Capítulo 1

Web Design Responsivo

Arlino Henrique Magalhães de Araújo

Abstract

The growing variety of devices where the websites are shown (laptops, tablets, netbooks, mobile phones, small screen desktops, giant screens iMacs, second monitor, etc.) makes it impossible to develop a version of the same site for each existing device. Responsive web design is one of the solutions for this problem where a site is developed so that its elements adapt automatically, regardless of the device type and the use context.

Resumo

O crescimento da variedade de dispositivos onde os websites são visualizados (laptops, tablets, netbooks, celulares, desktops com tela pequena, iMacs com telas gigantescas, segundo monitor e etc.) torna inviável o desenvolvimento de uma versão do mesmo site para cada dispositivo existente. O web design responsivo é uma das soluções para esse problema onde um site é desenvolvido de forma que os elementos que o compõem se adaptem automaticamente independente do tipo de dispositivo e do contexto de uso.

1.1. Introdução

Em um passado não muito distante, era necessário ter um *desktop* para acessar a *internet* e grande parte dos computadores usava resoluções e navegadores muito semelhantes. Nessa época, bastava fazer um *site* que funcionasse no navegador Internet Explorer com resolução máxima de 1024×768 *pixels* e tudo estava resolvido. Existiam outras características a serem consideradas, mas a grande maioria dos usuários de *internet* estava enquadrada no mesmo grupo e, no máximo, fora desse grupo, haviam os usuários de Mozilla Firefox [4].

Durante anos, o comum na *Web* era a existência de *sites* com páginas de tamanhos fixos e já haviam discussões sobre o desenvolvimento *layouts* como, por exemplo, se a melhor largura de uma página seria 960 ou 980 *pixels*. As páginas eram feitas levando em consideração apenas a resolução mais comum e esquecendo as restantes. Os *designs* eram

copiados *pixel* a *pixel* do desenho original de um editor de imagens como o Photoshop, por exemplo.

Desde então o mundo evoluiu bastante e atualmente o acesso às páginas *web* através dos mais diversos aparelhos como, por exemplo, a *internet* através de TVs de 50", de celulares com telas de 2" até 5", de *tablets* de 6" até 11", de computadores com telas na margem entre 11" a 26", entre outros dispositivos. Esta característica demonstra que a *Web* é flexível e adaptável e que, consequentemente, um *site* pode ser visto de diversas formas em diversos contextos, o que impossibilita saber de qual dispositivo o usuário irá acessar um *site*.

O *Web Design* Responsivo (*Responsive Web Design* - RWD) permite sair das limitações de um *browser desktop* e seu tamanho previsível possibilitando construir páginas com flexibilidade que suportem todo tamanho de tela, qualquer tipo de resolução e *interfaces* com *touch* ou *mouse* [3]. A figura 1.1 mostra um exemplo de um *site* responsivo exibido em vários dispositivos diferentes. A mesma página do *site* é mostrada em um celular, em um *tablet*, em um monitor *desktop* e em um *notebook*; mas com *layouts* ligeiramente diferentes. O conteúdo visualizado nos dispositivos é o mesmo, o que difere em cada um deles é a disposição dos elementos do *site* organizados de modo a otimizar a *interface* com o usuário.

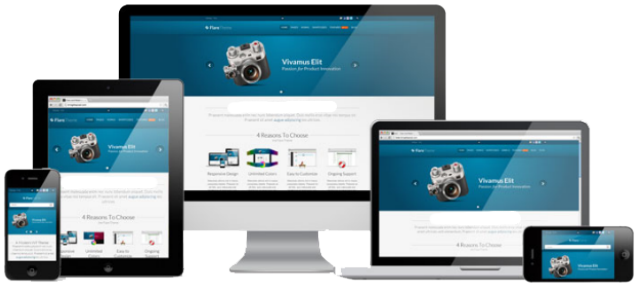

Figura 1.1. Website resposivo visualizado em quatro tipo de dispositivos diferentes.

1.2. Contextualização

1.2.1. HTML e CSS

Muitos dos sistemas utilizados atualmente estão na *web* e são executados através de navegadores, como redes sociais, portais, sistemas bancários e etc. As *interfaces* das páginas de sistemas ou de *sites* na *web* exibidas nos navegadores são feitas utilizando, basicamente, as linguagens HTML (*HyperText Markup Language* ou Linguagem de Marcação de HiperTexto) e CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) [2].

O HTML é uma linguagem de marcação de texto utilizada para estruturar conteúdos como textos, imagens, vídeos ou áudios em páginas *web*. Um documento HTML possui marcadores que são comandos utilizados para definir a formatação dos elementos. O CSS é voltado para a aparência da página (como cores, posicionamento ou tamanho de elementos), ou seja, serve para formatar as características visuais de conteúdos já estruturados pelo HTML.

1.2.2. Tipos de Medidas em CSS

Para formatar tamanhos, o CSS utiliza medidas. Os principais tipos de medidas utilizados em CSS são: *pixels*, pontos, porcentagens e ems.

O *pixel* é a unidade de medida fixa comumente considerada como um ponto indivisível na tela de exibição de um dispositivo, embora, atualmente existam vários tamanhos para um *pixel*. A unidade de medida ponto é tradicionalmente utilizada para CSS de impressão. Um ponto é igual a 1/72 polegadas. Assim como *pixels*, pontos são unidades de tamanho fixo [3].

