitando a regra de sempre ser 1.6 da altura da fonte.

Não estamos alterando a altura da fonte por enquanto - a propriedade font-size, para utilizar o valor padrão do navegador, que costuma ser de 16px em desktops e notebooks, enquanto outros aparelhos ou navegadores com congura ções diferentes possuem um tamanho padrão diferente. Outro detalhe importante, é que estamos adicionado estes estilos a tag body. Desta forma, todos os elementos da página irão herdar essas regras “padrões” da nossa página, e podemos alterá-las conforme a necessidade de cada elemento.

Figura 2.3: O container de 960px de largura e a alteração de fonte.

**2.3 Bordas e margens**

Se analisarmos com mais detalhe o nosso conteúdo, estamos falando sobre 3 luga res diferentes, onde cada um possui um título e uma descrição. Para essa denição transparecer no nosso código, podemos colocar os elementos de cada lugar dentro de um novo elemento, para agrupar cada um.

<div class='place'>

<!-- O título e a descrição da Avenida Paulista... --> </div>

<div class='place'>

<!-- O título e a descrição da Vila Madalena... -->

</div>

<div class='place'>

<!-- O título e a descrição do Parque do Ibirapuera... --> </div>

Esta alteração não tem impacto nenhum no visual da nossa página, mas nos dá por onde adicionar um estilo para cada um dos 3 lugares, já que agora existe um

12

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

elemento especíco que engloba o seu conteúdo. Assim, adicionamos um pouco de espaçamento e bordas a cada um deles.

.place {

border-color: #CCC #999 #999 #CCC;

border: 1px solid #CCC;

margin-bottom: 20px;

padding: 10px;

}

Agora, cada elemento que contenha o class='place', ou lugar, terá uma borda sólida - uma linha - de 1px. A princípio todas as bordas são do mesmo tom de cinza, mas vamos mudar as bordas da direita e de baixo para outro tom de cor, repe tindo o mesmo cinza para as outras bordas, através do border-color. Além disso, o margin-bottom, coloca 20px abaixo de cada um dos elementos, adicionando espaço entre eles e ao término da lista. Enquanto a propriedade margin-bottom adiciona o espaçamento do lado de fora dos elementos, a propriedade padding adiciona o espaçamento dentro dele, que no caso é de 10px em todos os lados.

As três propriedades - padding, margin e border - funcionam de maneira similar, sendo possível denir um valor especíco para cada um dos 4 lados do elemento. Caso você queria denir o valor para apenas um deles, é possível utilizar a variação -top/right/bottom/left, como o margin-bottom. Enquanto o uso da propriedade sem o suxo vai aplicar o mesmo valor para todos os lados do elemento, ou você pode informar uma lista de valores diferentes e eles serão aplicados em sentido horário - o primeiro será para o topo (top), o segundo para o lado direito e assim por diante.

Apesar dos 10px de padding, o espaçamento entre a borda e o título aparenta ser mais do que isso. Isso é resultado do margin padrão do elemento h2 somado com o padding da div que encapsula. Para ajustar isso, podemos remover a margem do h2 completamente. De brinde, adicionamos uma borda verde abaixo do título.

.place h2 {

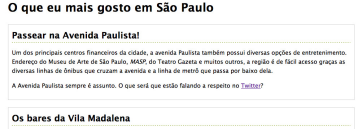
border-bottom: 1px dashed #7E9F19;

margin: 0;

}

Assim, todos os lados do elemento .place terão o mesmo espaçamento de 10px. Este estilo só se aplicará as ocorrências de h2 **dentro** de outro elemento, que possua a classe place. E diferente da borda sólida de antes, utilizamos o dashed no lugar de solid, para criar um efeito diferente, com uma borda tracejada.

13

2.4. Um pouco de cor sempre é bom Casa do Código Figura 2.4: As bordas e o espaçamento da classe ’.place’.

**2.4 Um pouco de cor sempre é bom**

Hora de trocar o preto e branco da nossa página e colocar algumas cores. Primeiro, cuidaremos do fundo da página. Além de uma nova cor nova, vamos combinar isso com uma imagem com um efeito de ruído, que será aplicado em cima da cor de fundo, assim:

body {

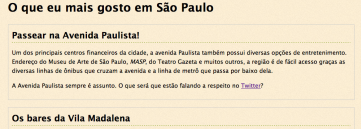
background: #FFF1D6 url(images/noise.png);

font-family: "Lucida Grande", "Lucida Sans Unicode", Verdana, sans-serif;

line-height: 1.6;

}

14

Casa do Código Capítulo 2. Os primeiros passos com o nosso site Figura 2.5: O fundo da nossa página.

Como a imagem irá preencher todo o fundo do body, não precisamos nos pre ocupar com as propriedades de background-repeat ou background-position, que são usadas para controlar mais detalhes sobre a aplicação das imagens de fundo. Lembre-se se ter a imagem noise.png no caminho correspondente ao que está no código - no nosso caso, em um diretório images que está junto ao arquivo .html da página. Você pode baixar este arquivo em http://saopaulo.herokuapp.com/images/ noise.png.

Agora, os elementos .place precisam de um fundo diferente, para ajudar a lei tura dos textos. Podemos então colocar um fundo branco neles.

.place {

background-color: #FFF;

border: 1px solid #CCC;

border-color: #CCC #999 #999 #CCC;

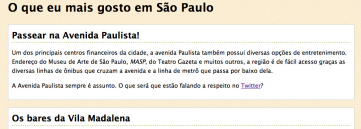
margin-bottom: 20px;

padding: 10px;

}

Melhor, certo? Como estamos apenas mudando a cor de fundo, podemos utilizar tanto a propriedade background quanto o background-color, a escolha é livre.

15

2.4. Um pouco de cor sempre é bom Casa do Código Figura 2.6: O fundo branco da div em cima do fundo da página.

Podemos também mudar a cor dos links existentes no nosso conteúdo. Desde que eles mantenham uma diferença em relação ao resto do texto, essa mudança não prejudica a usabilidade ou o visual da página. Podemos trocar o tom de azul (cor padrão dos navegadores para links), e mudar os links para negrito, o que é bastante simples.

.place a {

color: #2C88A7;

font-weight: bold;

}



Figura 2.7: O novo estilo dos links.

Assim, os 3 links da página estão com um estilo próprio, mas deixamos um link importante de fora, o da própria página. É comum adicionar um link ao título da página que leve para a página principal do site, que no nosso caso é a própria página em que estamos. Vamos mudar o código HTML do nosso h1 para adicionar um link.

<h1><a href=''>O que eu mais gosto em São Paulo</a></h1> 16

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

E é claro, uma cor especial para este link:

h1 a {

color: #7E9F19;

text-decoration: none;

}



Figura 2.8: Agora o nosso título está verde!

Reutilizamos a cor verde das bordas tracejadas que vimos anteriormente, e ti ramos o sublinhado do texto do link. Também podemos mudar a cor do link para quando se posicionar o cursor em cima dele, utilizando um pseudo-seletor do estado hover do elemento.

h1 a:hover {

background-color: #7E9F19;

color: #FFF;

}

Agora, ao se posicionar o cursor do mouse no título da página, o verde que antes estava no texto irá preencher o fundo do elemento, e o branco irá tomar o lugar do verde do texto. Quando o cursor sair de cima do título da página, o link voltará ao seu estilo tradicional.

Além do hover, existem pseudo seletores que podem ser utilizados para adici onar um estilo especíco a outros estados de links ou outros elementos, e os mais tradicionais são:

• :focus, para elementos que estejam focados pelo teclado, como links ou cam pos de um formulário;

• :visited, para endereços já visitados pelo usuário;

• :active, para o instante onde o elemento está sendo pressionado pelo mouse. 17

2.5. Primeiro contato com imagens Casa do Código 

Figura 2.9: O efeito de :hover do título.

Se for necessário trocar as cores do site, seria o caso de alterar apenas as ocor rências de color, background, background-color e talvez de border.

**2.5 Primeiro contato com imagens**

Um dos elementos clássicos que não utilizamos até agora é o img, utilizado para exibir imagens que fazem parte do seu conteúdo - diferentes de imagens utilizadas no estilo da sua página, que são aplicadas utilizando a propriedade background via CSS. Vamos dar uma olhada nisso enquanto adicionamos algumas fotos na descrição de cada um dos lugares de São Paulo que estamos falando. Vamos adicionar uma tag img entre cada h2 e os parágrafos de cada um dos elementos .place.

<div class='place'>

<h2>Passear na Avenida Paulista!</h2>

<img src='images/paulista.jpg' alt="O céu da avenida Paulista, foto por http://www.flickr.com/photos/jairo\_abud">

<!-- os parágrafos vem aqui... -->

</div>

<div class='place'>

<h2>Os bares da Vila Madalena</h2>

<img src='images/piola.jpg' alt="A varanda do Armazém Piola, foto por Fernando Moraes">

<!-- os parágrafos vem aqui... -->

</div>

<div class='place'>

<h2>O Parque do Ibirapuera</h2>

<img src='images/ibirapuera.jpg' alt="O Parque do Ibirapuera, por http://www.flickr.com/photos/soldon/">

<!-- os parágrafos vem aqui... -->

</div>

18

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

O atributo src deve conter o caminho para o arquivo da imagem, enquanto o alt é o um texto alternativo para a imagem caso ela não seja carregada pelo nave gador (no caso do caminho estar errado ou algum problema na rede do usuário). Este mesmo texto pode vir a ser usado no atributo title, para exibir um *tooltip* na imagem, mas deixamos ele de lado para manter o exemplo simples.

