CSS Avançado

Joseph R. Lewis e Meitar Moscovitz

Novatec

CAPÍTULO 1

Marcação dá suporte ao CSS

Dizem que toda jornada começa com um único passo. A jornada em que estamos começou em 1965 na Brown University, onde a ideia do *hipertexto*[1] surgiu. Hipertexto, ou texto em documentos que permite navegação não-linear (chamada *navegação hipertextual*), foi eventualmente codificada como uma linguagem de marcação denominada Linguagem de Marcação de Hipertexto (*Hypertext Markup Language*, ou HTML).

Conforme a Web se desenvolveu, o HTML foi esticado e deformado enquanto as pessoas tentavam torná-lo tudo para todos. Dentro dos documentos HTML existia uma mistura de informação, por vezes chamada de "sopa de tags" pelos desenvolvedores, que incluía lógica de visualização e código para interação de usuários. Em algum lugar no meio de todos esses metadados você também poderia acabar encontrando algum conteúdo.

Não demorou muito para as pessoas perceberem que essa abordagem não funcionaria no futuro porque ela era fundamentalmente limitante. Em vez de tentar disponibilizar um documento monolítico aos navegadores web, fazia muito mais sentido dar aos navegadores os blocos de construção do conteúdo em si e, então, deixar que o navegador cuidasse de juntar tudo. Esse princípio é conhecido como separação de interesses (*separation of concerns*).

Essa simples ideia é tão fundamental para compreender como a Web funciona que nos obriga a olhar mais de perto a marcação dos documentos sobre a qual as Folhas de Estilo em Cascata (*Cascading Style Sheet*, ou CSS) atuam. Vamos, então, examinar rapidamente a sintaxe e a semântica das linguagens de marcação.

A sintaxe das linguagens de marcação

Ao ler o código de uma linguagem de marcação, você está lendo uma linguagem real de forma muito semelhante a quando lê as palavras em português deste livro.

1 Mais precisamente, a ideia de hipertexto é geralmente creditada a Vannevar Bush em 1945, quando este escreveu sobre uma escrivaninha eletrônica que ele chamou de "Memex". Na sua maior parte, nossos computadores pessoais são manifestações modernas desse dispositivo. Foi em 1965 que Ted Nelson cunhou o termo hipertexto para descrever tal ideia.

Como você sabe, ao visualizar uma página web, você está requisitando ao seu navegador que execute um documento. Este documento é apenas um arquivo de texto construído de forma especial. Ele é construído de acordo com a gramática de uma linguagem de marcação em particular para que todo o conteúdo inserido nele seja representado em um formato estruturado.

Acontece que essa estrutura tem algumas semelhanças incríveis com as linguagens humanas. Colocado de maneira simples, quando construímos sentenças em linguagens humanas, costumamos usar uma estrutura gramatical que começa com um sujeito, seguido por um verbo e que termina com um objeto. Por exemplo, na sentença "A vaca pulou sobre a lua", a *vaca* é o sujeito, *pulou* é o verbo e a *lua* é o objeto.

Linguagens de marcação também têm regras de gramática. Existem *elementos*, cada um sendo um bloco de construção ou componente de um todo maior. Um elemento pode ter certas propriedades que especificam mais profundamente seus detalhes. Essas propriedades normalmente são *atributos*, mas também podem ser o próprio conteúdo[2] do elemento. Por fim, um elemento tem certa posição dentro do documento relativa a todos os outros elementos, dando-lhe um *contexto hierárquico*.

No código HTML a seguir, podemos ver um exemplo desses três conceitos:

```
<blockquote cite="http://example.com/">
  <p>Welcome to example.com!</p>
</blockquote>
```

Você pode pensar a respeito do elemento `<blockquote>` como sendo nosso assunto. Ele tem um atributo, `cite`, que descreve de onde o parágrafo citado veio, e você pode imaginá-lo como sendo nosso verbo. O elemento `<p>`, que é a citação em si, pode ser considerada nosso objeto. Em português, esse fragmento de código poderia ser traduzido para algo como, "Esta citação (*quote*) cita a página *http://example.com/*".

Já temos algumas observações interessantes a fazer sobre esse exemplo simplista. Em primeiro lugar, o documento é totalmente relacionado à *semântica*, ou seja, ao *significado* do código. (O papel importante que a marcação semântica tem na Web é discutido com mais detalhes no capítulo 7.) Em segundo lugar, *apenas* informação semântica e conteúdo em si (texto, nesse caso) estão presentes no código de exemplo. Em nenhum lugar você vê qualquer informação a respeito da forma em que esse conteúdo será exibido ao usuário ou quais opções o usuário possui para interagir com ele.

2 Entender quando usar determinadas estruturas sintáticas é uma questão comum que os desenvolvedores em linguagem XML levantam. Em seu artigo sobre os Princípios de Projeto XML (Principles of XML Design), Uche Ogbuji articula claramente quando e por que usar uma determinada estrutura, como os elementos, em vez de outra – atributos, por exemplo. O artigo pode ser encontrado em *http://www.ibm.com/developerworks/xml/library/x-eleatt.html*.

Essa falta de informação de apresentação e de comportamento é um exemplo da separação de interesses em funcionamento. Por ter entregado apenas o conteúdo semântico em si ao navegador, agora você tem a flexibilidade de misturar esse mesmo conteúdo com qualquer outra apresentação ou comportamento que desejar dar a ele. Por fim, é claro, esse exemplo foi escrito em XHTML, a linguagem de marcação usada para criar páginas web tradicionais.

A semântica do HTML remonta a uma linguagem de marcação que originalmente tinha a intenção de descrever artigos acadêmicos. Ela foi derivada de um subconjunto da Linguagem de Marcação Padrão Generalizada (*Standard Generalized Markup Language*, ou SGML) da IBM, que era usada por grandes empresas e organizações governamentais para codificar documentos industriais complexos.[3] É por isso que o HTML tem um vocabulário relativamente rico para descrever diferentes tipos de estruturas textuais normalmente encontradas em recursos escritos, como listas (`<ul>`, `<ol>` e `<dl>`) e saídas do computador (`<code>`, `<samp>`, `<kbd>`), mas tem uma escassez de outros tipos de vocabulários.

Por sorte, atualmente existe uma variação moderna do SGML genérico que permite a qualquer um, incluindo desenvolvedores individuais como nós, criar sua própria linguagem de marcação com seu próprio vocabulário. Essa tecnologia é conhecida como XML, que é indiscutivelmente a tecnologia sobre a qual a Web do futuro será construída.

Dialetos XML: os diversos sabores de conteúdo

Um dos desafios da Web é descrever os diferentes tipos de conteúdo. Com a semântica limitada do HTML, descrever todos os diferentes tipos de conteúdo disponíveis é desajeitado, na melhor das hipóteses, e impossível, na pior das hipóteses. Então, como isso é feito?

Desde o princípio, a World Wide Web Consortium (W3C), uma organização que governa a publicação de padrões de tecnologia web de amplitude industrial, percebeu que, para ter sucesso, a Web necessitava da habilidade de descrever todos os tipos de conteúdos, tanto os existentes quanto os que ainda estavam para ser criados, de uma forma que pudesse ser facilmente interoperável. Em outras palavras, se Joe inventou algo novo e o tornou disponível on-line, o ideal seria que ele o fizesse de maneira acessível a Jane.

A solução desenvolvida pela W3C foi a Linguagem de Marcação Extensível (*Extensible Markup Language*, ou XML). O XML tem duas características primárias que o torna excepcionalmente apropriado para a Web. Primeiro, é um formato genérico de documentos projetado para ser legível tanto por humanos quando por máquinas. Isso

3 Talvez o mais famoso dos documentos marcados em SGML seja o *Oxford English Dictionary*, que até hoje permanece codificado em SGML.

é feito usando-se um subconjunto estrito da sintaxe e gramática familiares que o HTML usa, então é muito fácil de ser lido pelos humanos e de ser analisado pelas máquinas. Segundo, os documentos marcados funcionam como um mecanismo de serialização de dados, permitindo aos sistemas autônomos trocar facilmente entre si os dados contidos neles.

O que separa o XML dos outros formatos-padrão de documentos é que, da mesma forma que o SGML, ele não é uma linguagem específica, e sim uma metalinguagem. Se alguém ou algum sistema nos disser que "consegue falar XML", nossa primeira questão seria: "Qual dialeto XML você fala?". Isso acontece porque todo documento XML é um tipo distinto de XML, que pode ou não ser um padrão mais amplamente conhecido, com seu próprio vocabulário (conjunto de elementos válidos) e semântica (coisa que esses elementos significam).

Conforme mais conteúdos díspares surgiam na Web, várias linguagens de marcação baseadas em XML foram desenvolvidas para descrever tais conteúdos. A mais famosa dessas linguagens é, sem dúvida, a Linguagem de Marcação de Hipertexto Extensível (*Extensible Hypertext Markup Language*, ou XHTML), que é simplesmente uma *aplicação* (jargão do XML para "instância específica") XML que copia todas as semânticas da versão derivada do SGML, do bom e velho HTML, e que as define usando a sintaxe do XML. Não existe qualquer distinção significativa entre HTML e XHMTL além das restrições impostas pela sintaxe mais restritiva do XML.