Uma unidade de medida ”em” equivale ao tamanho atual da fonte do elemento-pai. Por exemplo, se o tamanho da fonte do elemento (o corpo de um documento HTML, por exemplo) é 12px então 1em é igual a 12px. A unidade porcentagem é muito parecida com a unidade ”em”, a principal diferença é o fato de que o atual tamanho da fonte na porcentagem é igual a 100%. Por exemplo, se o tamanho do elemento é 12px então o tamanho é 100% [3].

As unidades de medidas utilizadas no *design* responsivo são as porcentagens e os ems, pois são unidades relativas e escaláveis podendo se adaptar e manter relações de tamanho com outros elementos de um documento.

1.2.3. Viewport

O *viewport* é espaço disponível para a página no navegador. Ele depende do tamanho da janela do navegador (maximizado ou restaurado) e desconsidera as barras de ferramenta, rolagem, navegação e etc. [1]. A figura 1.2 mostra o *viewport* em diferentes dispositivos e em diferentes disposições.

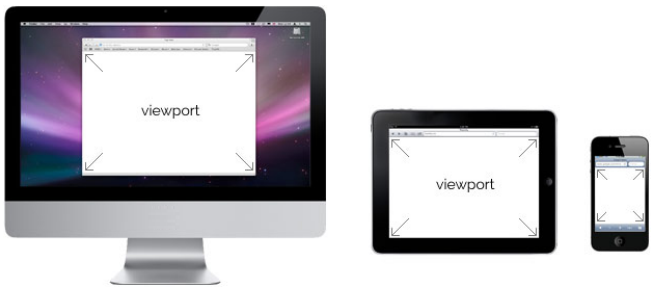

Figura 1.2. Viewport em três tipo de dispositivos diferentes.

O *viewport* muda de acordo com o *zoom*. Quando o usuário faz um *zoom* em *site*, os elementos ficam maiores e cabe menos informação na tela, ou seja, o espaço disponível para a página fica menor (o *viewport* diminui). O menor ponto da tela que pode receber uma cor é chamado de *Pixel* Físico (ou *Device Pixel*). E a medida escrita no código CSS é chamada de CSS *Pixel*. Na maioria dos casos um *pixel* físico é igual a um CSS *pixel*, até que seja feito um *zoom* [1]. Por exemplo, um *zoom* de 200% faz 300 CSS *pixels* serem renderizados como 600 *pixels* físicos (número de *pixels* físicos = número de CSS *pixels* $\times$ *zoom*).

Para um *site* de 980 CSS *pixels* feito para *desktop* ser visto inteiro em uma tela

de 320 *pixels* físicos de um *smartphone* o navegador faz um *zoom out* (diminui o *zoom*) na página. Assim, o usuário pode ver uma miniatura da página e para vê-la melhor é necessário fazer um *zoom in* (aumentar o *zoom*).

O *zoom* em *desktops* e *smartphones* é diferente. No *desktop*, o *zoom* aumenta o tamanho dos elementos e renderiza novamente toda a página. O *zoom* em *smartphones* dimensiona apenas a parte visível da página na tela e não altera o *design*. Para ver o restando da página é necessário fazer um gesto de ”arrastar e soltar”. Por isso, em *smartphones* há dois *viewports*: o *layout viewport* e o *visual viewport*. O *layout viewport* é toda a área disponível da página. O *visual viewport* é a parte da página visualizada na tela [1]. A figura 1.3 ilustra os dois tipos de *viewport*.

Figura 1.3. Layout viewport e Visual viewport.

1.3. Web Design Responsivo

Em 2000, John Allsopp publicou no *site* **A List Apart** o artigo **A Dao of Web Design** [6] onde ele já falava sobre a flexibilidade da *web*:

”O controle que os designers têm no meio impresso e, muitas vezes, desejam ter no meio web, é simplesmente um reflexo da limitação da página impressa. Devemos aceitar o fato de que a web não tem as mesmas restrições e devemos projetar (o web design) para essa flexibilidade”.

John Allsopp mostra no artigo que a mídia impressa possui limites de *layout*, mas *web* não possui esse tipo de problema. Nesse artigo ele também comenta:

”Faça páginas que são acessíveis, independentemente de navegador, plataforma ou tela que seu leitor escolha ou tenha que usar para acessar suas páginas. Isso significa páginas que são legíveis independentemente da resolução ou tamanho da tela, ou do número de cores.”

Allsopp não estava se referindo a celulares e *desktops*, visto que, essa artigo foi publicado quando ainda não existia uma variedade grande de dispositivos acessando a *internt* como hoje em dia. Mas mesmo nessa época, ele já chamava a atenção para a flexibilidade das páginas *web*.