Enquanto o navegador se encarrega de descobrir a largura e a altura da ima gem, também podemos xar esses valores com os atributos width e height - caso o tamanho real da imagem seja diferente, o navegador irá redimensionar a imagem conforme o necessário. Fixar a altura e largura da imagem no HTML também é útil para que o navegador já “reserve” os pixels necessários para exibir a imagem, evi tando que o conteúdo seja empurrado para os lados conforme é feito o *download* da imagem, evitando o chamando *repaint* da página. Caso queira praticar isso, nossas imagens estão com 156px de largura e de altura.

**Atenção aos tamanhos**

Imagens de tamanho muito grande são um dos principais responsáveis por problemas de performance na web. Mesmo se você xar um tamanho para as suas imagens que seja menor do que o seu tamanho real, o arquivo ori ginal será transferido pela rede e pode causar uma certa lentidão caso as suas imagens sejam pesadas, então que atento ao tamanho das imagens de fundo, ícones e fotos das suas páginas.

Por isso, não deixe de comprimir as suas imagens para economizar diversos bytes de tráfego. Caso você utilize o Mac OS X, aconselho a usar o **Image Optim** (http://imageoptim.com/) para esta tarefa

19

2.5. Primeiro contato com imagens Casa do Código 

Figura 2.10: a foto da Avenida Paulista, mas parece que quebramos algo no nosso layout.

Criamos um problema no layout da nossa página com essas imagens. O img e os parágrafos não cam um do lado do outro, e o texto só aparece logo abaixo da foto. Isto ocorre porquê a imagem é um elemento inline e o elemento p é block. Vamos entrar em mais detalhes a respeito disso na seção 6.3, mas o que precisamos agora é uma solução para a nossa página. Primeiro, vamos aplicar alguns estilos às imagens e utilizar a propriedade float para alinhar os elementos horizontalmente.

.place img {

border: 1px solid #7E9F19;

float: left;

margin: 10px 10px 0 0;

padding: 2px;

}

Isto vai adicionar uma borda verde na imagem, e um espaçamento entre ela e os demais elementos - como as bordas da div e o h2 acima dela. Apesar de solucionar o problema de posições dentro da div sobre a Avenida Paulista, nos demais elementos a imagem vaza para fora da div - um dos problemas clássicos de se usar a proprie dade float. Vamos resolver isto adicionando um elemento vazio com a classe clear logo após os parágrafos. Ele irá garantir que as imagens continuem dentro do seu respectivo div e que eles terão a altura adequada para conter todo o seu conteúdo.

<div class='place'>

<h2>Passear na Avenida Paulista!</h2>

20

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

<!-- a foto e os parágrafos aqui... -->

<div class='clear'></div>

</div>

<div class='place'>

<h2>Os bares da Vila Madalena</h2>

<!-- a foto e os parágrafos aqui... -->

<div class='clear'></div>

</div>

<div class='place'>

<h2>O Parque do Ibirapuera</h2>

<!-- a foto e os parágrafos aqui... -->

<div class='clear'></div>

</div>

E o CSS da classe .clear:

.clear {

clear: both;

}

Este é um truque bastante conhecido e utilizado, mas não sou muito fã dele de vido ao elemento vazio extra que é adicionado ao HTML, sem nenhum valor semân tico, que existe apenas para resolver um problema do estilo da página. Vamos voltar a este assunto sobre o float e aprender uma solução mais prática e elegante para isto.

21

2.6. Adicionando elementos secundários Casa do Código 

Figura 2.11: A foto com seu estilo, posicionada corretamente.

**2.6 Adicionando elementos secundários**

Bem, já temos todo o conteúdo da nossa página no seu devido lugar com um pouco de CSS para deixar tudo mais apresentável. Precisamos agora de mais alguns elemen tos adicionais para complementar a nossa página, como um rodapé e uma barra la teral com mais conteúdo, no caso uma lista adicional de outros lugares de São Paulo, mas sem descrições ou fotos como os que temos atualmente. Vamos primeiro ao rodapé, o mais simples dos dois, com o seguinte código, a ser adicionado no nal do elemento .container.

<div class='footer'>

<p>Parte do livro "HTML5 e CSS3: Domine a web do futuro."</p> </div>

E um pouco de CSS, é claro.

.footer {

font-size: 12px;

text-align: center;

}

O rodapé terá uma fonte menor e será centralizado. Diferente da margem auto mática que utilizamos no .container anteriormente, estamos centralizando o texto, e não o elemento do rodapé inteiro.

22

Casa do Código Capítulo 2. Os primeiros passos com o nosso site

O próximo da nossa lista é a barra lateral. O HTML dela deve ser adicionando antes do rodapé que acabamos de criar.

<div class='sidenote'>

<h3>E existem muitos outros lugares interessantes na cidade...</h3> <ol>

<li>O Mercado Municipal.</li>

<li>A Sala São Paulo.</li>

<li>Os estádios dos principais times de futebol da cidade.</li> <li>Diversos museus, como o Memorial da América Latina, Museu da Língua Portuguesa e o Museu do Ipiranga.

</li>

<li>E mais!</li>

</ol>

</div>

Mais uma div, um título em um h3 e uma lista ordenada, a tag ol, com alguns itens a mais. E é claro que este elemento acompanha um pouco de CSS.

.sidenote {

background-color: #FFFBE4;

border: 1px solid #C9BC8F;

padding: 10px;

}

.sidenote h3 {

font-size: 14px;

margin-top: 0;

}

.sidenote ol {

font-size: 12px;

}

23

2.6. Adicionando elementos secundários Casa do Código Figura 2.12: O estilo da nossa barra lateral.

Um pouco de cor e espaçamento, da mesma forma que vimos anteriormente, e alguns ajustes do tamanho das fontes já que este elemento não deve chamar mais atenção do que o nosso conteúdo principal. E por ser uma barra *lateral*, ela não deveria pegar toda a largura disponível, e se encaixar ao lado do nosso conteúdo principal. Outro caso de uso para o float, como utilizamos no caso das imagens. Primeiro, precisamos colocar todo os elementos com a classe place dentro de uma div nova para separá-los do resto do conteúdo, e então adicionar o estilo necessário para este elemento car sempre a esquerda da página.

<div class='places'>

<!-- conteúdo principal vem aqui... -->

</div>

.places {

float: left;

width: 660px;

}

Agora, o conteúdo principal terá 660px de largura, e o restante (dos 960px que denimos para a largura total da nossa página) ca disponível para a barra lateral. Então vamos atualizar o estilo dela para posicioná-la no seu lugar de direito.

.sidenote {

background-color: #FFFBE4;

border: 1px solid #C9BC8F;

float: right;

padding: 10px;

width: 260px;

}

24

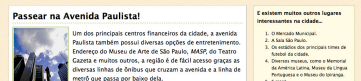
Casa do Código Capítulo 2. Os primeiros passos com o nosso site

Com a propriedade float com o valor right, a barra lateral vai car sempre no lado direito, enquanto o conteúdo está a esquerda. E a largura xa, que é menos do que o espaço restante da página, deixa um espaço a mais entre os elementos e garante que as suas bordas não vão se juntar e deixar o visual um pouco confuso.

Só o que falta é colocar o rodapé de volta no seu lugar. Com o uso do float nos outros elementos, ele agora foi parar abaixo da barra lateral, enquanto deveria estar abaixo de tudo, como antes. Simplesmente adicionamos a classe clear ao elemento do rodapé, e voilà!

<div class='footer clear'>

<p>Parte do livro "HTML5 e CSS3: Domine a web do futuro."</p> </div>

Figura 2.13: A barra lateral, no seu devido lugar.

**2.7 Faça para sua cidade também!**

Foi simples, não? Após apenas algumas páginas, já temos um exemplo simples, ele gante e funcional de um layout com uma pequena pitada de CSS. Vamos apren der muito mais nos próximos capítulos, inclusive técnicas que poderiam deixar esse exemplo muito mais interessantes.

Caso você esteja começando nesse mundo de HTML e CSS, que tal implementar uma página similar para a sua cidade? Coloque-a no ar e eu carei muito contente de fazer um link para ela. Apenas me conte por e-mail o endereço da sua página. Você pode enviar para casadocodigohtmlcss@gmail.com.

25

Capítulo 3

HTML5: o que mudou?

O surgimento do HTML5 mudou muitas coisas no mundo do desenvolvimento web, com novos elementos, novas funcionalidades e diversas outras novidades que pos sibilitam experiências melhores e integrações que antes eram apenas desejos e so nhos dos desenvolvedores por aí. Mesmo com algumas funcionalidades ainda em processo de denição, já podemos aproveitar diversas das novidades que o HTML5 trouxe para o mundo.

Vamos passar por algumas delas, inclusive algumas que são referente ao código que escrevemos: novos elementos, mudanças de sintaxe e atributos personalizados. Mas não deixe de conferir no capítulo 11 sobre por onde aprender mais a respeito de outras novidades.

**3.1 Escrevendo menos e fazendo mais**

Uma das minhas mudanças prediletas é como a tarefa de escrever HTML cou mais simples. Diversos pontos pequenos se tornaram opcionais ou desnecessários e o que

3.1. Escrevendo menos e fazendo mais Casa do Código

antes era repetitivo deixou de ser parte da nossa rotina e como consequência, o nosso código se torna mais simples e fácil de se ler.