Uma das outras vantagens fundamentais do XML é o seu suporte a *namespaces* (espaços de nomes). Um *namespace XML* é simplesmente uma formalização dessa ideia de "múltiplos dialetos". Especificamente, um namespace definido usando o atributo `xmlns` no elemento-raiz de qualquer linguagem de marcação baseada em XML identifica cada dialeto XML com uma string exclusiva, permitindo, assim, que qualquer linguagem de marcação baseada em XML se misture e corresponda à semântica de múltiplas aplicações XML dentro de um único documento. Isso é importante porque fornece uma maneira aos desenvolvedores de aplicações (incluindo, teoricamente, fabricantes de navegadores) de projetar linguagens de marcação de maneira modular, de acordo com os princípios e ideais da antiga abordagem "ferramentas de software" que os sistemas Unix iniciais adotavam. Conceitualmente, isso não é diferente de uma versão em computador da noção de humanos poliglotas.

Cada aplicação XML pode, por si só, ser descrita por uma outra forma de documento XML denominado Definição de Tipo de Documento (*Document Type Definition*, ou DTD). Arquivos DTD são como dicionários das linguagens de marcação; eles definem quais elementos são válidos, quais propriedades esses elementos têm e as regras de onde esses elementos podem aparecer dentro do documento. Esses documentos são escritos por desenvolvedores de aplicações XML, e a maior parte das utilidades desses

arquivos para os desenvolvedores web típicos é simplesmente servir como uma referência autorizada (por vezes críptica) da sintaxe particular de uma aplicação XML. Todo documento XML começa com duas linhas de código que definem a versão de XML que ele usa, denominada *prólogo do XML*, e qual arquivo DTD ele usa, denominada *declaração do tipo de documento* (ou DOCTYPE).

Apesar de o CSS poder ser usado em conjunto com algumas dessas tecnologias voltadas ao usuário, o restante deste livro focaliza em algumas aplicações XML específicas que são amplamente usadas atualmente. Vamos, então, fazer um *tour-relâmpago* de algumas dessas aplicações que vamos estilizar com o CSS no restante do livro.

RSS e Atom: formatos de syndication de conteúdo

Possivelmente, o formato mais comum de documento derivado de XML na Web atual, e que não seja uma página web tradicional, são aqueles formatos usados para sindicar conteúdos como "web feeds" ou "news feeds". Os dois padrões (razoavelmente equivalentes) para essa tecnologia são o RSS e o Atom. Tais padrões especificam um formato baseado em XML para documentos que resumem o conteúdo e o grau de novidade de *outros* recursos, como artigos de notícias, podcasts ou postagens em blogs na Web.

Apesar de existirem inúmeras versões diferentes de RSS, como todas elas têm o mesmo objetivo, usaremos o RSS 2.0 ao longo de todo o livro. Atom, na verdade, é um termo que se refere tanto a um formato de documento XML, chamado Atom Syndication Format, quanto a um protocolo HTTP peso-leve. Quando usarmos o termo *Atom* neste livro, estaremos nos referindo ao Atom Syndication Format, a menos que digamos o contrário.

Como o RSS e o Atom foram projetados para descrever outros conteúdos, seus vocabulários consistem quase que completamente de conjuntos de elementos que descrevem metadados de documentos. Por exemplo, cada formato tem um elemento de codificação do URI (endereço de rede) de um determinado recurso. No RSS, estamos nos referindo ao elemento `<link>`, e no Atom ao elemento `<guid>`. De maneira semelhante, ambos os formatos têm um elemento para codificar o título, a descrição e outras informações sobre recursos.

Tanto os feeds RSS quanto os feeds Atom podem aparecer como código-fonte sem estilos no navegador, apesar de alguns navegadores oferecerem funcionalidades nativas de leitura de feeds. Quando essas funcionalidades são ativadas, tais navegadores aplicam alguns estilos predefinidos ao feed carregado. Por exemplo, a figura 1.1 mostra o que o Firefox faz com um feed RSS visualizado via HTTP. Em contraste, a figura 1.2 exibe o mesmo feed no mesmo navegador, quando acessado por meio do sistema de arquivos local.

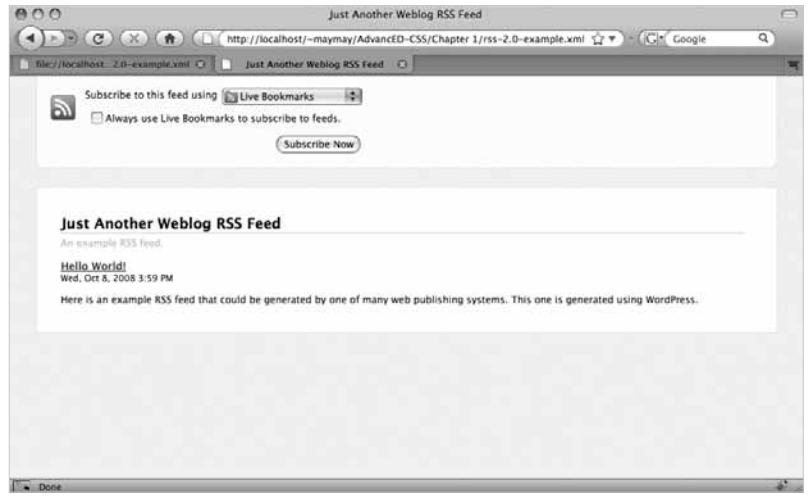

Figura 1.1 – O Firefox 3 usa uma combinação de XSLT e CSS para dar algum estilo aos web feeds.

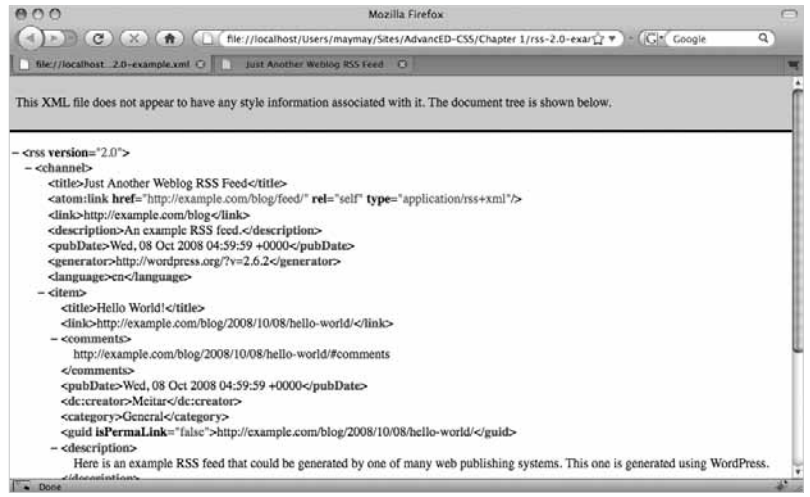

Figura 1.2 – O Firefox não aplica seus estilos predefinidos aos feeds RSS acessados por meio do esquema de arquivo.

Como você pode ver, as duas telas quase não têm relação uma com a outra. Na figura 1.1, o Firefox exibe os dados do feed usando uma interface de usuário amigável, ao passo que, na figura 1.2, é exibido o código-fonte (praticamente) bruto. O que é mais interessante para os nossos propósitos na figura 1.2 é a confissão do Firefox de que "Este arquivo XML aparenta não ter informações de estilo associadas a ele".

A informação de estilo à qual o Firefox se refere é qualquer prólogo XML `<? xml-stylesheet ... ?>`. Semelhante ao elemento `<link>` do XHTML, um `xml-stylesheet` pode receber um atributo `type` cujo valor pode ser `text/css`. Quando isso acontece, o Firefox usa a informação de estilo fornecida no arquivo CSS para alterar a apresentação do

feed. Apesar do mecanismo específico de vínculo variar, adicionar uma folha de estilos a um documento XML que não seja XHTML não é diferente de fazê-lo em uma página web comum. O uso do prólogo `xml-stylesheet` para adicionar folhas de estilos a documentos XML como esses será abordado com mais detalhes no capítulo 9.

Exploradores intrépidos devem ter percebido que a folha de estilos usada pelo Firefox na figura 1.1 tem seletores que apontam para elementos HTML, e não RSS. De fato, se utilizar o Firebug(*) para inspecionar o código-fonte da maioria dos feeds RSS da Internet no Firefox 3, você verá HTML e nenhum RSS. Como assim? De onde veio o HTML?

Acontece que, no jargão XML, uma "folha de estilos" pode, na verdade, significar duas coisas. Ela pode remeter a um arquivo CSS na forma que já conhecemos e amamos, mas também pode significar um arquivo de Transformações de Linguagem de Folhas de Estilo Extensível (*Extensible Stylesheet Language Transformations*, ou XSLT). Esses arquivos constituem uma outra forma de documento XML que, dado um arquivo XML original formatado, são usados para converter a marcação de um tipo de XML para outro. O Firefox usa um arquivo XSLT para transformar na hora os feeds RSS e Atom em um arquivo XHTML, de forma que o arquivo CSS usado para aplicar estilos ao feed é vinculado ao resultado da transformação XSLT (XHTML, no caso), em vez de ser vinculado ao feed original.

O XSLT está além do escopo deste livro, mas realmente recomendamos que você observe-o mais de perto, caso suspeite que terá de fazer uma quantidade significativa de processamento de documentos XML.