Nos meados de 2010, Ethan Marcotte escreveu um artigo no *site* A List Apart onde o termo *Web Design* Responsivo surgiu e que mudaria a filosofia e os métodos de desenvolvimento de *layouts*. A palavra ”responsivo” foi retirada da arquitetura onde é

referente a técnicas cujos materiais utilizados se adaptam ao ambiente que interagem [5, 1]. No artigo Ehtan explica:

"Ao invés de criar designs desconectados para cada um, do crescente número, de dispositivos web, nós podemos tratá-los como faces da mesma experiência. Podemos criar uma experiência de visualização ideal e embutir tecnologias padronizadas nos nossos designs para fazê-los não apenas mais flexíveis, mas mais adaptados para a mídia que os renderiza."

No decorrer do artigo, Ethan explica seus conceitos e sugestões (usando tecnologia que já era existente à época de sua publicação) para que as páginas fossem projetadas usando o que ele chamou de *web design* responsivo [5]. Para se conseguir implementar um *design* responsivo é necessário fazer um *design* flexível e adaptável, que se ajuste às características do navegador, do dispositivo e do contexto do usuário. As três tecnologias principais envolvidas nesse processo são:

1. *layout* fluído;
2. imagens e recursos flexíveis e;
3. *media queries*.

A figura 1.4 mostra o exemplo de como os componentes de um *site* responsivo podem se organizar de maneira a se adaptar a dispositivos diferentes. O conteúdo exibido em cada dispositivo é o mesmo, o que muda é a disposição dos componentes do *site* em cada um dos dispositivos.

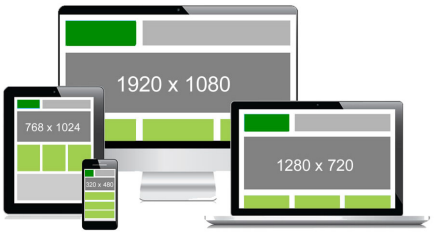

Figura 1.4. Componentes de um website responsivo visualizados em quatro tipo de dispositivos diferentes.

1.3.1. Mobile First e Progressive Enhancement

Mobile First (Móvel Primeiro) é uma metodologia de desenvolvimento *web* em que se deve primeiro planejar o *site* para dispositivos móveis e, somente depois, projetar para dispositivos maiores gradualmente. Os aparelhos móveis possuem limitações em relação ao *desktop*, como: tela menor, rede lenta, *hardware* mais lento, *touch* menos preciso do que *mouse* e etc.. Assim, é mais fácil iniciar um projeto levando em consideração as restrições do *mobile* e depois evoluir para o *desktop* que é menos limitado [1, 3].

O *mobile first* foca no que há de principal no *site* criando uma base sólida de código portável, simples e funcional; incrementando-o posteriormente para atingir recursos

mais avançados de navegadores mais capazes. Esse processo é chamado de *progressive enhancement* - melhoria progressiva da página [1].

Contudo, o desenvolvimento *mobile first* não precisa utilizar apenas restrições. Os dispositivos móveis possuem a capacidade de contextos diferentes. Enquanto, em geral, os *desktops* são usados nas mesmas situações (em casa ou escritório) os dispositivos móveis podem facilmente ser usados em muitos outros contextos, como estar em: restaurantes, lojas, lugares públicos e etc. Essa capacidade pode oferecer ao *mobile* novas possibilidades que o *desktop* não tem como, por exemplo, tirar proveito da localização do usuário [1].

1.3.2. A Fórmula do Desing Responsivo

O projeto responsivo requer a implementação de *layouts* utilizando apenas medidas relativas, ou seja, substituir medidas em *pixels* ou pontos por medidas em porcentagens ou ems. Para ajudar na conversão de medidas absolutas para relativas pode ser utilizada a fórmula da figura 1.5. Na fórmula, a variável Alvo representa o elemento com medida fixa que se deseja converter; Contexto é a medida do elemento-pai onde o elemento-alvo está posicionado e; Resultado representa o valor fluído convertido obtido a partir do elemento-alvo.

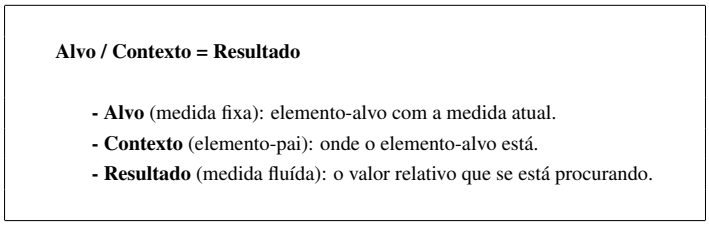

Figura 1.5. Fórmula para conversão de medida absoluta para medida relativa.

Para exemplificar a utilização da fórmula da figura 1.5, considere um título com tamanho de fonte igual 24px que deve ser convertido para ”em”. Esse título possui como elemento ascendente (elmento-pai) o corpo (*body*) do documento HTML. O tamanho padrão da fonte em navegadores *desktop* é 16px, ou seja, o elemento-pai possui o tamanho da fonte igual 16px. E, por fim, aplicando a fórmula a medida fluída se torna igual a **24 / 16 = 1,5ems**.

Como segundo exemplo de aplicação da fórmula da figura 1.5, considere que há um *link* com tamanho de 11px dentro do título citado no exemplo do parágrafo anterior. Nesse caso, o contexto do *link* é o título, visto que, o elemento-alvo (*link*) possui como elemento imediatamente ascendente o título. Aplicando a fórmula o valor relativo do *link* fica igual a **11 / 24 = 0,4583ems**.