Um ótimo exemplo disso é o Doctype do HTML5, a instrução que informa aos navegadores como ele deve processar o código HTML, que cou bastante simples em relação ao seus antecessores. No HTML 4 ou no XHTML 1, a linha do Doctype é algo assim:

*<!- DTD do HTML 4, em modo "strict".*

*<!DOCTYPE HTML*

*PUBLIC "-//W3C//DTD HTML 4.01//EN"*

*"http://www.w3.org/TR/html4/strict.dtd">*

<!-- DTD transitional do XHTML -->

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

Dicilmente alguém vai conseguir lembrar de tudo isso, e geralmente deixamos a responsabilidade de escrever a linha do Doctype para ferramentas de geração de projetos ou de código. Agora, ele está tão pequeno que é bem fácil escrever ele de cor.

*<!DOCTYPE html>*

Não existe mais URL de dtd ou tipo de Doctype para se preocupar. Bastante simples!

Além disso, as tags de link e script também emagreceram um pouco. Agora, para referenciar arquivos de CSS e JavaScript, você não precisa mais informar o atri buto type. Caso ele não esteja presente, o navegador irá assumir que o arquivo é do tipo text/css ou text/javascript.

<!-- O que antes era assim... -->

<link rel="stylesheet" href="normalize.css" type="text/css" /> <script src="prefixfree.min.js" type="text/javascript"></script>

<!-- ...pode ser escrito assim -->

<link rel="stylesheet" href="normalize.css">

<script src="prefixfree.min.js"></script>

Se você comparar os exemplos anteriores da tag link, pode notar que a barra no nal da tag também não está mais presente - mais um detalhe que se tornou opcional no HTML5. Tags que não precisam ser fechadas, como link, input, img, br e meta, não precisam mais terminar em />, apenas >.

28

Casa do Código Capítulo 3. HTML5: o que mudou?

**3.2 Atributos personalizados**

Em alguns casos precisamos adicionar metadados ou informações extras no nosso HTML, mas não queremos depender de elementos adicionais que não possuem valor de verdade a nossa página. Um exemplo muito comum, é fazer com que alguma biblioteca leia uma tag a procura de alguns atributos especícos, como por exemplo, um atributo indicando que a requisição de um link deve ser feita assincronamente via AJAX.

<a href='posts/2' remote='true'>Apagar</a>

O atributo remote não existe na denição da tag a. Esta abordagem funcionaria em páginas de HTML 4, por exemplo, mas o seu código estaria inválido de acordo com a ferramenta de validação do W3C.

Uma forma de resolver isto são com atributos personalizados, que fazem parte do domínio do seu projeto e ao mesmo tempo mantém o seu código válido. Por exemplo, para denir um link que irá apagar um post do seu blog, podemos utilizar algumas congurações via atributos personalizados com o prexo data-.

<a href='posts/2' data-remote='true' data-method='delete' data-confirm='Você tem certeza disto?'>Apagar</a>

Desta forma, o seu JavaScript pode ler os atributos de data-\* e executar o có digo necessário para apagar o post, pedindo uma conrmação do usuário com uma mensagem personalizada.

**3.3 Tags novas para elementos antigos**

No campo da semântica existe um problema complicado de se lidar, que é o uso excessivo da tag div. Um verdadeiro coringa do HTML, que por ser um elemento genérico, acabamos utilizando para denir quase toda a estrutura das nossas páginas, desde o cabeçalho ao rodapé.

Embora isso nunca tenha impedido ninguém de colocar um site de sucesso no ar, acaba gerando um certo ruído e uma necessidade de se utilizar classes CSS para iden ticar o papel de cada div dentro do que criamos. Com isso, o HTML5 traz alguns elementos para suprir essa falta de tags mais semânticas e descritivas para alguns elementos clássicos em diversos cenários, como cabeçalhos, menus de navegação e áreas de conteúdo secundário.

Algumas das alternativas existentes são:

29

3.3. Tags novas para elementos antigos Casa do Código

• section - utilizado para representar uma seção genérica, geralmente com um cabeçalho próprio e o seu conteúdo;

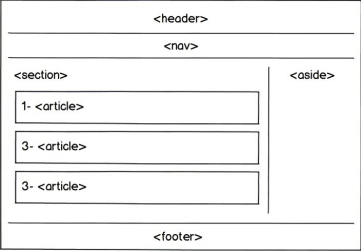
• nav - representação de um bloco principal de links de navegação - nem todo grupo de links deve ser tratado como um nav;

• aside - a tag aside pode ser utilizada para representar uma seção de conteúdo secundário ou auxiliar a outro pedaço de maior importância. Citações, links de referência ou notas adicionais, por exemplo;

• header - referente ao cabeçalho de uma seção especíca (ou da própria pá gina), contendo títulos, introduções e outros elementos similares;

• footer - o rodapé referente a um bloco de conteúdo;

• article - nalmente, o article identica o conteúdo em si, como uma no tícia de um portal, um post em blog ou um comentário em uma lista de co mentários.

Figura 3.1: O papel de cada elemento novo em uma página completa.

30

Casa do Código Capítulo 3. HTML5: o que mudou?

Um ponto crucial destas mudanças é que podemos denir diversas seções inde pendentes entre si, cada uma com a sua hierarquia própria. Por exemplo, podemos ter 2 elementos h1, cada um em sua seção, ou podemos criar um header para o ca beçalho de um blog, com o título e links de navegação, e cada post, devidamente criado em um article, ter um header com o título do post e a data em que ele foi publicado, por exemplo. Isso pode ser bastante útil para se arquitetar páginas mais modulares, para melhorar a qualidade e a facilidade de manutenção do nosso código.



Figura 3.2: Um mockup do exemplo descrito - a tag ’<header>’ pode aparecer em 2 seções diferentes.

Com base nesta lista ca fácil de pensar em diversos lugares onde uma tag div foi utilizada com alguma classe para identicar o seu propósito. O que acha de praticar um pouco esses elementos novos na nossa página sobre São Paulo?

**3.4 Refatoração da página de São Paulo**

Nossa página *O que eu mais gosto em São Paulo* tem alguns pontos que podemos substituir o uso da tag div por alguns dos elementos novos do HTML5. Não tere mos nenhuma mudança visual na página, apenas a semântica do código escrito irá mudar. A diferença no tamanho do arquivo ou na quantidade de código escrito é

31

3.4. Refatoração da página de São Paulo Casa do Código

insignicante, então não vamos nos preocupar com isso.

Vamos começar pelo nosso conteúdo principal, os 3 lugares de São Paulo que são descritos com a classe .place. Podemos alterar esses elementos para utilizar a tag article, removendo a necessidade da classe para identicar estes elementos.

<article>

<h2>Passear na Avenida Paulista!</h2>

<img src='images/paulista.jpg' alt="O céu da avenida Paulista, foto por http://www.flickr.com/photos/jairo\_abud">

<p>

Um dos principais centros financeiros da cidade, a avenida Paulista também possui diversas opções de entretenimento. Endereço do Museu de Arte de São Paulo, <em>MASP</em>, do Teatro Gazeta e muitos outros, a região é de fácil acesso graças as diversas linhas de ônibus que cruzam a avenida e a linha de metrô que passa por baixo dela. </p>

<p>

A Avenida Paulista sempre é assunto. O que será que estão falando a respeito no <a href='https://twitter.com/#!/search/Av.%20Paulista' target='\_blank'>Twitter</a>?

</p>

</article>

E claro, alterar o nosso CSS para reetir esta mudança, trocando onde antes usá vamos a classe place pela tag article.

O próximo lugar que podemos alterar é o elemento com a classe .sidenote, a nossa barra lateral, que possui um conteúdo adicional ao que mexemos anterior mente. Um ótimo candidato a se tornar um aside.

<aside>

<h3>E existem muitos outros lugares interessantes na cidade...</h3> <ol>

<li>O Mercado Municipal.</li>

<li>A Sala São Paulo.</li>

<li>Os estádios dos principais times de futebol da cidade.</li> <li>Diversos museus, como o Memorial da América Latina, Museu da Língua Portuguesa e o Museu do Ipiranga.</li>

<li>E mais!</li>

</ol>

</aside>

32

Casa do Código Capítulo 3. HTML5: o que mudou?

Sem deixar de alterar o CSS, trocando as referências a classe .sidenote pela tag aside.

aside {

background-color: #FFFBE4;

border: 1px solid #C9BC8F;

float: right;

padding: 10px;

width: 260px;

}

aside h3 {

font-size: 14px;

margin-top: 0;

}

aside ol {

font-size: 12px;

}

E for nal, o uso da classe .footer no nosso rodapé é outro caso óbvio de mu dança.

<footer class='clear'>

<p>Parte do livro "HTML5 e CSS3: Domine a web do futuro."</p> </footer>

footer {

font-size: 12px;

text-align: center;

}

Existem outros pontos que poderíamos mudar no nosso HTML: substituir a div com a classe .places por um section, e colocar um header em volta do título da nossa página h1, mas não precisamos chegar a tanto.

Nossos usuários não notariam esta mudança, mas estamos agregando mais se mântica ao nosso código e facilitando a leitura dele, para manutenções futuras.

33

3.5. Seja pragmático Casa do Código

**Preenchendo alguns buracos**

Quando um navegador não possui suporte a uma tag desconhecida para ele, o elemento é exibido na página, porem nenhum estilo é aplicado. É assim que versões antigas do Internet Explorer, por exemplo, tratam elementos *no vos* como o section e o header. Se você for utilizar algumas dessas tags e precisa dar suporte a navegadores que ainda não entendem elementos no vos do HTML5, a solução é utilizar um pouco de JavaScript para forçar a aplicação de estilo pelo navegador.

O script mais utilizado para isto é o **html5shiv** (https://github.com/aFarkas/ html5shiv/), Que além de tratar a aplicação de estilos nesses elementos, tam bém trata alguns problemas relacionados a impressão da página que utilize tags do HTML5.