(*) O Firebug é um excelente add-on para o navegador web Firefox que dá aos usuários a capacidade de inspecionar e alterar o documento carregado pelo navegador. Acreditamos que esta é uma ferramenta fundamental para qualquer desenvolvedor web sério.

SVG: gráficos em XML baseados em vetores

Gráficos vetoriais e animação 2D na Web foram popularizados pelo Flash, um formato binário adquirido pela Macromedia (que, desde então, fundiu-se com a Adobe) e que é comercializado sob o nome ShockWave Flash. No entanto, desde 2001 (bem antes de o Flash ter popularizado o formato), já existe um padrão aberto baseado em XML para definição desses tipos de gráficos. Esse padrão é conhecido como Gráficos Vetoriais Escaláveis (*Scalable Vector Graphics*, ou SVG).

A história do SVG é bem interessante. Criado pela Sun Microsystems e Adobe (que, ironicamente, agora é a proprietária da Macromedia), o SVG foi (e ainda é) amplamente usado na aplicação de edição gráfica Adobe Illustrator como um formato de gráficos vetoriais facilmente exportável e importável. Na verdade, a maneira mais simples de se criar um documento SVG atualmente é simplesmente desenhar algo no Adobe Illustrator e escolher salvá-lo no formato SVG a partir da opção de menu **Save As**.

Infelizmente para a Adobe, a Microsoft criou uma linguagem competidora de marcação de gráficos vetoriais chamada Linguagem de Marcação Vetorial (*Vector Markup Language*, ou VML) e a implementou no seu navegador web, o Internet Explorer. Durante os anos da guerra dos navegadores devido à ascensão do Flash e a falta de uma implementação de uma linguagem de marcação que funcionasse em todos os navegadores, o SVG caiu no esquecimento. O SVG permaneceu como um padrão desconhecido e intocado por um longo tempo.

No entanto, recentemente, graças aos interesses renovados sobre os padrões web, vem acontecendo uma ressurreição das implementações SVG nativas entre os navegadores. Todos os principais navegadores, incluindo Firefox, Safari e Opera, atualmente suportam um subconjunto da especificação SVG em graus variados. Em alguns casos, o próprio plug-in SVG da Adobe adiciona ainda mais suporte. Mais ainda, com a ajuda de scripts e o trabalho duro dos desenvolvedores da biblioteca JavaScript[4], agora é possível usar gráficos vetoriais de maneira confiável em vários navegadores.

Em virtude de sua natureza extremamente visual, o SVG fornece um ambiente ideal para examinar o CSS mais de perto. Como todas as formas de XML, os documentos SVG começam com um prólogo XML, seguido por um DTD, seguido por um elemento-raiz que engloba todo o conteúdo do documento. Esse é o mesmo padrão usado nos documentos XHTML[5], então ele deve soar familiar. A única distinção é que, no caso do SVG, o elemento-raiz é `<svg>`, em vez de `<html>`.

Portanto, o esqueleto de um documento SVG poderia se parecer com o seguinte fragmento:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <!-- É aqui que os elementos SVG serão colocados. -->
</svg>
```

Como o SVG foi projetado para descrever objetos visuais, em vez de objetos textuais, o vocabulário disponível para desenvolvedores SVG difere radicalmente daquele disponível no XHTML. Por exemplo, no SVG não existem elementos `<p>` ou `<div>`. Em

4 Um desenvolvedor digno de nota é Dmitry Baranovskiy, cujo trabalho na biblioteca Raphaël Javascript permite que um arquivo SVG seja traduzido na hora para o formato VML da Microsoft, de forma que as imagens baseadas em SVG possam funcionar tanto nos navegadores compatíveis com os padrões quanto no Internet Explorer.

5 Na prática, a maioria dos documentos que usam um DTD XHTML não inclui um prólogo XML. Isso acontece porque a inclusão do prólogo de forma errônea força o Internet Explorer a entrar no modo quirks (um modo não-compatível com os padrões). Além disso, apesar da presença de um prólogo, todos os navegadores ainda tratarão os arquivos XHTML como se fossem arquivos HTML simples, a menos que o servidor web que os despachou o faça com o cabeçalho HTTP content-type apropriado de *application/xhtml+xml*, em vez do padrão *text/html*. Veja *http://hixie.ch/advocacy/xhtml*.

vez disso, para criar texto na tela você usa o elemento `<text>` e para agrupar elementos, você usa o elemento `<g>`.

Aqui está um exemplo extremamente básico do que uma imagem SVG que contenha o texto "Hello world!" em letras vermelhas se pareceria. Tudo que é necessário é inserir nosso elemento `<text>` na marcação do documento com os atributos apropriados, como esse:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <text x="50" y="50" style="fill: red">Hello world!</text>
</svg>
```

Isso resulta na imagem exibida na figura 1.3. Os atributos `x` e `y` do elemento `<text>` definem a posição inicial do texto na imagem. O atributo `style` usa CSS para tornar vermelho o conteúdo do elemento.

Figura 1.3 – Um exemplo simples e minimalista de SVG estilizado.

> Sim, você aí atrás com a mão levantada. "Atributos de elemento que definem posicionamento visual? Pensei que o CSS servisse para isso," ouço você perguntando. Sim, é para isso que serve o CSS, mas lembre-se de que o SVG é uma linguagem que descreve gráficos. Poderia-se argumentar, portanto, que o posicionamento é um metadado do texto, e, portanto, faz sentido usar a própria semântica da linguagem de marcação para defini-lo. Além disso, como você verá no próximo capítulo que o CSS não é apenas, ou necessariamente, uma linguagem de estilização visual. Na verdade, o atributo `style` que define a propriedade `fill` neste exemplo também poderia ser completamente substituído pela propriedade `fill`.

Até agora, está tudo muito simples. A parte interessante está na maneira que o CSS foi usado para colorir de vermelho o texto. Em vez de usar a propriedade `color`, usamos a propriedade `fill`. Isso acontece porque o SVG é gráfico e, portanto, a semântica mudou radicalmente daquela usada para desenvolver em XHTML.

Aqui está outro exemplo ilustrativo. Para fazer o texto se parecer com um tipo de botão em forma de pílula, você poderia usar o seguinte código:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect width="150" height="50" x="20" y="20" rx="15" ry="25"
    style="fill: teal; stroke: #000; stroke-width: 3;" />
  <text x="50" y="50" style="fill: red">Hello world!</text>
</svg>
```

O efeito desse código é mostrado na figura 1.4.

Figura 1.4 – Uma forma retangular, definida com o elemento <rect>, é tudo que é necessário para criar um plano de fundo.

No entanto, mesmo com essas semânticas visuais, a estrutura ainda é importante. Se isso fosse um botão de verdade em um gráfico SVG real interativo, então tanto o texto quanto o plano de fundo poderiam conceitualmente ser tratados como um objeto lógico. É por essa razão que o SVG define o elemento `<g>` para especificar grupos de elementos a serem tratados como um único objeto.

Semelhante ao XHTML, todos os elementos podem ter atributos `class` e `id` para identificá-los, então podemos usar um atributo para identificar nosso botão de exemplo. O exemplo agrupado não muda na tela, mas agora ficou assim no código:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <g id="example">
    <rect width="150" height="50" x="20" y="20" rx="15" ry="25"
      style="fill: teal; stroke: #000; stroke-width: 3;" />
    <text x="50" y="50" style="fill: red">Hello world!</text>
  </g>
</svg>
```

Como você pode ver, existe uma miríade de formatos XML em uso corrente atualmente. Cada aplicação XML define semânticas que se relacionam com seus propósitos. Ao usar namespaces XML, os desenvolvedores podem importar a semântica de uma aplicação XML e usá-la dentro de seus próprios documentos, criando infinitas possibilidades para descrever conteúdo de maneiras incrivelmente ricas.

No entanto, todas essas possibilidades permanecerão como mero potencial se não conseguirmos colocar esses documentos semânticos e estruturados nas mãos dos usuários. Para isso, precisamos entender como funcionam as ferramentas de software que os usuários usam para acessar o nosso conteúdo. Tais ferramentas são chamadas de agentes de usuários, e, bem ou mal, eles são os porteiros entre nossos usuários e nosso conteúdo.

Agentes de usuário: nossos olhos e ouvidos no ciberespaço

Um agente de usuário é nada além de uma entidade que age no lugar do usuário.[6] Isso significa que é importante compreender tanto os usuários quanto seus agentes. Agentes de usuário são as ferramentas que usamos para interagir com as inúmeras possibilidades existentes na Internet. São como extensões de nós mesmos. De fato, eles são (cada vez mais literalmente) nossos olhos e ouvidos no ciberespaço.

Entendendo os usuários e seus agentes

Desenvolvedores web já estão familiarizados com vários agentes de usuário comuns: os navegadores web! Somos até notórios por algumas vezes lamentarmos sobre o grande número de navegadores já existentes. Talvez precisemos reavaliar por que fazemos isso.

Existem vários tipos diferentes de usuários por aí, cada um com necessidades potencialmente e radicalmente diferentes. Portanto, para entender por que existem tantos agentes de usuário, precisamos entender quais são as necessidades de todos esses usuários. Isso não é simplesmente um exercício teórico. O fato é que compreender as necessidades do usuário nos ajuda a apresentar nosso conteúdo para ele da melhor maneira possível.