1.4. Layouts Fluídos

O desenvolvimento de *sites* com *layouts* fluídos prima pela não especificação de medidas fixas (pontos, centímetros, milímetros e etc.) no *layout* do projeto, tornando possível que o *site* se adapte automaticamente ao que deve ser apresentado na tela. O segredo é não usar *pixels* nas medidas, mas priorizar um *layout* fluído e flexível baseado em porcentagens e ems no CSS do *site* [4]. Seja qual for o tamanho e resolução do dispositivo que se faz

acesso à *internet*, o *layout* fluído evita barras de rolagem e/ou conteúdos ”cortados” (não exibidos completamente) no navegador [3].

É possível utilizar qualquer tipo de medida relativa no RWD, mas há um consenso obtido através de testes empíricos: utilizar porcentagens para formatar tamanhos de *layout* (larguras, margens, espaçamentos e etc.) e usar ems para formatar tamanhos de fontes. O código C_1 apresentado na figura 1.6 mostra um exemplo de código CSS de um projeto fluído onde há definições de tamanho de *layout* dos elementos HTML *body* e *article* utilizando porcentagens. A *tag body* ocupa 100% da página e a *article* ocupa 75% com margem interna igual a 10% da página.

```
body {
    /* a página ocupa a largura de toda a tela */
    width: 100%;
}
article {
    /* o article ocupa 3/4 da página */
    width: 75%;
    /* e possui uma margem interna de 10% do tamanho do pai */
    padding: 10%;
}
```

Figura 1.6. Código C_1: CSS fluído de formatação de tamanho das tags body e article.

A figura 1.7 possui o código CSS fluído C_2 que formata o tamanho da fonte das *tags p*, *h1* e *h2* utilizando a medida ”em”. O cálculo da fonte de cada uma dessas *tags* foi feito levando em consideração que o corpo do documento HTML (com fonte padrão igual 16px) é o elemento-pai de cada uma delas.

```
body {
    /* possui um font-size implícito que equivale a 16px na maioria dos navegadores */
}
p {
    /* parágrafos com tamanho de 18px / 16px = 1.125ems */
    font-size: 1.125em;
}
h1 {
    /* título principal com o dobro da fonte base (32px / 16px = 2ems) */
    font-size: 2em;
}
h2 {
    /* título secundário é 50% maior que o valor padrão (24px / 16px = 1.5ems) */
    font-size: 1.5em;
}
```

Figura 1.7. Código C_2: CSS fluído de formatação de fonte das tags p, h1 e h2.

1.4.1. Flexibilidade entre Elementos

Os elementos de um *site* responsivo possuem medidas relativas e escaláveis mantendo relações de tamanho com os outros no documento. Essa característica garante que os elementos se ajustem de forma harmônica em diferentes tamanhos na tela.

Para ilustrar melhor a flexibilidade entre elementos de um *site* responsivo, considere o código HTML C_3 (figura 1.8) e o seu código CSS C_4 (figura 1.9). A *tag article* possui fonte igual a 20px (16px * 1.25em), visto que, seu elemento-pai é o *body* e o tamanho padrão da fonte em navegadores *desktop* é 16px. As *tags h1* e *p* tem como elemento-pai a *tag article*, pois esta última é imediatamente ascendente as duas primeiras. Assim, *h1* e *p* possuem fonte igual a 40px (20px * 2em) e 18px (20px * 0.9em), respectivamente. Qualquer alteração na medida de *article* irá afetar, indiretamente, as medidas de *h1* e *p*.

```
<html>
<body>
    <article>
        <h1>Web Desing Responsivo</h1>
        <p>Um novo jeito de pensar na web.</p>
    </article>
</body>
</html>
```

Figura 1.8. Código C_3: HTML de uma página com tags article, h1 e p.

```
article {
    font-size: 1.25em;      // 20 / 16 = 1.25
}
h1 {
    font-size: 2em;      // 40 / 20 = 2
}
p {
    font-size: 0.9em;      // 18 / 20 = 0.9
}
```

Figura 1.9. Código C_4: CSS responsivo do código HTML C_3 utilizando a medida ”em”.

1.4.2. Menos Flexibilidade entre Elementos

A grande flexibilidade da medida ”em” pode ser um problema em projetos com muitos elementos. Se um documento HTML possuir muitos elementos em muitos níveis diferentes, torna-se difícil controlar as medidas utilizando ”em”. Uma mudança na medida de um elemento afetará todos os seus elementos-filhos, inclusive os filhos em todos os outros níveis descendentes.

Existe outra medida fluída no CSS, a medida rem, que não afeta a medida na hierarquia de elementos. A medida rem é semelhante a ”em”, mas com a diferença do

cálculo das medidas dos elementos ser relativo apenas ao elemento raiz. Modificando o código CSS C_4 da figura 1.9 para utilizar a medida rem ao invés de ”em”, o código fica igual ao da figura 1.10. Considerando que a fonte padrão nos navegadores *desktop* é 16px (ou seja, a fonte do elemento raiz é 16px), ao aplicar a fórmula de conversão de medida fixa para relativa (figura 1.5), as novas medidas das *tags* são: *article* possui 16px * 1.25rem = 20px; *h1* possui 16px * 2rem = 32px e; *p* possui 16px * 0.9rem = 14px.

```
article {
    font-size: 1.25rem;       // 20 / 16 = 1.25
}
h1 {
    font-size: 2rem;       // 32 / 16 = 2
}
p {
    font-size: 0.9rem;       // 14 / 16 = 0.9
}
```

Figura 1.10. Código C_5: CSS do código HTML C_3 utilizando a medida rem.