**3.5 Seja pragmático**

Enquanto este mundo de novas possibilidades e tags a nossa disposição pode pa recer a solução, é muito fácil se perder no mérito das discussões e denições do tipo de *"isto aqui deveria ser um article ou um section?”*. Continuar utilizar a tag div não causa mal nenhum, e você sempre pode voltar no seu código depois e tro car alguma tag por outra que você considere mais adequada. Divya Manian fala bastante sobre os problemas que essa busca incessante por semântica pode causar em um post na Smashing Magazine http://coding.smashingmagazine.com/2011/11/ 11/our-pointless-pursuit-of-semantic-value/. O ponto importante é não perder o foco do que realmente precisamos fazer.

34

Capítulo 4

O que todo desenvolvedor precisa saber sobre CSS

Entender a sintaxe e as propriedades do CSS é vital, porém mais importante do que isso é aprender alguns truques e técnicas para resolver problemas do dia a dia. Só saber a linguagem não é o suciente, é preciso saber como usá-la e quais são as suas opções quando precisar colocar a mão na massa.

**4.1 A incompatibilidade dos browsers e a razão dos resets de CSS**

Todos os navegadores, das primeiras versões do Internet Explorer aos navegadores de celulares e tablets, possuem estilos padrões que são aplicados em todas as páginas, como o negrito da tag strong, a borda do legend e os tamanhos de fonte diferentes do h1 ao h6. E é claro que existem diferenças entre os padrões de um navegador para outro. Algumas dessas diferenças, por menores que possam ser, costumam afetar o

4.1. A incompatibilidade dos browsers e a razão dos resets de CSS Casa do Código

nosso trabalho, impactando no resultado nal das nossas páginas quando visuali zadas em diferentes navegadores e soluções especícas para estes cenários precisam ser utilizadas.

Para entender melhor esta situação, podemos testar um pouco de HTML, sem estilos, e ver como que ele será exibido no Google Chrome e no Firefox, por exemplo.

<h1>Título</h1>

<fieldset>

<legend>Campos do formulário</legend>

<abbr title="Hypertext Markup Language">HTML</abbr>

<p><label>Faça a sua pesquisa:</label><input type="search"></p> <textarea>Um pouco de texto</textarea>

</fieldset>



Figura 4.1: O resultado no Firefox 9.0.1



Figura 4.2: O resultado no Chrome 19.0

Conseguimos notar leves diferenças, que podem impactar o nosso trabalho nal. As bordas do fieldset, a ausência do sublinhado no elemento abbr no Chrome e as diferenças dos campos de busca e de texto.

36

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

Como qualquer problema recorrente no mundo de desenvolvimento, soluções já foram criadas para não nos preocuparmos com coisas assim e focarmos no traba lho de verdade. Os arquivos de *reset*, como são chamados, possuem uma gama de regras para alinhar os navegadores em um mesmo patamar de estilo, seja corrigindo problemas ou resolvendo inconsistências.

Talvez o mais famoso dos *resets* seja o Escrito pelo Eric Meyer, que criou a versão *1.0* em 2008 - bastante tempo, não? - sua última atualização foi feita em 2011, e você pode encontrá-la no próprio blog do seu criador, http://meyerweb.com/eric/tools/ css/reset/.

Os desenvolvedores do Yahoo! também criaram um *reset* sólido, parte do *Yahoo! User Interface Library*, disponível no site http://yuilibrary.com/. Um ponto negativo do *reset* do Yahoo! é que ele remove alguns padrões que eu considero importante em alguns elementos, como o negrito da tag strong e a marcação lateral de itens em uma lista.

**Normalize.css**

O **Normalize.css** (http://necolas.github.com/normalize.css/), é o *reset* que esta mos utilizando neste livro e toma uma abordagem diferente a este problema. No lugar de sobrescrever diversas propriedades para denir um novo padrão de estilo para os navegadores, ele apenas adequa os pontos diferentes, preservando diversos estilos aplicados pelos navegadores, além de aproveitar para denir algumas melho rias sutis, removendo o outline de links e melhorando a formatação de elementos pre, por exemplo. [3]

O código fonte do Normalize é muito bem escrito e documentado, o que lhe permite remover partes dele que não te interesse ou entender melhor o que ele faz por debaixo dos panos. Conra o exemplo de código a seguir:

/\*

\* 1. Improves usability and consistency of cursor style between image-type 'input' and others

\* 2. Corrects inability to style clickable 'input' types in iOS \* 3. Removes inner spacing in IE7 without affecting normal text inputs \* Known issue: inner spacing remains in IE6

\*/

button,

input[type="button"],

37

4.2. Compreendendo o Box model Casa do Código

input[type="reset"],

input[type="submit"] {

cursor: pointer; /\* 1 \*/

-webkit-appearance: button; /\* 2 \*/

\*overflow: visible; /\* 3 \*/

}

Cada propriedade é devidamente documentada, e sabemos quais versões dos navegadores precisa das correções. Caso você não queria se preocupar com versões do Internet Explorer abaixo da 8, por exemplo, você pode remover do código fonte do Normalize as regras especícas destas versões.



Figura 4.3: O nosso HTML de teste, agora com o Normalize.css

**Não saia de casa sem o seu**

Independente de qual *reset* você for usar, o mais importante é estar usando um, e faça disso parte do seu processo de desenvolvimento ao participar de novos projetos. Isto não tira a responsabilidade de testar manualmente as páginas em diversos navegadores, mas impede que alguns problemas relacionados a posicionamento, ta manhos e fontes apareçam do nada e você tenha que dedicar tempo a esses problemas ao invés de se dedicar ao que precisa ser criado.

**4.2 Compreendendo o Box model**

Não importa quais cores, imagens de fundo ou posição você adicionar em um ele mento, ele sempre será tratado como uma caixa. Isto pode não ser perceptível a primeira vista, mas basta começar a usar um inspetor de elementos como o Firebug ou o console do WebKit e você irá se acostumar com esta abordagem. O **Box model** é como as propriedades de CSS compõem as dimensões, onde além do width e do

38

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

height, as propriedades border e padding também inuenciam no resultado nal, mas de uma forma um tanto quanto confusa. Considere o código a seguir.

.button {

border: 1px solid #999;

height: 26px;

padding: 5px 15px;

width: 90px;

}



Esta classe button, ao ser aplicada em um elemento qualquer, irá ocupar 122px de largura e 38px de altura na sua página. Isto acontece por que os valores de padding e border são somados a largura e altura denidas, fazendo com que as propriedades width e height sejam responsáveis por denir apenas a largura e a altura do seu conteúdo, e não do elemento como um todo. Além da confusão que pode ser gerada ao se calcular os tamanhos de antemão, isso torna bastante complicado o uso de valores com porcentagem nestas propriedades - um elemento com 100% de width e 2px de bordas cará maior do que os 100% denidos. Complicado, não?

Outro exemplo para ilustrar este problema: ambos elementos possuem o mesmo valor de width, mas o valor do padding do segundo o deixa com 300px de largura.

.box {

background-color: LimeGreen;

width: 250px;

}

.box-with-padding {

background-color: LightBlue;

padding: 0 25px;

width: 250px;

}

<div class='box'>elemento sem padding</div>

<br>

<div class='box-with-padding'>elemento com padding</div> 39

4.2. Compreendendo o Box model Casa do Código 

Figura 4.4: O padding do segundo elemento expande a sua largura.

Esta regra do Box model aplicar estes valores a apenas o conteúdo e não utili zar como limites xos do elemento como um todo pode complicar diversos casos de diagramação de elementos em páginas complexas, mas é possível alterar este com portamento e não car recalculando larguras e alturas baseado em bordas e espaça mentos. A propriedade box-sizing, uma das novidades do CSS 3, permite alterar esta regra de content-box para o valor border-box, que força o navegador a respei tar estes limites e ocupar o espaço interno do elemento com os valores de border e padding, no lugar de expandir o elemento.

Desta forma, as duas denições abaixo irão gerar elementos com a mesma lar gura

\* {

box-sizing: border-box;

}

.box {

background-color: LimeGreen;

width: 250px;

}

.box-with-padding {

background-color: LightBlue;

padding: 0 25px;

width: 250px;

}

40

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS 

Figura 4.5: O box-sizing tornando ambos elementos do mesmo tamanho.

**4.3 Utilizando pseudo elementos**

Existem casos onde as tags que utilizamos para denir o nosso conteúdo não são o suciente para estruturar os elementos de interface que desejamos. Seja para tratar problemas de oat, denir bordas adicionais ou algo similar, muitas vezes acabamos dependendo de elementos vazios para resolver estes problemas. Por exemplo, ao trabalhar com bordas arredondadas sem o uso de CSS 3, já vi (e escrevi), coisas assim, onde o span é utilizado para criar as bordas do lado direito de um botão.

<div class='button'>

<a href='#'>Clique aqui</a>

<span></span>

</div>

Já foi muito comum utilizar soluções assim para trabalhar com sombras, bordas arredondadas, letras iniciais de títulos e outros detalhes de interface que não conse guimos criar apenas com o mínimo de código HTML. Mas existem alternativas bas tante interessantes para isto, utilizamos apenas CSS. São *pseudo elementos*, que não existem no nosso código HTML, mas podem receber estilos especícos e auxiliar a estruturar elementos mais complexos nas nossas interfaces. Os mais tradicionais deles são os elementos ::before e ::after, que vivem antes e depois do **conteúdo** de uma tag (e não antes e depois da tag em si), e estão disponíveis para todas as tags que adicionarmos ao body da nossa página.