Apresentar o conteúdo de forma apropriada para os usuários e, por extensão, para seus agentes de usuário está além do argumento típico sobre acessibilidade, que afirma a importância de tornar seu conteúdo disponível para todos (apesar de que certamente usaremos esse argumento também). Os princípios por trás da compreensão das necessidades do usuário são muito mais importantes do que isso.

6 Tal definição é propositalmente ampla porque não estamos falando apenas de páginas web aqui, mas de todos os tipos de tecnologia. Os princípios são universais. No entanto, existem definições mais exatas disponíveis. Por exemplo, a W3C inicia a especificação do HTML 4 com algumas definições formais, incluindo o que é um "agente de usuário". Veja *http://www.w3.org/TR/REC-html40/conform.html*.

Você deve se lembrar de que a web apresenta dois desafios fundamentais. Um desafio é que qualquer fragmento de conteúdo dado, um único documento, precisa ser apresentado de múltiplas maneiras. Esse é o problema que o CSS foi projetado para resolver. O outro desafio é o inverso: vários tipos diferentes de conteúdos precisam ser disponibilizados, cada tipo necessitando de uma apresentação semelhante. Isso é o que o XML (e sua linguagem companheira de "folha de estilos", o XSLT) foi projetado para resolver. Portanto, combinar as funcionalidades poderosas do CSS e do XML é o caminho que devemos tomar para compreender, tecnicamente falando, como resolver esse problema e apresentar conteúdo aos usuários e seus agentes de usuário.

Já que um agente de usuário específico é apenas uma ferramenta para um usuário específico, a forma que o agente de usuário assume depende das necessidades do usuário. Na semântica formal dos casos de uso, tais usuários são chamados de atores, cujas necessidades podemos descrever determinando os passos que eles devem seguir para realizar algum objetivo. De forma semelhante, em cada caso de uso, certa ferramenta (ou ferramentas) usada para realizar tais objetivos define o que o agente de usuário é naquele cenário em particular.[7]

Um exemplo simples disso é que, quando Joe fica on-line para ler sobre as últimas notícias de tecnologia do Slashdot, ele usa um navegador web para fazer isso. Joe (nosso ator) é o usuário, seu navegador web (seja lá qual for que ele escolheu para usar) é seu agente de usuário, e ler as últimas notícias é o objetivo. Isso é uma interação bem tradicional, e em tal cenário podemos seguramente fazer algumas suposições sobre como Joe, sendo humano e tal, lê notícias.

Agora, vamos visualizar um cenário mais estranho para desafiar nossa compreensão do princípio. Joe precisa ir às compras para reabastecer sua geladeira e ele prefere comprar os itens de que precisa dirigindo o mínimo possível devido ao aumento no preço da gasolina. É por isso que ele possui a (fictícia) Frigerator2000, uma geladeira conectada à web que mantém indicações sobre as quantidades em estoque das quitandas e supermercados mais próximos, ajudando Joe a planejar sua rota. Isso o ajuda a evitar dirigir até uma loja onde ele não será capaz de comprar os itens de que necessita.

Se isso soa demais como ficção científica para você, pense novamente. Isso é uma aplicação diferente do mesmo princípio usado por leitores de feeds, mas, em vez de agregar artigos de notícias de websites, estamos agregando quantidades em estoque das quitandas. Todo o necessário para tornar isso uma realidade seria um formato XML para descrever os níveis de estoque das lojas, um pouco de software embutido, uma placa de rede em uma geladeira e algumas quitandas com experiência em tecnologia para publicar tal conteúdo na Internet.

7 Em casos de uso reais, jargão técnico e ferramentas específicas, como um navegador web, são omitidos porque eles são usados para definir os requisitos de um sistema, não sua implementação. Apesar disso, a noção de um ator e de seus objetivos é útil para compreender o "usuário" misterioso e o software desse usuário.

No entanto, nesse cenário o nosso agente de usuário é radicalmente diferente do navegador web tradicional. Ele é uma geladeira! É claro, não existem (ainda) agentes desse tipo espalhando-se pela Web atualmente, mas existem vários agentes de usuário que não são navegadores web e que fazem exatamente isso.

Mecanismos de busca como o Google, Yahoo! e Ask são, provavelmente, os exemplos mais famosos de usuários que não são pessoas. Tais empresas têm programas automatizados, chamados *spiders* (aranhas), que "rastejam" (*crawl*) pela Web indexando todo o conteúdo que conseguem encontrar. Diferente dos humanos, e muito semelhante ao nosso agente de usuário baseado em uma geladeira, tais spiders não podem observar o conteúdo com seus próprios olhos, ou ouvir o áudio com seus próprios ouvidos, então suas necessidades são muito diferentes de alguém como Joe.

Há ainda outros sistemas de vários tipos que existem para nos permitir interagir com websites e eles, também, podem ser considerados como agentes de usuário. Por exemplo, vários websites fornecem uma API que expõe alguma funcionalidade na forma de serviços web. O Microsoft Word 2008 é um exemplo de uma aplicação desktop que você pode usar para criar artigos de blogs em softwares de blogging como o WordPress ou MovableType, pois ambas as ferramentas de blogging suportam a API MetaWeblog, uma especificação XML-RPC.[8] Nesse caso, o Microsoft Word pode ser considerado um agente de usuário.

Como foi mencionado anteriormente, as várias encarnações dos leitores de notícias existentes são uma outra forma de agente de usuário. Vários navegadores web e aplicações de e-mail, como o Mozilla Thunderbird e o Apple Mail, também fazem isso.[9] Leitores de feeds fornecem uma maneira particularmente interessante de se examinar o conceito de agente de usuário porque atualmente existem vários web sites de leitura de feeds, como o Bloglines e o Google Reader. Se Joe abrir seu navegador web e logar-se em sua conta no Bloglines, o navegador web do Joe será o agente de usuário e Joe será o usuário. No entanto, quando Joe lê os feeds de notícias aos quais ele está inscrito no Bloglines, o servidor do Bloglines vai buscar os feeds formatados em RSS ou Atom no site-fonte. Isso significa que, do ponto de vista do site-fonte, *Bloglines.com* é o usuário e o processo de servidor do Bloglines é o agente de usuário.

Chegar a essa compreensão significa que, como desenvolvedores, podemos entender os agentes de usuário como uma abstração dos objetivos de um usuário específico, bem como de suas funcionalidades. Isso, claro, é uma definição intencionalmente

8 XML-RPC é um termo que se refere ao uso de arquivos XML que descrevem chamadas a métodos e dados transmitidos via http, tipicamente utilizados por sistemas automatizados. Este, portanto, é um excelente exemplo de uma tecnologia que tira vantagem das funcionalidades de serialização de dados do XML, e que muitos consideram como o precursor das técnicas Ajax usadas atualmente.

9 Na verdade, foi da tecnologia de e-mails, que é muito mais antiga, que surgiu o temo agente de usuário; um programa-cliente de e-mails é mais tecnicamente chamado de agente de usuário de correio (Mail User Agent, ou MUA).

vaga porque é tecnicamente impossível para você, como desenvolvedor, prever as funcionalidades presentes em qualquer agente de usuário em particular. Este é um desafio sobre o qual falaremos bastante ao longo de todo o livro, por ser uma das características que define a Web como um meio de publicação.

No entanto, em vez de essa falta de previsibilidade ser um problema, a restrição de não saber quem ou o que acessará nosso conteúdo publicado é, na verdade, uma coisa boa. Acontece que uma marcação bem projetada também é uma marcação ignorante de seu usuário, pois ela focaliza apenas na descrição de si própria. Você poderia até mesmo chamá-la de narcisista.

Por que dar controle ao usuário não significa desistir

Falar de marcação autodescritiva é apenas uma outra forma de falar sobre marcação semântica. Nesse paradigma, o conteúdo no documento recebido é estritamente segregado de sua apresentação final. Apesar disso, em algum momento o conteúdo precisa ser, de alguma forma, apresentado ao usuário. Se a informação sobre como fazer isso não é fornecida pela marcação, então onde ela está e quem decide o que ela é?

De primeira, com certeza você ficaria tentado a dizer que essa informação está na folha de estilos do documento e que é o desenvolvedor do documento quem decide o que ela é. Como você verá com detalhes no próximo capítulo, essa resposta está apenas, na maior parte, correta. Em cada caso, no final das contas é o agente de usuário quem determina quais estilos (em quais folhas de estilo) serão aplicados à marcação que ele recebe. Além disso, vários agentes de usuário (especialmente os navegadores web modernos) permitem que os próprios usuários modifiquem mais ainda as regras de estilo que serão aplicadas ao conteúdo. No fim, você pode apenas influenciar – não controlar – a apresentação final.