1.5. Imagens e Recursos Flexíveis

Layouts realmente fluídos precisam de imagens que se adaptem a todo tipo de resolução. As imagens não são naturalmente flexíveis, visto que, elas são baseadas em *pixels*. Através de técnicas variadas, é possível fazer com que os *assets* (recursos como imagens, vídeos e etc.) do *site* sejam flexíveis [3].

Para implementar imagens flexíveis, porcentagens podem ser utilizas para dimensioná-las de uma maneira bem simples. No código CSS C_6 da figura 1.11 a propriedade *max-width* restringe, de forma genérica, a largura de imagens em um intervalo de 100% do elemento onde a imagem está contida. Ou seja, as imagens podem ter qualquer largura até no máximo 100% do local disponível que estão contidas.

```
img {
    max-width: 100%;
}
```

Figura 1.11. Código C_6: CSS para imagens flexíveis.

A mesma regra utilizada no código da figura 1.11 pode ser estendida a outros tipos de mídia e recursos, como: *iframe*, *object*, *embed* e *video*. O CSS C_7 na figura 1.12 apresenta uma mesma regra para tornar vários tipos de mídia flexíveis.

1.5.1. Problemas de Imagens Flexíveis

A técnica apresentada na seção 1.5 é bastante útil para apesentar imagens flexíveis. Entretanto, é preciso cuidado, aumentar uma imagem além do seu tamanho original deixará

```
img,
iframe,
object,
embed,
video {
    max-width: 100%;
}
```

Figura 1.12. Código C_7: CSS para tornar vários tipos mídia flexíveis.

o *layout* distorcido. Além disso, carregar uma imagem de alta resolução para depois diminuí-la via código pode deixar a visualização da página muito custosa, pois é necessário carregar um arquivo com peso (tamanho em KB) maior do que seu tamanho (número de *pixels*) apresentado no dispositivo.

Na técnica utiliza no código da figura 1.11 uma mesma imagem é exibida em qualquer tamanho da tela, sendo renderizada de forma a se adaptar a resolução do dispositivo utilizado. Contudo, isso pode levar a problemas no desempenho do *site* acessado. Por exemplo, um usuário *mobile* terá que esperar uma imagem grande (possivelmente feita para usuário *desktop*) ser carregada para só depois ser redimensionada em sua pequena tela e/ou conexão lenta.

O ideal ao apresentar imagens flexíveis seria servir a imagem de tamanho mais adequado para cada tamanho de tela ou velocidade de conexão do usuário. É necessária a existência de várias imagens, em resoluções diferentes, para cada contexto. Assim, para cada tela, pode ser carregada a imagem mais apropriada. Existem várias soluções para lidar com imagens responsivas com alta performance, seguem algumas delas:

- Riloadr [12];
- jQuery Picture [13]: ;
- Picturefill [14];
- Adaptive Images [15];
- Foresight [17] e;
- HISRC [16].

Em algumas dessas técnicas é necessário ter várias versões de tamanhos diferentes da imagem e em outras a imagem é redimensionada antes de ser entregue. Cada uma delas possui suas vantagens e desvantagens e para escolher a melhor a ser utilizada é necessário fazer um estudo e ponderar o custo/benefício de cada uma delas.

1.5.2. Nova Especificação HTML para Imagens: Picture

Uma outra maneira de fornecer imagens para *layouts* responsivos que estará disponível em um futuro muito próximo nos navegadores é o elemento *Picture*. O *picture* foi criado por um grupo de desenvolvedores chamado *Responsive Images Community Group* (RICG) [18] utilizando o princípio de que as fotos são o principal motivo da lentidão no carregamento de uma página.

Através de *picture* é possível declarar diversas fontes para uma única imagem no HTML e controlar qual deve ser apresentada utilizando *media queries*. As *media queries* serão descritas na seção 1.6. Com essa nova *tag* é possível especificar a apresentação de imagens diferentes de acordo com a largura e/ou altura da janela do navegador, orientação do dispositivo, densidade de *pixels*, *layout* para a impressão e etc..

A figura 1.13 possui um trecho de código HTML com um exemplo de utilização da *tag picture*. Esse código especifica a escolha de um entre três arquivos de uma imagem de acordo com a largura do dispositivo usado. Para *viewports* que possuírem pelo menos 45em de largura, será carregada a imagem ”grande.jpg”. Se o *viewport* possuir menos de 45em e, no mínimo, 18em de largura; será exibida a imagem ”media.jpg”. Para *layouts* com menos de 18em de largura, será exibida a imagem ”pequeno.jpg”.

```
<picture>
    <source media="(min-width: 45em)"srcset="grande.jpg">
    <source media="(min-width: 18em)"srcset="media.jpg">
    <img src="pequena.jpg"alt="descrição da imagem">
</picture>
```

Figura 1.13. Código C_8: trecho de código HTML utilizando a tag picture.