**4.4 Desenhando uma faixa com “*before” e “*after”**

Vamos criar uma faixa que dá a ilusão de que ela contorna um section que o con tém, e os elementos ::before e ::after irão fazer o trabalho pesado para a gente. Começamos com o mínimo de HTML necessário, uma tag section e um título para a faixa.

41

4.4. Desenhando uma faixa com “*before” e “*aer” Casa do Código

<section>

<h1>Um exemplo de pseudo elementos</h1>

</section>

Vamos adicionar um pouco de CSS para denir a largura do section e a posição do título.

section {

border: 1px solid #000;

height: 100px;

margin: 40px auto;

width: 400px;

}

h1 {

font-size: 1.2em;

text-align: center;

}



Figura 4.6: O elemento section com 400px de largura.

Vamos começar a transformar o nosso h1 na faixa que queremos criar. Além da cor, vamos posicionar as laterais da faixa para fora da seção.

h1 {

background-color: #990000;

color: #FFF;

font-size: 1.2em;

left: -10px;

padding: 5px 0;

position: relative;

text-align: center;

width: 420px;

}

42

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS 

Figura 4.7: O lado frontal da faixa vermelha.

Agora, temos 10px de cada lado da faixa *vazando* pela seção, só precisamos criar o efeito da faixa contornando o section, utilizando pseudo elementos. Primeiro, vamos criar este efeito para o lado esquerdo da faixa, utilizando o ::before.

h1::before {

border: 5px solid #7C0000;

content: "";

left: 0;

position: absolute;

top: -10px;

}



Figura 4.8: Desenhamos um quadrado no canto da faixa com o pseudo elemento ’::before’.

Fazemos referência ao before utilizando a mesma sintaxe de pseudo classes que utilizamos para links, com o : separando os elementos.

Utilizando o position junto de coordenadas exatas com as propriedades %le e top, conseguimos posicionar o elemento exatamente no canto superior esquerdo da faixa e com uma borda de 5px, desenhamos um quadrado de 10px

A propriedade content é utilizada para denir o conteúdo do elemento - já que não estamos adicionando o elemento pelo HTML - e neste caso não precisamos de conteúdo algum. Para conseguir o efeito que queremos, precisamos transformar o

43

4.4. Desenhando uma faixa com “*before” e “*aer” Casa do Código

quadrado que criamos em um triângulo, só precisamos de um ajuste nas cores das bordas.

h1::before {

border-color: transparent #7C0000 #7C0000 transparent; border-style: solid;

border-width: 5px;

content: "";

left: 0;

position: absolute;

top: -10px;

}



Figura 4.9: O quadrado se tornou uma aba da faixa.

Trocando algumas das cores para transparent, conseguimos desenhar um tri ângulo no lugar do quadrado vermelho. Para duplicar este efeito no outro lado da faixa, utilizando o ::after, só precisamos inverter alguns detalhes - o que era feita com a propriedade left será feito com a right, e as posições da lista de cores tam bém precisa ser modicada, desta forma.

h1::after {

border-color: transparent transparent #7C0000 #7C0000; border-style: solid;

border-width: 5px;

content: "";

position: absolute;

right: 0;

top: -10px;

}

44

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS 

Figura 4.10: A faixa completa, utilizando ’::before’ e ’::aer’.

Feito! Criamos o efeito da faixa contornando a seção, sem precisar de novos elementos no HTML.

**“:after” ou “::after"?**

É comum ver por ai o uso dos pseudo elementos before e after em duas formas diferentes: prexada com dois pontos (:before) ou quatro pontos (::before). Você sabe a diferença entre eles?

Efetivamente, nenhuma. A especicação do CSS 2 criou a versão com dois pontos, mas ela foi mudada no CSS 3 para ser diferente das denições de pseudo seletores. Navegadores modernos possuem suporte aos dois forma tos, enquanto algumas versões antigas só entender o formato de dois pontos do CSS 2.

**4.5 Decorando mensagens**

Outra utilidade dos pseudo elementos é criar formas de botões que antes só era pos sível com o uso de imagens de fundo. Usando uma lógica similar a utilizada no exemplo da faixa, podemos criar um balão de ajuda com um indicação sobre o que ele é a respeito. Imagine o seguinte parágrafo, sendo usado em um formulário de cadastro de um site de um concurso qualquer:

<p class='help'>

Preencha o campo com um e-mail válido, assim poderemos entrar em contato com você para informar o resultado do nosso concurso. </p>

Podemos adicionar um pouco de estilo e transformar este parágrafo em uma caixa de ajuda.

45

4.5. Decorando mensagens Casa do Código

.help {

background-color: #F1EFE6;

border: 1px solid #D3CDAE;

font-size: 0.9em;

padding: 10px;

position: relative;

width: 300px;

}



Figura 4.11: O estilo da caixa de ajuda.

Considerando que este elemento está ao lado de um campo de e-mail, podemos simular uma seta na lateral da caixa de ajuda apontando para o campo do formulário, demonstrando a relação entre a mensagem e o campo que o usuário irá preencher. Utilizar o ::before para criar esta seta é uma ótima opção.

.help::before {

border-color: transparent #D3CDAE transparent transparent; border-style: solid;

border-width: 14px;

content: "";

left: -28px;

margin-top: -14px;

position: absolute;

top: 50%;

}



Figura 4.12: A seta lateral da caixa da mensagem de ajuda.

46

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

Utilizamos a mesma lógica das bordas da faixa que criamos anteriormente para transformar este elemento em um triângulo, e denimos a posição exata dele com a combinação da propriedade position e das coordenadas com top e left. Mas parte do truque é a propriedade top com 50% e a margin-top negativa, o que força o elemento a car sempre centralizado, independente da altura do parágrafo.

**4.6 Criando conteúdo através de CSS**

Além de serem bastante úteis para desenhar novos elementos na página, estes pseudo elementos também podem gerar conteúdo para complementar seus elementos de referência. A propriedade content aceita pedaços de texto para preencher o seu conteúdo - mas você não pode adicionar código HTML, infelizmente.

Talvez o exemplo mais clássico dessa funcionalidade é o de exibir o endereço de links da página logo após o seu texto - uma ótima pedida para estilos de impressão, que veremos logo mais neste capítulo. Mas também podemos adicionar ícones e ou tros tipos de elementos decorativos para complementar o nosso conteúdo. Exemplo, em um texto informativo não tão relacionado, conseguimos adicionar a indicação de uma mão apontando para o texto, assim:

<p class='tip'>Você sabia que...</p>

Para posicionar uma indicação próxima ao texto, basta o uso do ::before junto de um símbolo Unicode que representa uma mão apontando o dedo.

.tip::before {

content: "\261E";

margin-right: 10px;

}



Figura 4.13: Adicionamos o símbolo unicode através da propriedade ’content’. Pronto - não precisamos de uma imagem de um ícone especíco para isso. 47

4.6. Criando conteúdo através de CSS Casa do Código

Outro caso de uso para gerar conteúdo com pseudo elementos é ao adicionar estilo a elementos de citação, como o blockquote. É bastante simples adicionar as pas em torno do texto para indicar que aquele conteúdo se trata de uma citação. Começando com o mínimo necessário de HTML:

<blockquote>

O problema com citações na Internet é que você não pode confirmar a sua veracidade.

</blockquote>

E um pouco de CSS para formatar a citação:

blockquote {

color: #444;

font-style: italic;

}

Figura 4.14: Um bloco de citação levemente formatado.

Podemos adicionar aspas ao redor do texto, com uma fonte maior e uma cor dife rente, apenas adicionando o conteúdo necessário no ::before e ::after e ajustando o seu estilo como necessário.

blockquote::before, blockquote::after {

color: #000;

font-size: 3em;

}

blockquote::before {

content: "\201C";

}

blockquote::after {

content: "\201D";

}

48

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS Figura 4.15: As aspas adicionais via pseudo elementos.

Como o ::before e o ::after herdam as propriedades do seu elemento, preci samos sobrescrever a cor do texto e denir um tamanho de fonte (três vezes) maior.

**Um ponto chave para posicionar pseudo elementos**

Como você pode ter notado em alguns dos nossos exemplos, as proprieda des position, top e left são bastante importantes para garantir que nossos pseudo elementos estejam em seus lugares exatos. Não deixe de conferir a seção 6.4, onde iremos conferir em detalhes o uso destas propriedades.

**4.7 Arquitete o seu CSS para o futuro**

Escrever CSS pode ser fácil, mas escrever CSS bem escrito é uma habilidade relativa mente rara e preciosa. Conforme os seus projetos vão ganhando novas funcionalida des e sofrendo mudanças visuais, você pode se encontrar em situações onde alterar os estilos de um elemento ou adicionar uma variação de algo existente pode ser uma tarefa herculana, e a duplicação de código só traz mais complicações e problemas para a sua equipe e o seu projeto. Por isso, é importante reetir sobre o código CSS escrito e tratar a sua manutenção com a mesma atenção que se cuida do código da sua aplicação, independente da linguagem que você utilize.

**Escolha bem seus seletores**

Um dos vilões clássicos de um CSS mal cuidado é a complexidade de seletores usados. Além da degradação da performance das suas páginas, compreender o es copo de uma seletor para alterar alguma propriedade pode ser complicado.

Para entender o impacto de performance é necessário entender como os navega dores processam os seletores que escrevemos. Considere o seguinte seletor: ul#nav li a.