Apesar de parecer surpreendente para alguns, esse modelo, na verdade, faz todo o sentido. Permitir aos usuários o controle final da apresentação do conteúdo ajuda a garantir que você consiga atingir a possível necessidade de cada usuário. Ao usar CSS, autores de conteúdo, publicadores e desenvolvedores – ou seja, você – podem fornecer folhas de estilo que facilmente acomodariam, digamos, 80 por cento das necessidades de 90 por cento dos usuários. Mesmo no cenário mais otimista, casos extremos dos quais você nem teria como ter consciência ainda lhe escaparão, não importa o quão duro você trabalhe para tentar acomodar as necessidades de todo mundo.[10] E mais: mesmo que você tenha essa capacidade de recursos ilimitados, talvez você não saiba qual a melhor maneira de aperfeiçoar a situação para aquele usuário. Com tudo isso

10 Como era de se esperar, esse é o mesmo argumento que os defensores do software open-source fazem a respeito do porquê de tais softwares open source normalmente serem bem-sucedidos em corresponder às expectativas de mais usuários do que os sistemas proprietários de código fechado, controlados apenas por uma única empresa com (por definição) recursos relativamente limitados.

em mente, quem melhor para determinar a apresentação de um documento XML que precisa ser apresentado de maneiras bem específicas do que os próprios usuários que possuem tais necessidades específicas?

Um exemplo comum dessa situação na vida real poderia ocorrer no caso de Joe ser daltônico. Se ele quisesse visitar algum site de notícias onde os links dentro dos destaques do artigo fossem muito semelhantes ao plano de fundo do destaque, ele poderia não perceber que tais elementos, na verdade, são links. Por sorte, como o navegador do Joe permite a ele definir um website com sua própria folha de estilos, ele pode alterar a cor desses links para algo mais legível. Se o CSS não fosse projetado com isso em mente, seria impossível para Joe personalizar a apresentação desse site de notícias, otimizando-o para o seu caso particular.

Para muitos designers que vêm de indústrias tradicionais, como o design de mídia impressa, o fato de os usuários poderem alterar a apresentação de seu conteúdo é um conceito alarmante. Apesar disso, esta não é apenas a forma como a Web foi concebida para funcionar; esta é a única maneira em que poderia ter funcionado. Filosoficamente, a Web é uma tecnologia que coloca o controle nas mãos dos usuários. Portanto, nossa função como web designers é julgar as diferentes necessidades das pessoas como tendo a mesma importância, e não podemos fazer isso se tratarmos todos os usuários exatamente da mesma forma.[11]

Abstraindo a apresentação do conteúdo com CSS

Até agora, conversamos muito sobre conteúdo, marcação e agentes de usuário. Então onde o CSS se encaixa nisso? CSS é uma *camada de apresentação* para seu conteúdo.

O CSS utiliza a marcação existente para criar uma apresentação. Ele também pode reutilizar os mesmos elementos para apresentar conteúdo de maneiras especializadas, dependendo do tipo de agente de usuário que estiver consumindo-o. O CSS não se limita a apresentar conteúdo visualmente. Separar a apresentação do conteúdo em si abre várias portas. Uma delas é ser capaz de reestilizar o mesmo conteúdo mais tarde. Outra é poder criar simultaneamente diferentes apresentações do mesmo conteúdo.

Isso é importante porque diferentes agentes de usuário interpretarão seu CSS para apresentar o conteúdo da maneira que for mais apropriada para ele. Como o conteúdo pode ser transmitido pelos agentes de usuário de diversas formas para pessoas ou outras máquinas, o CSS abstrai os mecanismos específicos, ou "mídias", pelos quais isso ocorre. É assim que um mesmo conteúdo, com a mesma marcação subjacente, pode ser apresentado em várias mídias, incluindo o monitor conectado ao seu computador, a impressora na sua mesa, seu telefone celular, um leitor de tela *text-to-speech*, entre outros.

11 Essa filosofia é incorporada no estudo formal de ética, que é um tópico obrigatório para nós, desenvolvedores CSS, considerando a vastidão das implicações do que descrevemos aqui.

O CSS define essa abstração com base na mídia em uso, e fornece uma maneira de detectá-la. Isso ocorre por meio do conceito de tipo de mídia CSS, então vamos começar por aí. Após isso, descreveremos brevemente uma extensão para esse conceito chamado consultas de mídia CSS.

A natureza da saída: agrupando saídas com tipos de mídia CSS

A forma como um agente de usuário exibe o conteúdo depende da mídia-alvo a que o conteúdo é destinado. O propósito do CSS sempre foi fornecer uma maneira de descrever a apresentação de algum conteúdo de forma padronizada. No entanto, apesar de seu objetivo de permanecer independente da implementação, o CSS ainda precisa ter alguma noção de quais serão as propriedades físicas do meio de apresentação.

Para implementar isso, a especificação do CSS2.1 identifica dez características de paradigmas de interação diferentes que, juntos, formam quatro *grupos de mídias*. Essas características são coisas como saber se o conteúdo de um documento deve ser dividido em blocos discretos (como em páginas impressas) ou se ele pode ser apresentado de uma só vez, sem um limite teórico (como dentro da área de visualização de um navegador web com barras de rolagem). Essas duas características opostas formam um dos grupos de mídias.

Cada um dos *tipos de mídias* simplesmente define características específicas para cada um dos quatro grupos de mídias. Portanto, várias das propriedades CSS são acopladas a um tipo específico de mídia, já que nem todas as propriedades podem ser aplicadas a todos os tipos de mídias. Por exemplo, mecanismos de exibição monocromáticos nem sempre fazem uso completo da propriedade `color` (apesar de vários tentarem, de qualquer forma). De maneira semelhante, agentes de usuário puramente visuais não fazem uso da propriedade `volume`, que é utilizada em apresentações auditivas e, portanto, é amarrada a tipos de mídias que incorporem funcionalidades auditivas.

Os nove tipos de mídias definidos pelo CSS2.1 são `braille`, `embossed`, `handheld`, `print`, `projection`, `screen`, `speech`, `tty` e `tv`. A tabela 1.1, retirada da especificação do CSS2.1, exibe os relacionamentos entre grupos de mídias e tipos de mídias.

Tabela 1.1 – Relacionamentos entre Grupos de Mídias e Tipos de Mídias

Tipo de ídia	Contínuo/ Paginado	Visual/Auditivo/ Oral/Tátil	Grade/Bitmap	Interativo/Estático
`braille`	Contínuo	Tátil	Grade	Ambos
`embossed`	Paginado	Tátil	Grade	Estático
`handheld`	Ambos	Visual, auditivo, oral	Ambos	Ambos
`print`	Paginado	Visual	Bitmap	Estático
`projection`	Paginado	Visual	Bitmap	Interativo

Tipo de ídia	Contínuo/ Paginado	Visual/Auditivo/ Oral/Tátil	Grade/Bitmap	Interativo/Estático
`screen`	Contínuo	Visual, auditivo	Bitmap	Ambos
`speech`	Contínuo	Oral	N/D	Ambos
`tty`	Contínuo	Visual	Grade	Ambos
`tv`	Ambos	Visual, auditivo	Bitmap	Ambos

Além do paradigma contínuo/paginado, temos:

- Um paradigma visual/auditivo/oral que especifica de forma ampla quais sentidos humanos são utilizados para consumir o conteúdo.
- Um paradigma de grade/bitmap que especifica duas categorias amplas de tecnologias de exibição.
- Um paradigma interativo/estático que define se uma mídia de saída é ou não é capaz de atualizações dinâmicas.

Cabe aos agentes de usuário implementarem individualmente o suporte às funcionalidade que assume-se estarem presentes em um grupo de mídias específico. Além disso, cabe às implementações individuais o reconhecimento de um tipo de mídia específico e a aplicação de seus estilos.

Resumidamente, o direcionamento de cada um dos tipos de mídia é apresentado a seguir:

- `screen` (tela) é direcionado a todas as variedades de telas de computador, mas certos navegadores web mais novos encontrados em dispositivos móveis, como o Apple iPhone, também utilizam esse tipo de mídia. Na maior parte do tempo, esse é o tipo de mídia a que as pessoas se referem quando falam sobre web design.
- `print` (impressão) é direcionado a impressoras, ou para saída de arquivos de impressão (ou seja, PDF, PostScript), e também ocorre nos diálogos de visualização de impressão da maioria dos navegadores. Discutiremos o tipo de mídia `print` como muito mais detalhes no capítulo 4.
- `handheld` (portátil) é direcionado para telefones celulares, PDAs, smartphones e outros dispositivos que normalmente caberiam na sua mão ou no bolso. Esses dispositivos geralmente compartilham um conjunto de características, como CPU e recursos de memória limitados, largura de banda restrita e tamanho de tela reduzido, o que torna muito útil poder almejá-los com seu próprio tipo de mídia. Exceções notáveis surgem na forma de versões móveis do WebKit e em versões mais recentes do Opera, que elegeram o uso do tipo de mídia `screen` em vez do `handheld`.