1.6. Media Queries

Com *media queries* é possível criar regras no CSS que só se apliquem a determinadas situações, como em resoluções de tela específicas, orientação do dispositivo (paisagem ou retrato) ou sua densidade de *pixels*. O uso de *media queries* torna possível ocultar, fazer aparecer e reposicionar elementos e interações do *site* conforme a resolução do dispositivo que está sendo usada no momento da visualização [1].

As necessidades e desejos de visualização de um *site* podem se alterar dependendo das resoluções dos aparelhos utilizados. Por exemplo, um usuário que acessa um *site* através de um celular cuja tela não tenha uma boa resolução não precisa de um *sidebar* (coluna lateral que pode conter menus, *banners*, *links* e etc.). Para uma tela pequena, as opções de um *sidebar* podem aparecer logo após o fim de um conteúdo principal que é de maior interesse para o usuário [1].

A figura 1.14 ilustra o efeito de *media queries* na organização dos componentes de um *site* em dois dispositivos diferentes. Alguns componentes da figura 1.14 (B) estão dispostos em uma única coluno devido ao pequeno *viewport* em que são apresentados. No *layout* (A) da figura 1.14, os mesmos componentes estão em quatro colunas.

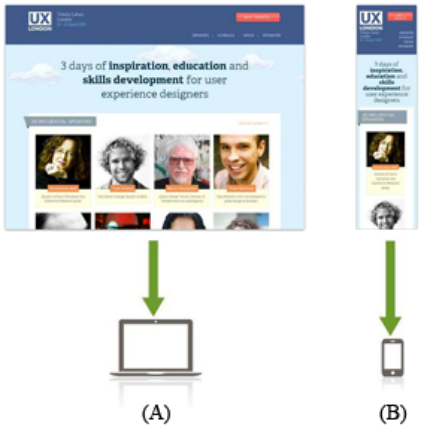

Figura 1.14. Organização dos componentes de um site com media queries.

O código C_9 da figura 1.15 representa a implementação de uma parte do CSS do *site* da figura 1.14. As regras contidas no bloco do comando *@media* serão aplicadas apenas se sua condição for válida. A condição (*max-width: 400px*) indica que as regras serão aplicada apenas se a janela do navegador possuir, no máximo, 400 *pixels* de largura. Se a janela do navegador possuir até 400 pixels de largura (telas de celulares, geralmente), todos os quadros de destaque ocuparão apenas uma coluna (100% da tela). Nas demais larguras da janela, os quadros estarão distribuídos em quatro colunas (25% da tela).

```
.destaque {
    width: 25%;
}
@media max-width: 400px
    .destaque {
        width: 100%;
    }
}
```

Figura 1.15. Código C_9: CSS com media querie.

O uso de medidas fluídas são a principal técnica para que o *design* se adapte as diversas resoluções existentes. As *media queries* devem ser utilizadas apenas quando é necessária uma maior restruturação do *design* para melhorar a experiência do usuário. Podem ser feitas quantas *media queries* forem necessárias visando os diferentes tamanhos de tela, mas a ordem das *media queries* é importante, visto que, a regra que vem por último sobrescreve a anterior.

A *media querie min-width* também pode ser utilizada de forma semelhante a *max-width*. Essas duas *media queries* são as mais comumente utilizadas. Contudo, existem várias outras *media queries* para situações específicas. A seguir estão algumas *media queries* que podem ser potencialmente usadas:

- width: largura da janela do navegador (possui versões *max-width/min-width*);
- height: altura da janela do navegador (possui versões *max-height/min-height*);
- device-width: largura do dispositivo (possui versões *max-device-width/min-device-width*);
- device-height: altura do dispositivo (possui versões *max-device-height/min-device-height*);
- orientation: orientação do aparelho (*portrait* ou *landscape*);
- aspect-ratio: proporção da área de exibição do navegador (por exemplo, *widescreen* de proporção 16:9);
- device-aspect-ratio: proporção do display do dispositivo;
- color: número de *bits* por componente de cor do dispositivo;
- color-index: número de entradas na tabela de cores do dispositivo;
- grid: indica se é um dispositivo de grade ou de mapa de *bits* (*bitmap*);
- monochrome: número de *bits* por *pixel* em dispositivos monocromáticos (escala de cinza);
- resolution: resolução (densidade de *pixels*) do dispositivo;
- scan: processo de escaneamento que um dispositivo do tipo ”TV” pode fazer.

1.6.1. Media Types

As *media queries* podem possuir *media types* que permitem apresentar a página com o estilo mais apropriado ao dispositivo que está sendo usado. Seguem algumas *media types* e os dispositivos que são aplicadas:

- all: todos os dispositivos;
- braille: dispositivo tátil;
- embossed: impressora em braille;
- handhel: dispositivo móvel;
- print: impressora;
- projection: projetor;
- screen: computador;
- speech: sintetizador de voz;
- tt: dipositivo para exibição de caracteres (como *teletypes* e terminais) e;

- tv: TV.