49

4.7. Arquitete o seu CSS para o futuro Casa do Código

O navegador começará a ler o seletor de trás para frente - ao contrário do que nós estamos acostumados a ler, o que surpreende diversos desenvolvedores - e irá procurar na página *todos* os elementos da tag a. Após isso, ele irá ltrar esta lista por todos que estejam contidos em um elemento li, e então por todos que também estejam dentro de um elemento com id nav. E por m, todos em que o elemento com o id nav seja da tag ul.

Considerando que o HTML para uma estrutura dessas seria uma lista não or denada apenas com links, algo como #nav a seria uma solução mais simples para o navegador processar, e com o mesmo efeito. Ou até mesmo utilizando uma classe nav-link, por exemplo, diretamente nos elemento a dentro da lista com o id nav.

Outro ponto que gera muita discussão e confusão entre desenvolvedores: utilizar ids ou classes? Enquanto ids podem ser mais rápidos para o navegador procurar os elementos necessários na árvore do DOM, a diferença de performance é ínma, então a discussão acaba mais no assunto de preferência pessoal. Classes são ótimas para compor diversos aspectos compartilhados entre vários elementos, e ids ajudam a indicar que o seletor é especíco para um elemento único da página.

Uma regra pessoal que eu tento seguir é evitar seletores com mais de 3 partes, como header .nav a. Caso você se encontre escrevendo seletores maiores que algo assim, pare e reveja a sua estrutura para que não seja necessário escrever seletores tão grandes - seja utilizando mais classes ou reduzindo a especicidade dos seus seletores.

**Compondo padrões visuais através de classes**

Nicole Sullivan popularizou uma abordagem na escrita de CSS e composição de estilos, similar a Orientação a Objetos, que é utilizada em linguagens de programa ção. O OOCSS - Object Oriented CSS - propõe a separação da estrutura dos estilos e os *containers* de seu conteúdo.

O objetivo é escrever um código mais fragmentado e reutilizável, onde diver sas classes diferentes são utilizadas para compor o comportamento nal de um ele mento, independente da tag que ele possua ou o contexto que ele se encontre - dentro de uma lista, um formulário ou o cabeçalho da página.

Tome como exemplo os 2 botões a seguir.

50

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS 

Figura 4.16: Como você estruturaria o CSS para estes elementos.

Podemos separar os aspectos destes elementos em 4 classes diferentes: uma para denir os estilos básicos de um botão, uma para as cores do botão principal, e outras duas para controlar os tamanhos de cada um, assim:

<a href="#" class='button primary-button big-button'>Enviar mensagem</a> <a href="#" class='button small-button'>Cancelar</a>

.button {

background-color: #999999;

color: white;

display: inline-block;

font-weight: bold;

padding: 0.5em 1em;

text-decoration: none;

}

.primary-button {

background-color: #389739;

}

.big-button {

font-size: 1.1em;

}

.small-button {

font-size: 0.9em;

}

Desta forma, botões em outras páginas e que possuam detalhes diferentes (como uma cor de fundo diferente, ou um ícone posicionado ao lado do texto) podem com partilhar os estilos básicos de um botão e aplicar os seus pontos especícos com ou tras classes, como danger-button ou icon-button. Esta abordagem pode ser esten dida para menus de navegação ou conteúdos relacionados, ou uma relação de posts - o famoso **media object** dos exemplos tradicionais de OOCSS. [7]

51

4.8. Gere relatórios inteligentes e simples com os estilos de impressão Casa do Código

**4.8 Gere relatórios inteligentes e simples com os estilos de impressão**

Os navegadores nos permitem adicionar estilos especícos para quando uma página for impressa pelo usuário. Isto pode ser feito de duas maneiras: Com o atributo media na tag link ou criando um bloco de CSS dentro da diretiva @media print, conforme os exemplos a seguir.

@media print {

/\* Este CSS só será aplicado quando a página for impressa \*/ }

<!-- O atributo "media" indica que estes estilos

só devem ser aplicado para impressões -->

<link href="print.css" media="print">

A primeira forma é a mais adequada, já que evitamos uma nova requisição ao arquivo de impressão. E como os estilos de impressão costumam ser pequenos, não existem problemas em adicionar este CSS ao nal do estilo padrão da sua página.

Pode-se considerar que a impressão de páginas é algo do passado, mas depen dendo do tipo de conteúdo que você trabalha, e o seu público, isto pode ser bastante importante para o seu projeto.

Por exemplo, ao utilizar o *Airbnb* (http://airbnb.com) para alugar um aparta mento para uma viagem aos Estados Unidos, é interessante imprimir o itinerário da viagem, que possui endereços, datas e telefones úteis caso você tenha algum pro blema ao pousar no seu destino, onde você provavelmente não terá uma conexão de internet disponível logo ao chegar. Ou ao criar um trajeto pela cidade no Google Maps, você pode imprimir o mapa com o seu roteiro para ter em mãos ao dirigir.

Além disso a impressão das páginas não ca resumida a ter o conteúdo em uma folha de papel, já que hoje em dia é bastante simples de se exportar uma página da web para um arquivo .pdf utilizando o processo de impressão dos navegadores. Você consegue compartilhar páginas e e-mails ou arquivar páginas de recibos de compras ou pagamentos de serviços pela internet.

Independente do tipo de conteúdo que você estiver trabalhando, existem práti cas recomendadas para tratar os estilos de impressão, visando a legibilidade do seu conteúdo e tirando elementos desnecessários da frente de coisas mais importantes. Vamos repassar alguns deles.

52

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

**Não conte com cores**

Não podemos depender da qualidade e da gama de cores disponíveis ao imprimir a página, pois não temos controle ou informações sobre a qualidade do hardware ou da disponibilidade de tinta, então o recomendado é se manter no branco e preto de praxe. Então podemos remover todo tipo de cor ou imagem de fundo, e forçar a cor do nosso para texto, assim:

@media print {

\* {

background: transparent *!important* ;

color: #000 *!important* ;

}

}

O uso do !important é para garantir que esta regra sobrescreva qualquer outro seletor mais especíco do seu CSS padrão, para garantir que tudo se mantenha na cor preta. Além disso, é recomendado remover qualquer outra propriedade de estilo visual, como box-shadow e text-shadow, além de revisar a cor das bordas utilizadas.

@media print {

\* {

background: transparent *!important* ;

border-color: #000 *!important* ;

box-shadow: none *!important* ;

color: #000 *!important* ;

text-shadow: none *!important* ;

}

}

Ao remover a propriedade background completamente, também estamos remo vendo imagens de fundo. Caso alguma destas imagens devesse estar presente na sua versão de impressão, considere movê-la para uma tag img. E o contrário também - navegações criadas com imagens deveriam ser trocadas por texto simples ao ser impresso.

53

4.8. Gere relatórios inteligentes e simples com os estilos de impressão Casa do Código 

Figura 4.17: O cabeçalho do Last.fm, que utiliza diversas imagens para compor o logo e o menu principal.



Figura 4.18: O cabeçalho impresso, sem o logo e com as imagens do menu no lugar de um texto comum.

**Links devem ser links**

Com a ausência de cores, todos os seus links azuis, verdes ou vermelhos terão a mesma cor preta do seu conteúdo. Uma forma de diferenciar os links do texto comum é sublinhando-os, caso você tenha retirado isto do estilo dos seus links.

a {

text-decoration: underline *!important* ;

}

Outra técnica recomendada para tratar links em casos de impressão é exibir o caminho que eles possuem logo após o seu texto. Isto é possível utilizando o pseudo elemento ::after dos links, junto da função attr(), que permite ler os atributos de um elemento através de CSS.

a[href]::after {

content: " (" attr(href) ")";

}

Desta forma, todos os links com um href presente terão o seu caminho exibido entre parênteses ao seu lado. Links que não possuem uma url a seguir, como links que utilizam funções de JavaScript não devem receber este efeito, então precisamos

54

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

remover o pseudo elemento nestes casos, utilizando um seletor mais especíco que o anterior:

a[href]::after {

content: " (" attr(href) ")";

}

a[href^="javascript:"]::after, a[href^="#"]::after {

content: "";

}

**Controlando quebras de páginas**

Enquanto no navegador o nosso conteúdo existe em apenas uma página, na im pressão existe a necessidade de se quebrar o conteúdo em páginas de acordo com a conguração do usuário. Por padrão isto será feito de acordo com o tamanho da sua página, mas é possível tomar controle sobre isto e denir regras especícas para informar em qual parte do seu conteúdo a quebra de página será feita. Atu almente, existem 3 propriedades disponíveis na maioria dos navegadores para isso: page-break-before, page-break-inside e page-break-after.

O page-break-before e page-break-after denem se a quebra de página deve ocorrer antes ou depois do elemento. A propriedade aceita os valores always, utili zado para forçar a quebra, ou avoid, indicando para o navegador que a quebra deve ser evitada. Com eles é possível, por exemplo, ao imprimir uma relação de posts de um blog, deve existir uma quebra de página entre um post e outro.

article {

page-break-after: always;

}

Ou caso você precise de um controle mais renado das quebras, você pode de- nir um elemento vazio para posicionar as quebras em pontos especícos do seu conteúdo.

.page-break {

display: none;

}

@media print {

.page-break {

55

4.8. Gere relatórios inteligentes e simples com os estilos de impressão Casa do Código

display: block;

page-break-before: always;

}

}

E então adicionar uma tag div, por exemplo, com a classe .page-break onde for necessário forçar a quebra de página.

Já com o page-break-inside, que aceita apenas as opções auto e avoid, po demos denir que o conteúdo de um parágrafo **não** deve ser quebrado entre duas páginas.

p {

page-break-inside: avoid;

}

Este tipo de controle é interessante para blocos de conteúdo extensos, como ma nuais, documentações ou mesmo e-books criados em HTML. Além destas 3 propri edades existem outras duas, utilizadas para denir os limites de linhas que deve car em uma página ou em outra: orphans e widows. Mas infelizmente estas propriedades não possuem um suporte extenso pelos navegadores.