- `aural` (auditivo) é direcionado para folhas de estilo que descrevem saídas do tipo *text-to-speech* (texto-para-fala) e costuma ser implementado em tecnologias de acessibilidade, como softwares de leitura de tela. No rascunho atual das especificações do CSS3, esse tipo de mídia está sendo deixado de lado em favor de um novo tipo de mídia denominado *speech* (oral).
- `braille` é direcionado para leitores Braille, atualmente a única forma de dispositivo de resposta tátil para o qual o CSS foi desenvolvido.
- `embossed` (em relevo) é semelhante ao `braille`, mas é direcionado a impressoras Braille *paginadas* (em vez de *contínuas*). Esse tipo de mídia poderia, teoricamente, ser usado para almejar impressoras Braille, apesar de não conhecermos agentes de usuário que implementem essa funcionalidade.
- `projection` (projeção) é direcionado para sistemas de projeção no alto (*overhead*), geralmente utilizado em casos como apresentação de slides. Semelhante aos dispositivos móveis, os dispositivos de exibição em projeção compartilham um conjunto de características que os torna únicos, como resoluções de tela mais baixas e profundidade de cor limitada. Se for especificada a aplicação de estilos a esse tipo de mídia, o Opera alterna dos estilos de `screen` para os estilos de `projection` no momento em que entra no modo de tela cheia ou no modo quisoque.
- `tv` é direcionado para exibição em televisões. Ainda estamos para ver esse tipo implementado de verdade em um agente de usuário, já que, tal como os portáteis e projetores, o hardware das televisões constitui um outro tipo de dispositivo com propriedades de exibição únicas, e seria extremamente útil se dispositivos como consoles de videogames ou TVs inteligentes adotassem esse tipo.
- `tty` é direcionado para dispositivos de exibição que usem uma grade de caracteres com largura fixa, como terminais e teletipos. Semelhante ao `embossed`, desconhecemos agentes de usuário que suportem esse tipo de mídia, e seu uso está se tornando cada vez mais anacrônico.
- `all` (todos) é o tipo de mídia usado quando um tipo de mídia não é especificado, e qualquer estilo aplicado a esse tipo de mídia também é aplicado a todos os outros tipos de mídias.

Tipos de mídias são especificados usando-se o atributo `media` em um elemento `<link>` ou `<style>` ou eles podem ser aplicados usando-se regras de CSS `@media` dentro das folhas de estilos. Aqui está um exemplo que usa o elemento `<link>` para referenciar uma folha de estilos a todos os tipos de mídias:

```
<link rel="stylesheet" href="default.css" type="text/css" media="all">
```

E aqui está um exemplo que referencia uma folha de estilos embutida ao tipo de mídia `handheld`:

```
<style type="text/css" media="handheld">
  div.foo { color: red; }
</style>
```

O uso de `@media screen` aplica os estilos aos vários dispositivos de exibição tradicionais:

```
<style type="text/css">
  @media screen {
    div.foo { color: red; }
  }
</style>
```

A regra CSS `@import` também pode usar um parâmetro opcional de tipo de mídia para importar uma folha de estilos especificando a mídia-alvo (`print`, nesse caso):

```
<style type="text/css">
  @import url(print.css) print;
</style>
```

Considerações de direcionamento aos tipos de mídias

Ao compor folhas de estilos, a maioria dos desenvolvedores CSS ainda tende a pensar em termos de alguns poucos agentes de usuário utilizados em um único meio: navegadores web em computadores desktop tradicionais. No entanto, como acabamos de ver, o CSS pode ser usado em uma arena muito mais ampla do que a sugerida por esse escopo limitado. Por que se limitar ao princípio de um projeto? Com um pouco de planejamento e atenção, você pode gerar uma presença web muito mais utilizável, ampla e bem-sucedida, considerando possibilidades para todos os tipos de mídias desde o início.

Referenciando telas

Os tipos de mídia `screen`, `handheld` e `projection` são um tanto semelhantes por terem a intenção de serem apresentados visualmente usando-se tecnologias de visualização que emitam luz. O tipo de mídia `handheld` é tipicamente uma versão reduzida do design para o tipo `screen`, ao passo que o tipo de mídia `projection` é normalmente – mas não necessariamente – usado para formatar apresentações em slide. Apesar de serem semelhantes, ainda existem diferenças distintas entre esses tipos de mídias, e um exame mais aproximado deles ilustrará como os diferentes grupos de mídias aos quais um tipo de mídia se refere influenciam nas decisões e possibilidades de design.

Com o influxo crescente de agentes de usuário no mercado, faz mais sentido discutir os mecanismos de exibição (*rendering engines*) dos agentes de usuário do que discutir especificamente os produtos ao usuário final, já que a capacidade de transmissão de marcações e CSS de cada mecanismo é semelhante do ponto de vista dos produtos que o utilizam. Os quatro principais mecanismos de exibição existentes atualmente no mercado são:

- **Trident**, que é usado em todas as versões do Internet Explorer.
- **Gecko**, usado nos produtos que utilizam a base de código do Mozilla (como o Firefox, Flock, Camino e outros derivados).
- **WebKit**, que foi originalmente desenvolvido como KHTML para o navegador web Konqueror do Linux e que agora é usado no Safari e em vários smartphones da Nokia, entre outros produtos.
- **Presto**, desenvolvido e usado pela linha de produtos do Opera.

Portanto, no restante deste livro, ao discutirmos um mecanismo de exibição em particular, você poderá assumir com segurança que estaremos falando sobre a maioria dos agentes de usuário que utilizam tal mecanismo. Inversamente, ao discutirmos sobre um agente de usuário específico, você poderá assumir com segurança que o comportamento de exibição dos outros navegadores que utilizam o mesmo mecanismo de exibição será semelhante.

O tipo de mídia screen

Uma característica do tipo de mídia `screen` é que ele é *contínuo*, o que significa que o conteúdo fluirá de maneira ininterrupta para além do limite inferior de uma área de visualização determinada. Como resultado desse comportamento, a largura do layout torna-se a preocupação primária, e decisões de como o site será composto em cenários com diferentes larguras devem ser feitas. Em contraste, você não precisa dar muita atenção para a altura do layout, já que a página pode ser tão longa quanto for necessário para que o conteúdo se encaixe verticalmente dentro dela.

As larguras de telas podem variar enormemente de um usuário para outro. Além disso, a janela de um navegador web normalmente pode ser redimensionada para qualquer largura que o usuário desejar, então é melhor que o seu design seja o mais flexível possível com relação às resoluções e larguras de navegador que ele consegue acomodar. Uma forma de se conseguir isso com um layout de largura variável, ou "líquido", que expanda e contraia para preencher qualquer espaço disponível. No entanto, mesmo que o seu design consiga ser flexível dessa maneira, normalmente é necessário que alguns elementos (como as imagens) mantenham dimensões de largura fixa, então em algum momento certos julgamentos deverão ser feitos.

Conforme as resoluções de exibição foram evoluindo, diferentes larguras têm sido usadas como base da grade de layout. Recentemente, tem havido uma tendência no sentido de se utilizar uma largura-base de 960 pixels, um valor que funciona bem, considerando-se que a maioria dos monitores atuais tem resoluções de exibição com largura de 1024 pixels ou mais. A largura de 960 pixels não só acomoda elementos do navegador, como as barras de navegação, como também é divisível por 3, 4, 5, 6, 8, 10, 12, 15 e 16, permitindo um grande número de possibilidades para se implementar vários layouts com base em grades tão estreitas quanto for necessário.[12]

O tipo de mídia projection

O tipo de mídia `projection` é interessante em comparação ao tipo de mídia `screen` porque, apesar de almejar uma tecnologia de exibição semelhante, é considerado um tipo *paginado*, em vez de *contínuo*. Em outras palavras, em vez de ter o conteúdo rolando infinitamente além do limite inferior da área de visualização, você pode especificar que o conteúdo seja dividido em partes discretas que serão, ostensivamente, tão altas quanto a resolução do projetor permita. Então, para navegar pelo conteúdo do documento, você se movimenta para frente e para trás de um pedaço a outro, de maneira separada.

De todos os tipos de mídias relacionados a telas, `projection` é o menos suportado e o menos utilizado. O Opera é um dos navegadores que tem algum suporte para esse tipo de mídia com uma funcionalidade denominada Opera Show. Quando invocado pressionando-se Alt+F11, ou escolhendo **View ➤ Full Screen** a partir do menu, o Opera Show expandirá a área de visualização do navegador para que ocupe a largura total do dispositivo em que estiver sendo executado. No entanto, isso é mais do que um modo de visualização de "tela cheia", pois com uma única regra CSS você pode transformar os blocos contínuos de seu documento existente em itens paginados, resultando em slides de apresentação semelhantes aos que você teria ao usar o Microsoft PowerPoint ou o Apple Keynote:

```
.slide {
  page-break-after: always;
}
```

Com essa regra CSS aplicada, todo elemento de bloco que use a classe `.slide` será exibido como um elemento discreto (um "slide") quando visualizado no modo Opera Show. Use a tecla Page Down para mover para o próximo slide do maço de slides e use Page Down para retroceder um slide. Isso cria a oportunidade de lhe dar uma

12 Cameron Moll postulou o uso de 960 pixels em um blog intitulado "Optimal width for 1024px resolution?" ("Melhor largura para resolução de 1024px?") publicado em *http://www.cameronmoll.com/archives/001220.html*. Tanto o artigo quanto os comentários são uma leitura interessante para qualquer pessoa que fique imaginando como a comunidade padroniza tais números aparentemente arbitrários.

ferramenta de apresentação aberta e não-proprietária que é facilmente portável e que não travará seus dados – uma funcionalidade interessante que desejamos que mais navegadores suportem![13]

Também existem outras possibilidades além dos slides de apresentação. Qualquer coisa que funcione bem na tela quando apresentada em partes separadas que você expõe incrementalmente, em vez de expor tudo de uma só vez para que seja rolado, pode fazer uso do tipo de mídia `projection`. Por exemplo, dispositivos de exibição de menor escala, como leitores de e-book, poderiam simular a experiência de "virar páginas" com esse tipo de mídia, enquanto tira vantagem do conteúdo *interativo* e da estilização, o que não seria possível se usassem o tipo de mídia `print`, que é considerado *estático*.