O código C_{10} (figura 1.16) possui duas *média types* que restringem a utilização do arquivo ”print_handheld_style.css” a *layouts* visualizados no modo de impressão ou em dispositivos móveis. No código C_{11} (figura 1.17) as *média types* foram inseridas dentro do CSS. Nesse exemplo, o CSS formata diferentes tamanhos de fonte para dois dispositivos diferentes. A *tag body* possuirá fonte igual 10 *pixels* se a página for exibida em modo de impressão ou 13 *pixels* em um computador.

```
<link rel=”stylesheet” type=”text/css” media=”print, handheld” href=”print_handheld_style.css”>
```

Figura 1.16. Código C_{10}: media types para impressão e para dispositivo móvel.

```
@media print {
    body {
        font-size: 10pt;
    }
}
@media screen {
    body {
        font-size: 13pt;
    }
}
```

Figura 1.17. Código C_{11}: media types para modo de impressão ou para tela de computador.

Nem sempre é fácil classificar um determinado dispositivo de acordo com as *media types* disponíveis. Alguns dispositivos possuem características de mais de um dispositivo, como o iPhone, por exemplo, que possui características de *mobile* e *desktop*. E outros aparelhos nem possuem *media types*. Assim, além de *media types*, na maioria do casos, é necessário delimitar os dispositivos utilizando o tamanho da tela. Na figura 1.18, o código C_{13} apresenta *medias queries* que verificam se o dispositivo é um computador e possui tela mínima de 767 *pixels*.

```
@media screen and @media (min-width: 767px) {
  }
```

Figura 1.18. Código C_{13}: media querie com operadores screen e min-width.

1.6.2. Operadores Lógicos

Uma *media querie* podem utilizar expressões com operadores lógicos. Seguem os operadores lógicos de *media queries* e suas descrições:

- and: representa o operador lógico *and*;
- , (vírgula): representa o operador lógico *or*;
- not: representa o operador lógico *not*;
- only: a palavra *only* não é exatamente um operador lógico, ela serve para esconder *media queries* de agentes mais antigos que não as reconhecem.

Na figura 1.19 há um código exemplo com uma *media querie* que utiliza alguns operadores lógicos. Nesse código, as regras do bloco da *media querie* serão utilizadas apenas em computadores que não estejam no modo de impressão ou em dispositivos móveis. A palavra *only* impede que as regras sejam aplicadas em navegadores que não reconhecem a *media querie*.

```
@media only screen and not print, handhel {
  }
```

Figura 1.19. Código C_{14}: operadores lógico em media querie.

1.6.3. Breakpoints

Breakpoints (pontos de interrupção) são ”pontos” em que se deve utilizar estilos diferenciados para que a experiência do usuário seja a mais eficiente possível ao navegar no *site* de seu dispositivo. Os *breakpoints* são, basicamente, as *media queries* que delimitam determinadas regras CSS a determinados tipos de dispositivos. A divisão do CSS em *breakpoints* depende das características de cada projeto. Não é fácil fazer *breakpoints* que atinjam todos os dispositivos existes, assim, o projeto deve buscar expecificar *breakpoints* destinados aos dispositivos de seu escopo. Existem alguns modelos de *breakpoints* propostos que ajudam a iniciar um projeto. Alguns desses modelos são listados a seguir:

- 320 and Up [19];
- Less Framework [8];
- Skeleton [10] e;
- Bootstrap [7].

Os *breakpoints* do *framework* Bootstrap estão escritos no código C_{15} da figura 1.20 para ilustrar melhor o uso de *breakpoints*. O Bootstrap foi projetado levando em

consideração o tamanho da tela e orientação (retrato ou paisagem) de vários dispositivos. Nesse *framework* não há a *media querie orientation* responsável por verificar a orientação do dispositivo, mas isso é feito através do tamanho da largura da tela no momento da orientação do dispositivo.

```
/* Telefones em landscape e abaixo */
@media (max-width: 480px) {
}
/* Telefone em landscape a tablet em portrait */
@media (max-width: 767px) {
}
/* tablet em portrait a landscape e desktop */
@media (min-width: 768px) and (max-width: 979px) {
}
/* Desktop grande */
@media (min-width: 1200px) {
}
```

Figura 1.20. Código C_{15}: Breakpoints do framework Bootstrap.

1.7. Frameworks para Web Design Responsivo

Construir um *site* com características responsivas pode ser um processo mais longo do que o *design* tradicional. Desenvolver *sites* com flexibilidade e adaptação é mais difícil e complicado do que fazê-los fixos com *pixels*, pois as ferramentas de desenho ainda não estão preparadas para *layouts* responsivos e os *designers* gráficos costumam ter dificuldade em pensar responsivamente. Para facilitar o desenvolvimento de *sites* responsivos, já existem *frameworks* para RWD. Esses *frameworks* são uma espécie de esqueleto que pode ser moldado de acordo com as necessidades do *site*. Alguns dos *frameworks* para *web design* responsivo mais populares são:

- Bootstrap [7];
- Wirefy [11];
- Skeleton [10];
- Less Framework [8] e;
- Foundation [9].

Para escolher o *framework* mais adequado ao desenvolvimento de um projeto é necessário conhecer melhor as vantagens e desvantagens de cada um deles. Na seção 1.7.1 são detalhadas algumas características do *framework* Bootstrap.