**Impressão de tabelas**

Vamos praticar um pouco e melhorar uma tabela de um relatório nanceiro com débitos e lucros obtidos. Além dos valores, a tabela possui alguns links para navegar entre os dados e tomar outras ações no sistema. O HTML é bem simples, tendo apenas uma tabela e algumas linhas.

<h1>Transações de Maio, 2011</h1>

<table>

<tr>

<td>

<a href='http://exemplo/transacao/1'>Transação 1</a>

</td>

<td class='expense'>R$ 100,00</td>

<td class='actions'>

<a href="http://exemplo/transacao/1/editar">Editar</a> </td>

</tr>

<tr>

56

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

<td>

<a href='http://exemplo/transacao/2'>Transação 2</a>

</td>

<td class='profit'>R$ 200,00</td>

<td class='actions'>

<a href="http://exemplo/transacao/2/editar">Editar</a> </td>

</tr>

</table>

E um pouco de CSS para melhorar o estilo da tabela, denindo sua largura, cores para as bordas das células, alinhamentos e também a denição de links:

table {

width: 500px;

}

td {

border: 1px dotted #666;

padding: 5px;

}

td a {

color: #3B5998;

}

.expense, .profit, .actions {

text-align: right;

}

.expense, .profit {

font-weight: bold;

}

.expense { color: red; }

.profit { color: green; }

.actions a {

font-size: 0.9em;

color: white;

padding: 2px 5px;

background-color: #3B5998;

57

4.8. Gere relatórios inteligentes e simples com os estilos de impressão Casa do Código

text-decoration: none;

}



Figura 4.19: A tabela do relatório nanceiro, em sua versão padrão.

Além de denir uma largura para a tabela e denir algumas regras de alinha mento, registros de lucro carão com a cor verde e gastos serão exibidos em verme lho. Ao imprimir esta página em preto e branco não possuímos nenhuma distinção visual disso e os links não estarão disponíveis no papel. Vamos escrever algumas regras de CSS para impressão e ajustar isto.

Primeiro, vamos garantir que todo o conteúdo esteja na cor preta, e que a tabela ocupe o máximo de espaço possível.

@media print {

\* {

color: #000 *!important* ;

}

table {

width: 100%;

}

td {

border-color: #000;

}

}

Também podemos trabalhar um pouco nos links existentes na página. Os da primeira coluna podem ser expandidos utilizando a técnica do pseudo elemento. Já a coluna com os links de edição é dispensável para a nossa visão de impressão, por ser um elemento de navegação sem nenhum conteúdo relevante para este cenário, podemos então esconder esta coluna inteira.

58

Casa do Código Capítulo 4. O que todo desenvolvedor precisa saber sobre CSS

a::after {

content: "(" attr(href) ")";

margin-left: 2px;

}

.actions {

display: none;

}

Por m, temos a coluna de receitas e débitos a tratar. Indo na onda de pseudo elementos, podemos adicionar duas características a casa célula desta coluna. Um sinal de **-** ou **+**, dependendo do caso, antes do valor, e os termos **Receita** e **Despesa**.

Com o uso do vermelho e do verde removido nos estilos de impressão, estas indicações serão úteis para identicar o tipo de transação de cada linha. O CSS para adicionar este conteúdo é o seguinte:

.expense::before {

content: "- ";

}

.expense::after {

content: " - Despesa";

}

.profit::before {

content: "+ ";

}

.profit::after {

content: " - Receita";

}



Figura 4.20: A versão de impressão nalizada.

59

4.8. Gere relatórios inteligentes e simples com os estilos de impressão Casa do Código

Assim é possível adequar diversos aspectos do visual das nossas páginas para o modo de impressão, de acordo com as necessidades dos seus usuários e os objetivos do tipo de projeto que você estiver fazendo. Claro que este tipo de tratamento não é necessário em todos os projetos, mas é uma carta valiosa na sua manga.

60

Capítulo 5

O que você consegue fazer com CSS 3

Diversas adições do CSS 3 são extremamente úteis para substituir vários tipos de imagens que utilizávamos para adicionar cores e formas aos elementos HTML, o que não conseguíamos atingir com apenas CSS. Além da redução de arquivos para se trabalhar e a ausência de dependências externas que impactam na performance dos sites, a exibilidade dessas propriedades permite diversas combinações diferentes, que geram muitos estilos que você não imaginaria ser tão simples de se criar com apenas CSS.

O poder de propriedades como box-shadow e funções como o linear-gradient é tão grande que é comum encontrar designers e desenvolvedores - eu faço parte deste grupo - que aposentaram os editores de imagens e criam as formas, cores e efeitos das páginas direto no código, conferindo os resultados ao vivo em seus nave gadores.

5.1. A regra @font-face Casa do Código

**5.1 A regra @font-face**

Uma das funcionalidades que adiciona muita exibilidade ao dar vida aos designs das nossas páginas é a regra @font-face. Ela permite denir novas famílias de fontes que as nossas páginas utilizam, e informa ao navegador e ao sistema operacional do usuário onde o arquivo necessário para desenhar os traços da sua fonte estão localizados. A denição de uma nova fonte é a seguinte:

@font-face {

font-family: "Lobster";

font-style: normal;

font-weight: 400;

src: local('Lobster'), url(/fonts/lobster.woff) format('woff'); }

Utilizando as mesmas propriedades usadas para alterar os estilos de fonte de um elemento, podemos denir os detalhes da nova fonte. No caso de denir fontes com itálico ou negrito, precisamos informar os valores corretos para as propriedades font-style e font-weight de acordo com o uso. Já a propriedade src é responsável por denir a localização da fonte: a função local informa um possível nome para a fonte que será pesquisado na máquina do usuário, ou uma URL externa para que a fonte seja baixada como um recurso adicional, igual a uma imagem de fundo, por exemplo.

Com a fonte denida, podemos utilizá-la normalmente:

h1 {

font-family: "Lobster", cursive;

}

Caso a nossa fonte Lobster não esteja disponível, por um problema de rede ou a falta de um arquivo compatível, uma fonte alternativa como cursive deve ser de- nida para ser aplicada ao elemento. A compatibilidade de formatos diferentes é um ponto irritante: existem diversos formatos de fontes diferentes que você pode pre cisar dependendo dos navegadores que você pretende atender: .eot para algumas versões do Internet Explorer e .svg, .ttf ou novamente o .eot para o navegador nativo do Android, ou versões desatualizadas do Safari, tanto em desktops como no iOS. O formato tomado como padrão, suportado por todos os navegadores moder nos, é o woff.

62

Casa do Código Capítulo 5. O que você consegue fazer com CSS 3 

Figura 5.1: As diferenças da ’Lobster’ para o ’cursive’, uma alternativa aceitável caso a fonte não esteja disponível.

**5.2 Como utilizar serviços de distribuição de fon tes**

Existem alguns serviços disponíveis na Internet que facilitam o uso de fontes adi cionais, disponibilizando o código necessário, servindo os arquivos de fonte e tra tando de assuntos relacionados a licenças de uso, por exemplo. Os mais famo sos neste campo são o **Typekit** (https://typekit.com) e o **Google Web Fonts** (http: //www.google.com/webfonts).

Alguns sites alternativos disponibilizam o download dos arquivos necessários e um exemplo de uso das fontes em CSS, mas sem servir os arquivos direta mente dos seus servidores, deixando esta responsabilidade a nós desenvolvedo res, como o **FontSpring** (http://www.fontspring.com/) e **FontSquirrel** (http://www. fontsquirrel.com/).

Podemos utilizar o serviço de fontes do Google, o Google Web Fonts, para pra ticar um pouco. Comece com o seguinte HTML.

<h1>HTML & CSS</h1>

Agora, vamos carregar a fonte Press Start 2P (http://www.google.com/ webfonts/specimen/Press+Start+2P), para adicionar um visual nostálgico de 8 bits ao título. Adicione a seguinte linha de HTML ao head da sua página.

<link href='http://fonts.googleapis.com/css?family=Press+Start+2P' rel='stylesheet'>

Se você acessar o endereço desta folha de estilo, http://fonts.googleapis.com/css? family=Press+Start+2P, verá que ela não faz nada além de denir a fonte Press Start 2P, referenciando um arquivo .woff presente nos servidores do Google. Podemos então utilizar esta fonte no h1 que criamos.

63

5.3. Substituição de ícones por fontes Casa do Código

h1 {

font-family: 'Press Start 2P', cursive;

}



Figura 5.2: A fonte ’Press Start 2P’ em uso.

**5.3 Substituição de ícones por fontes**

Outro uso bastante interessante para fontes é substituição de pacotes de ícones. No lugar de uma pilha de imagens, você pode utilizar uma fonte composta de ícones, e não caracteres normais, e ganhar o poder de controlar o tamanho e cor do seus ícones sem precisar editar imagens e exportar novos arquivos para dentro de seus projetos.

Existem diversas fontes e ferramentas existentes para ajudar a compor gru pos de ícones e adicionar eles as suas páginas sem muito trabalho. Algu mas das coleções de ícones mais interessantes por ai: Pictos (http://pictos.cc/), Font Awesome (http://fortawesome.github.com/Font-Awesome/) e Iconic (http:// somerandomdude.com/work/iconic/). E para compor ou pesquisar outras opções, existem opções como Fontello (http://fontello.com/), IcoMoon (http://keyamoon. com/icomoon/#toHome) e o Shiicons (https://www.shiicons.com/).