O tipo de mídia handheld

De várias maneiras, o tipo de mídia `handheld` é o mais amorfo dentre os tipos de mídias relacionados a tela. Ele pode ser contínuo ou paginado, suporta interações visual, auditivo e oral; pode ser baseado em tecnologias de exibição em grade ou bitmap; e pode apresentar conteúdo estático ou dinâmico. Na verdade, o tipo de mídia `handheld` costuma ser usado para projetar telas pequenas que são encontradas nos dispositivos móveis, então sua aplicação típica tem sido linearizar e simplificar os estilos de tela de um design.

O tipo de mídia `handheld` também é o centro de algumas discussões acaloradas relativas aos tipos de mídias em geral, pois o Safari no Apple iPhone e as últimas versões do Opera Mobile começaram a ignorá-lo em favor do tipo de mídia `screen`. Entre outras críticas, Apple e Opera Software alegam que a maioria das folhas de estilos do tipo `handheld` não fornece uma experiência de usuário adequada às capacidades de seus dispositivos, quando comparadas às folhas de estilos do tipo `screen`, e, por isso, eles introduziram as consultas de mídia como uma extensão aos tipos de mídias, discutidas posteriormente neste capítulo. Apesar disso, o tipo de mídia `handheld` ainda prevalece e é útil no contexto de outros telefones celulares e PDAs.

Nos anos anteriores, projetar para dispositivos móveis era considerado um luxo, um adicional opcional, caso o tempo e o dinheiro permitissem. Este não é mais o caso. Como a web móvel começou a crescer rapidamente nos últimos anos, devotamos um capítulo inteiro deste livro para o desenvolvimento CSS em um contexto móvel (veja o capítulo 5).

13 Por sorte, Eric Meyer criou o Simple Standards-Based Slide Show System (Sistema Simples de Exibição de Slides Baseado em Padrões, ou S5), que reproduz e até mesmo expande as funcionalidades do Opera Show para o restante dos navegadores conscientes dos padrões existentes por aí. O S5 cria uma base excelente para a construção de apresentações independentes de plataforma usando o XHTML e o CSS que você já conhece. Ele pode ser encontrado em *http://meyerweb.com/eric/tools/s5/*.

Em seu livro *Mobile as 7th of the Mass Media* (2008, Futuretext Ltd.), Tomi T. Ahonen argumenta que a mídia móvel não deve ser ignorada: 31% dos gastos dos consumidores na indústria fonográfica são gastos em compras relacionadas a mídias móveis, ao passo que na indústria de jogos o número é de 20%. Deduziu-se que – no momento em que escrevemos – aproximadamente um bilhão e meio de conexões à Internet estão sendo geradas por telefones celulares e que 63% da população global tem um telefone celular potencialmente capaz de acessar a Internet. Cerca de 60 países em todo o mundo têm uma penetração de telefonia celular superior a 100% – o que significa que muitas pessoas possuem não um, mas dois dispositivos móveis. E finalmente, Nielsen, em maio de 2008, relatou que os sites líderes da Internet aumentaram seu tráfego em 13%, comparado ao tráfego provindo apenas de computadores desktop, e que, em certos casos, como meteorologia e entretenimento, o tráfego subiu até 20%. Isso representa indicadores significativos para o crescimento da web móvel, que, sem dúvidas, continuará a crescer nos próximos anos.

O tipo de mídia print

O tipo de mídia `print` é outra categoria familiar de trabalho com CSS para a maioria dos desenvolvedores. Esse tipo de mídia é implementado por agentes de usuário capazes de imprimir fisicamente em papel ou de gerar equivalentes eletrônicos em formatos como PDF ou PostScript.

Documentos impressos apresentam um conjunto de questões significativamente diferentes da saída em tela à qual estamos mais acostumados, pois o paradigma de design dentro de uma área de visualização dinâmica é substituído pela noção de uma área de página estática representando uma área física imprimível do papel. Em razão da natureza estática da mídia impressa, os designers perdem várias funcionalidades dinâmicas do CSS, como a pseudoclasse `:hover`, e precisam considerar formas alternativas de exibir as informações aos leitores.

As razões para se imprimir páginas web incluem legibilidade (imprimir um documento longo para diminuir o cansaço nos olhos), portabilidade (carregar cópias impressas para ler durante uma viagem) ou utilidade (como quando imprimimos formulários on-line que necessitam de assinaturas físicas). Usuários raramente necessitam de certas porções da página como a navegação, já que, obviamente, você não pode clicar em um hiperlink no papel, então isso deveria ser cortado. Além disso, informações suplementares como as que são comumente encontradas em barras laterais também deveriam ser removidas. Muitas dessas questões de transformação podem ser resolvidas com declarações CSS como `display: none` e definindo a largura do conteúdo principal para melhor preencher o espaço disponível no papel.

Todos os principais navegadores web modernos suportam o tipo de mídia `print`, então isso não é apenas útil, mas também fácil de implementar e testar. Discutiremos a mídia `print` em detalhes no capítulo 4.

Mídia auditiva

Ao ler um texto em um website, você escuta uma voz masculina ou uma voz feminina em sua mente? Desconhecida de vários web designers, o tipo de voz com que o conteúdo é lido em voz alta pelos agentes de usuário com capacidades *text-to-speech*, especificada pela propriedade `voice-family`, é uma das várias propriedades auditivas que o CSS oferece a você.

Na verdade, o CSS oferece um conjunto relativamente rico de propriedades auditivas para serem usadas pelos designers, incluindo propriedades auditivas espaciais para especificar a direção da qual um som provém, usando as propriedades `azimuth` e `elevation`, ênfase auditiva com as propriedades `stress` e `pitch` e até mesmo a velocidade de leitura, usando a propriedade `speech-rate`. Além disso, diferentes elementos podem receber "deixas" auditivas usando as propriedades `cue-before` e `cue-after`, para que toques específicos ou outros sons possam preceder links, informações de licença ou o texto alternativo de uma imagem. Isso é o equivalente auditivo da forma como certos ícones representam o significado de um elemento em apresentações visuais.

Os agentes de usuário auditivos mais comuns são os leitores de tela, mas eles representam apenas uma das classes de potenciais implementações. Como o nome indica, leitores de tela literalmente necessitam de uma tela que será lida em voz alta, e não é preciso ser um gênio para perceber que exigir que pessoas com deficiência visual usem uma interface visual não é a melhor solução para elas. Uma tela que emite luz não tem utilidade para pessoas que não conseguem enxergar, então dispositivos que focalizem em outras mídias, como o som, fazem muito mais sentido. E aqui é o ponto em que as folhas de estilos auditivas poderiam brilhar.

É claro que é inteiramente plausível que navegadores web auditivos também encontrem outros usos em outros mercados, como no caso de carros com acesso à web, sistemas caseiros de entretenimento e até mesmo em apresentações multimodais de conteúdo web tradicional.[14] Frequentemente usamos as funcionalidades de text-to-speech de nossos sistemas operacionais para escutar notícias e artigos longos em blogs enquanto realizamos as tarefas domésticas, então seria extremamente útil ganharmos mais controle sobre essa transformação na apresentação. Por exemplo, pausas mais longas entre um título e o texto subsequente poderiam ser inseridas para melhorar o entendimento, já que vários títulos não têm pontuação completa (como um ponto final) e, portanto, são lidos muito juntos do texto subsequente pela maioria dos programas text-to-speech.

14 Tais usos para navegadores web auditivos foram reconhecidos pela W3C já nos idos de 1999, que publicou uma nota técnica que sugeria adições à especificação CSS1 para suportar propriedades de estilo auditivas para que tais navegadores implementassem-nas. Veja *http://www.w3.org/Style/CSS/Speech/NOTE-ACSS*.

Navegadores com suporte a CSS auditivo são raros. O Opera com o Voice (disponível apenas para Windows 2000 e XP) tem o suporte mais completo da especificação. No entanto, podemos antever um dia não muito distante em que o suporte melhorará, abrindo possibilidades completamente novas que, até então, eram inimagináveis. Por exemplo, talvez blogs inteiros possam ser transformados em netcasts auditivos com a simples aplicação de um feed RSS e uma folha de estilos auditiva. Isso não seria interessante?

Detecção de funcionalidades por meio das consultas de mídia CSS

Apesar de a noção do CSS de tipos de mídias dar aos desenvolvedores CSS certa quantidade de controle sobre quais estilos serão aplicados em quais contextos de exibição, eles ainda acabam sendo amplos demais. Acredite ou não, os dispositivos com funcionalidades web atuais são ainda mais heterogêneos do que os dispositivos do ano passado. Novos formatos e novas tecnologias vêm desafiando alguns dos pressupostos que as especificações CSS atuais têm feito sobre tipos de mídias, em particular o tipo de mídia `handheld`. Desenvolvedores CSS precisam de maneiras mais precisas de determinar as funcionalidades dos agentes de usuário.