1.7.1. Framework Bootstrap

O Bootstrap é um *framework front-end* de código *opensource* (aberto) para criação de *sites* responsivos que foi desenvolvido pela equipe do Twitter [20]. Ele possui dezenas de componentes prontos para o desenvolvimento de *sites*, como *plugins* para *jQuery*. Esses componentes facilitam a construção do *site*, mas, por outro, podem limitar o desenvolvimento do *site* a utilização dos componentes.

Esse *framework* utiliza um sistema de *grids* (figura 1.21 (A)), onde o *site* pode ser divido em até 12 colunas de mesma largura. Os *grids* podem ser mesclados para formar, por exemplo: 1 coluna de 12 *grids*, 2 colunas de 6 *grids*, 3 colunas de 4 *grids* e etc. (figura 1.21 (B)). Os *grids* são responsivos e permitem definir conjuntos de *grids* para resoluções de telas diferentes (figura 1.21 (C)). Esses comportamentos são definidos através das classes xs (dispositivos muito pequenos), sm (dispositivos pequenos), md (dispositivos médios) e lg (dispositivos grandes).

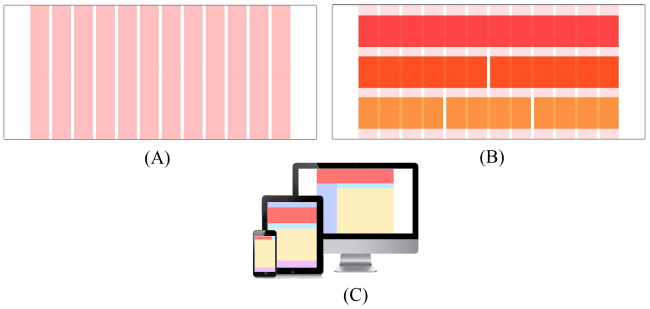

Figura 1.21. Grids no framework Bootstrap.

1.8. Considerações Finais

A *web* pode ser acessada por vários tipos de aparelhos diferentes (*laptops*, *tablets*, *netbooks*, celulares, *desktops* e etc.). Essa característica tornou a *web* bastante popular e acessível. Mas por outro lado, servir *sites* a todos esses dispositivos causou um problema aos desenvolvedores, visto que, cada aparelho possui características peculiares que influenciam no *design*, como: tamanho da tela e contexto de uso.

Juntamente com essa nova *web* mais acessível, surgiu o conceito *One Web* (*Web* Única): usar uma só URL do *site* para servir todos os dispositivos. O *web design* responsivo é a resposta para uma *web* única, adaptando *sites* aos mais variados tipos de dispositivos e contextos de uso através de três técnicas simples: *layout* fluído, imagens flexíveis e *media queries*.

Referências

[1] Lopes, S., A Web Mobile: programe para um mundo de muitos dispositivos, Casa do Código, 2013.

[2] Mazza, L., A Web Mobile: programe para um mundo de muitos dispositivos, Casa Mazzado Código, 2013.

[3] Zemel, T., Web Design Responsivo: páginas adaptáveis para todos os dispositivos, Casa Mazzado Código, 2013.

[4] Silva, M. S., Web Design Responsivo, Novatec, 2014.

[5] Marcotte, E. (2010) "Responsive Web Design - A List Apart Article", http://alistapart.com/article/responsive-web-design, Maio.

[6] Allsopp, J. (2000) "A Dao of Web Design - A List Apart Article", http://alistapart.com/article/dao, Abril.

[7] Bootstrap "Bootstrap: The world's most popular mobile-first and responsive front-end framework", http://alistapart.com/article/responsive-web-design, acessado em março de 2015.

[8] Less Framework "Less Framework 4", http://lessframework.com, acessado em março de 2015.

[9] Foundation "Foundation: The Most Advanced Responsive Front-end Framework from ZURB", http://foundation.zurb.com, acessado em março de 2015.

[10] Skeleton "Skeleton: Responsive CSS Boilerplate", http://getskeleton.com, acessado em março de 2015.

[11] Wirefy "Wirefy", http://getskeleton.com, acessado em março de 2015.

[12] Riloadr "tubalmartin/riloadr - GitHub", https://github.com/tubalmartin/riloadr, acessado em março de 2015.

[13] jQuery Picture "Abban/jQuery-Picture - GitHub", https://github.com/Abban/jQuery-Picture, acessado em março de 2015.

[14] Picturefill "scottjehl/picturefill - GitHub", https://github.com/scottjehl/picturefill, acessado em março de 2015.

[15] Adaptive Images "Adaptive Images in HTML", http://adaptive-images.com, acessado em março de 2015.

[16] HISRC "teleject/hisrc", https://github.com/teleject/hisrc - GitHu, acessado em março de 2015.

[17] Foresight "adamdbradley/foresight.js - GitHub", https://github.com/adamdbradley/foresight.js, acessado em março de 2015.

[18] RICG "ResponsiveImages.org", http://responsiveimages.org, acessado em março de 2015.

[19] Stuff Nonsense "320 and Up Stuff & Nonsense", http://stuffandnonsense.co.uk/projects/320andup, acessado em março de 2015.

[20] Twitter "Bem vindo ao Twitter", https://twitter.com, acessado em março de 2015.