Pegue a fonte Iconic, por exemplo, para testar um pouco. Você pode baixar o pacote direto do site do seu criador, P.J. Onori, em http://somerandomdude.com/ work/iconic/, ou pegar a última versão da fonte no seu repositório no GitHub - https: //github.com/somerandomdude/Iconic.

Em vez de utilizar o CSS que o projeto já disponibiliza para você, vamos escrever um exemplo do zero para ver o uso da fonte. Começando com a denição da fonte no CSS.

@font-face {

font-family: 'IconicFill';

src: url('iconic\_fill.woff') format('woff');

font-weight: normal;

font-style: normal;

}

64

Casa do Código Capítulo 5. O que você consegue fazer com CSS 3

Então, adicionamos um elemento e uma classe para denir qual ícone será exi bido junto do nosso título. Uma ótima forma de exibir tais ícones de fontes é utili zando a propriedade content dos pseudo elementos:

<h1 class='icon-next'>Avançar</h1>

.icon-next::after {

content: '\2192';

font-family: 'IconicFill';

margin-left: 10px;

}



Figura 5.3: O ícone criado utilizando a fonte ’Iconic’.

Utilizando um pseudo elemento e um símbolo Unicode, o ícone de uma echa para a direita é exibido ao lado do título. Qualquer mudança dos estilos do texto do elemento ou do seu pseudo elemento irá afetar o ícone, adequando cores e tamanhos. Podemos trocar a cor de todos os ícones associados a um h1 e até utilizar o ícone em outro elemento, como um p, adicionando o seguinte código:

<p class='icon-next'>Ir para a próxima página</p>

h1.icon-next {

color: blue;

}



Figura 5.4: Os ícones reetem o estilo do elemento que os contém. 65

5.3. Substituição de ícones por fontes Casa do Código

Desta forma conseguimos posicionar e estilizar todos os ícones disponíveis no Iconic facilmente com CSS. Outro exemplo são os clássicos ícones para ações feitas com sucesso ou erros de validações de formulários.

.icon-ok {

color: #489D00;

}

.icon-ok::before {

content: "\2714";

font-family: 'IconicFill';

margin-right: 10px;

}

.icon-invalid {

color: #990000;

}

.icon-invalid::before {

content: "\2718";

font-family: 'IconicFill';

margin-right: 10px;

}

<p class='icon-ok'>E-mail enviado com sucesso!</p>

<p class='icon-invalid'>Atenção, preencha todos os campos do formulário

</p>



Figura 5.5: Combinando ícones e cores para criar mensagens.

Enquanto o Iconic utiliza símbolos Unicode como caracteres para os seus íco nes, outras fontes podem utilizar outros caracteres para isto. É necessário conferir a documentação - caso exista uma - a respeito de cada fonte antes de utilizá-la.

66

Casa do Código Capítulo 5. O que você consegue fazer com CSS 3

**5.4 Explore novas possibilidades com bordas**

Todo elemento HTML é comparado a uma caixa, mas nem todos os elementos das nossas interfaces precisam ser quadrados. Em vez de se utilizar imagens a torto e di reito, podemos utilizar o border-radius para arredondar os cantos dos seus elemen tos. A propriedade aceita até 4 valores, seguindo o modelo de outros como margin e padding, sendo a única diferença que o primeiro valor é referente ao canto su perior esquerdo do elemento, seguindo em diante no sentido horário. Enquanto alguns cantos arredondados podem parecer apenas um pequeno detalhe estético, o border-radius consegue criar mais do que isso. Veja a seguir.

<h3 class='tnt'>TNT</h3>

.tnt {

border-radius: 50%;

border: 5px solid #000;

height: 50px;

line-height: 50px;

text-align: center;

width: 50px;

}



Figura 5.6: Utilizando border-radius para criar círculos

Podemos utilizar porcentagens e criar círculos ou elipses, independente do ta manho real do elemento. Fora isto, cada canto do elemento pode receber dois valores para denir exatamente qual será o raio utilizado para criar o efeito.

<span class='counter'>37</span>

.counter {

background-color: #000;

border-top-left-radius: 25px 10px;

border-top-right-radius: 25px 10px;

color: white;

display: block;

67

5.4. Explore novas possibilidades com bordas Casa do Código

font-size: 1.7em;

height: 50px;

line-height: 50px;

text-align: center;

width: 50px;

}



Figura 5.7: Um exemplo de uso mais renado do border-radius.

Este é um caso de uso mais raro de se encontrar, visto que a maioria das aplica ções de border-radius são perfeitamente circulares, e geralmente feitos para realçar detalhes de certos elementos, como botões, caixas de texto ou caixas de mensagens.

**Criando nuvens**

Não, não vamos falar sobre *Cloud Computing* neste livro. Mas vamos desenhar um ícone de uma nuvem utilizando pseudo elementos e o border-radius, para de monstrar como é possível criar ícones e outras formas que geralmente se utiliza como imagens no seu código. No nosso HTML será bastante simples, sem nenhum con teúdo.

<div>

<span class='cloud'></span>

</div>

Primeiro, com o span iremos desenhar a base da nuvem, e a div servirá como um fundo do nosso ícone.

div {

background-color: #000;

width: 50px;

height: 10px;

padding: 20px 50px;

}

68

Casa do Código Capítulo 5. O que você consegue fazer com CSS 3

.cloud {

background-color: #FFF;

border-radius: 48px;

display: inline-block;

height: 16px;

position: relative;

width: 48px;

}



Figura 5.8: A base da nuvem, criada com border-radius.

O ponto chave destes valores é o border-radius de 48px, que é 3 vezes o valor da altura dele - 16px. Caso fosse necessário uma nuvem maior, só iríamos manter esta proporção de valores ao denir novos tamanhos. Vamos ao próximo pedaço:

.cloud::before {

background-color: red;

border-radius: 50%;

content: '';

height: 14px;

position: absolute;

right: 7px;

top: -6px;

width: 14px;

}



Figura 5.9: O pseudo elemento posicionado acima da base.

Utilizando o ::before, desenhamos uma esfera de 14px posicionada de certa forma a deixar metade do elemento para fora da base. O fundo vermelho servirá apenas para ajudar a distinguir cada um dos 3 elementos entre si. Precisamos agora

69

5.4. Explore novas possibilidades com bordas Casa do Código

de outro círculo, um pouco maior, posicionado ao lado direito do span. Podemos reaproveitar as denições de posição e tamanho do ::before e sobrescrever apenas alguns valores para criar outro elemento maior.

.cloud::before, .cloud::after {

background-color: red;

border-radius: 50%;

content: '';

height: 14px;

position: absolute;

right: 7px;

top: -6px;

width: 14px;

}

.cloud::after {

background-color: blue;

height: 23px;

left: 7px;

right: auto;

top: -10px;

width: 23px;

}



Figura 5.10: Mais uma parte da nuvem, posicionada no outro lado da base.

Agora o ::after, utilizando da mesma técnica para se desenhar círculos, ocupa o lado direito da nuvem. Agora basta alterar os pseudo elementos para que ambos quem com a mesma cor de fundo do span - #FFF% - e ver o resultado nal do experimento.

70

Casa do Código Capítulo 5. O que você consegue fazer com CSS 3 

Figura 5.11: Trocando os fundos para branco conseguimos ver a forma completa da nuvem.

Um caso clássico de uso para border-radius é na criação de estilos para com ponentes de formulários: de caixas de texto a botões. Esta aplicação merece uma aprofundamento maior, que você encontra na seção 7.9 deste livro.

**5.5 Manipulação de cores com rgba e gradientes**

Além de fontes e bordas, outro artifício visual ganhou muito com a chegada do CSS 3 - as cores. Antigamente limitadas a apenas textos e fundos, o que sempre resultava na mesma situação das bordas arredondadas - uso de imagens para criar gradientes e opacidades, o que impacta na performance e na exibilidade de manutenção do seu código. Funções muito úteis sugiram para suprir diversos casos de uso de imagens, como o rgba e o linear-gradient.

Primeiro, ao rgba. A função permite denir a opacidade de uma cor, permitindo criar efeitos de transparências utilizando apenas a cor de fundo ou as bordas de um elemento. Um ponto crucial é que a função não recebe um valor hexadecimal, como #000000, e sim utilizando os 3 decimais que representam a cor, no caso do preto são 0,0,0. O quarto argumento da função é a opacidade deseja, de 0.0 a 1.0.

Vamos a ação. Com o HTML a seguir, nós temos uma foto e uma legenda, uti lizando novas tags como o figure e o figcaption para agrupar o img com a sua legenda.

<figure>

<img src="/images/sf.jpg" alt="A Golden Gate.">

<figcaption>

San Francisco, Califórnia

<small>

Por Salim Virji (http://www.flickr.com/photos/salim/402618628/) </small>

</figcaption>

</figure>

71

5.5. Manipulação de cores com rgba e gradientes Casa do Código

Nosso primeiro passo com o CSS será posicionar a mensagem por cima da foto, alguns pixels de distância do nal dela.

figure {

position: relative;

}

img {

display: block;

}

figcaption {

bottom: 5px;

margin: 0 5px;

padding: 5px;

position: absolute;

width: 300px;

}



Figura 5.12: Posicionamos a legenda sobre a foto, mas não é fácil ler o que está escrito.

Desta forma não temos contraste nenhum para ler o texto da legenda. Hora de aplicar um fundo preto e trocar a fonte da legenda para branco, melhorando sua leitura.

figcaption {

background-color: #000;

72