Isso é exatamente o que as *consultas de mídias*, introduzidas como parte da especificação[15] ainda em desenvolvimento do CSS3, procuram resolver. Consultas de mídia estendem a noção de tipos de mídias definindo um conjunto de *funcionalidades de mídia* que os agentes de usuário podem se propor a ter. O desenvolvedor CSS fornece um conjunto de condições como uma expressão, da qual farão parte um tipo de mídia e uma ou mais funcionalidades de mídia. Aqui está um exemplo de uma consulta de mídia que você pode utilizar atualmente, que importa uma folha de estilos externa apenas se o agente de usuário suportar o tipo de mídia `screen` e sua tela física for menor do que 481 pixels de largura:

```
<link type="text/css" rel="stylesheet" media="only screen and (max-device-width:480px)"
  href="webkit.css" />
```

De maneira conveniente, isso descreve a largura de um Apple iPhone na orientação de paisagem, bem como vários outros dispositivos móveis baseados no WebKit que existem atualmente no mercado. No exemplo anterior, a funcionalidade de mídia que está sendo pesquisada é `device-width`. Outras consultas poderiam ser direcionadas para a profundidade de cor, capacidade de exibir cores, aspect ratio e atributos semelhantes. Aqui está outro exemplo, que vincula duas folhas de estilos para impressão. Uma é específica para impressoras coloridas, ao passo que a outra trata das impressoras branco-e-pretas:

```
<link rel="stylesheet" media="print and (color)" href="print-color.css" />
<link rel="stylesheet" media="print and (monochrome)" href="print-bw.css" />
```

15 A especificação de consultas de mídia é um W3C Working Draft no momento em que escrevemos, e pode ser encontrado em *http://www.w3.org/TR/css3-mediaqueries*.

Como os valores de cor às vezes se igualam em saídas branco-e-pretas, agora temos uma maneira de definir especificamente um valor de contraste maior nas impressoras branco-e-pretas do que nas coloridas. Assim, o texto impresso ficará mais legível no caso de usuários que estejam imprimindo em tons de cinza, mas ainda podemos reter o escopo desejado de cores para usuários que estejam imprimindo a cores.

No momento em que escrevemos isso, os únicos dentre os principais navegadores que suportam consultas de mídia são o Safari 3, Konqueror 4 e versões do Opera superiores a 7. O principal uso das consultas de mídia tem ocorrido no direcionamento de estilos para dispositivos móveis que executem navegadores baseados no Opera ou WebKit. Conforme a especificação CSS3 evoluir e mais implementações aparecerem ao longo do tempo, esperamos ver uma adoção mais ampla das consultas de mídia entre os demais tipos de mídia.

Um documento, múltiplas faces

No passado, era muito comum que websites redirecionassem seus usuários para uma versão de um documento ou folha de estilos, caso estivessem usando certos navegadores web, ou para outra versão se estivessem usando outros navegadores. Isso não só era difícil de se realizar, mas também extremamente caro de se manter. Frustrações crescentes por parte dos desenvolvedores web eventualmente levaram ao abandono desses esforços em favor de advogar os padrões web, em que uma única versão do código poderia ser usado ao longo de todos os agentes de usuário que se conformassem a esses padrões.

Ironicamente, atualmente não é incomum que os websites forneçam apenas uma versão de suas páginas para visualização on-line com um navegador web baseado em desktop, uma página diferente para impressão e ainda outra versão para acesso por dispositivos móveis. Mais uma vez, muitos usam esquemas de detecção de agentes de usuário para tentar rotear o tráfego de acordo – um exercício de futilidade, considerando-se a natureza efêmera das strings de agentes de usuário nos cabeçalhos HTTP. Muitos também colocam links e botões em suas páginas com o texto "Imprima esta página" ou "Visualize usando acesso móvel".

Como era de se esperar, todos esses esforços são amplamente desnecessários, redundantes e muito caros de se manter. Essa funcionalidade pode ser substituída simplesmente pelas construções que já existem no CSS para fornecer o mesmo documento subjacente para todos os agentes de usuário. E mais, fazer isso aumenta a usabilidade do site porque a transformação de um formato a outro é transparente e automática. Já não existem mais razões para forçar seus usuários a navegar de forma dolorosa por um design baseado em desktop para encontrar aquele link de "acesso móvel".

Ao utilizar as funcionalidades que os tipos de mídia CSS e consultas de mídia fornecem, autores de conteúdo podem projetar folhas de estilos que serão usadas por agentes de usuário com base no próprio ambiente do agente. Como discutiremos nos próximos capítulos, consultas de mídia são a maneira recomendada de se apontar estilos para o Apple iPhone e iPod Touch, e elas funcionam incrivelmente bem com a multiplicidade de variações nos dispositivos móveis, no caso dos navegadores que as suportam. Para os navegadores que não as suportam, o JavaScript pode, por vezes, ser usado para aproximar tais comportamentos sem lançar mão da detecção de agente de usuário.[16]

Complementado a semântica com CSS

O CSS descreve como os elementos são estilizados, mas a *semântica* é definida na marcação em si. Apesar disso, o CSS é uma excelente maneira de se destacar o significado da semântica, de explorar as nuances da semântica e de fornecer confirmações no nível da apresentação a respeito do que um elemento deve representar. Para fins de ilustração, vamos tentar descrever "Sarah" como um ser humano usando o CSS.

```
#Sarah {
  /* Sarah possui 166 centímetros de altura. */
  height: 166cm;
  /* Ela tem pele branca e sardas. */
  background: white url(freckles.png);
}
#Sarah #hair {
  /* Ela é ruiva */
  color: red;
  /* E o cabelo possui uma largura de 150 centímetros. */
  height: 150cm;
}
#Sarah .eye {
  /* Os olhos dela são azuis */
  color: blue;
  /* e cada um tem uma largura de 2,2 centímetros. */
  width: 2.2cm;
}
```

É claro que você não pode descrever completamente qualquer conceito, muito menos nossa amiga Sarah, usando o CSS. A cor do seu cabelo faz parte integral de seu ser, de sua identidade? Talvez. Em parte, cabe ao autor do conteúdo determinar quais partes de sua publicação são conteúdo e quais são apresentação. A totalidade da experiência é a estrutura *e* a apresentação. Algumas vezes fica difícil distinguir entre o que seria um e o que seria outro.

16 Um exemplo impressionante disso pode ser encontrado no site do Cameron Adams. Adams escrevia sobre layouts independentes de resolução já nos idos de 2004. Seus artigos sobre o assunto estão disponíveis em *http://themaninblue.com/writing/perspective/2004/09/21/*.

O que podemos fazer é usar o CSS para que funcione em conjunto com a marcação semântica para ampliar e ilustrar mais ainda o significado de nosso conteúdo. Da melhor maneira possível, o CSS deve se esforçar para ser autodocumentado, claro, significativo e solidário à marcação para a qual foi projetado para representar.

Resumo

Neste capítulo, revisamos alguns dos princípios básicos e filosóficos sobre os quais a *World Wide Web* foi construída, então primeiro conversamos sobre linguagens de marcação. Você aprendeu porque a ideia de semântica do documento, popularizada por meio do hipertexto, encontra-se no núcleo da maioria das tecnologias web atuais. Exibimos exemplos de RSS e SVG, linguagens de marcação que destacam dois formatos de documento bem diferentes, sendo ambas baseadas na Linguagem de Marcação Extensível (XML).

A modularização que o XML fornece dá aos desenvolvedores algumas vantagens importantes sobre o HTML derivado do SGML para se escrever marcações de maneira semântica. Ao criar linguagens de marcação completas para propósitos especiais (aplicações XML) para se descrever coisas como feeds de notícias, gráficos vetoriais e outros tipos de conteúdos, torna-se possível à Web abrigar um ecossistema de informações inteiro contendo inúmeras coisas diferentes. Os namespaces XML permitem que múltiplas aplicações XML usem semânticas diferentes dentro de um único documento XML. Quando ferramentas de software "poliglotas" conseguem enxergar o uso desses dialetos XML adicionais, eles podem fazer mais pelos usuários.

Já que diferentes usuários possuem necessidades diferentes, a forma como ferramentas de software (agentes de usuário) realmente se comportam pode variar de um usuário a outro. Por isso, os desenvolvedores precisam reconhecer que eles estão limitados a influenciar – não controlar – a apresentação de seu conteúdo. Por sorte, dar o controle final nas mãos dos usuários é uma restrição que, na verdade, ajuda a manter publicada a semântica da marcação, encorajando uma separação apropriada dos interesses entre conteúdo e apresentação.

Neste capítulo, você aprendeu por que adicionar o CSS como uma camada de apresentação sobre a marcação subjacente é valioso do ponto de vista técnico e semântico. Discutimos como o CSS faz isso usando sua noção de tipos de mídia, pegando um único bloco de conteúdo e apresentando-o de várias maneiras diferentes. Por fim, você aprendeu como essa abordagem aumenta a acessibilidade e a reusabilidade (ou seja, novos propósitos) do conteúdo em si.

Com esses fundamentos conceituais em mente, vamos, em seguida, explorar o CSS em si com mais detalhes. Quanta influência você tem sobre a apresentação de um documento usando o CSS? Como veremos nos próximos capítulos, você tem muito mais controle do que imaginara previamente.