Web Design Responsivo

Páginas adaptáveis para todos os dispositivos

Casa do Código

TÁRCIO ZEMEL

Edição

Adriano Almeida
Vivian Matsui

Revisão

Bianca Hubert
Vivian Matsui

[2017]

Casa do Código

Livros para o programador

Rua Vergueiro, 3185 - 8º andar

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

www.casadocodigo.com.br

SOBRE O GRUPO CAELUM

Este livro possui a curadoria da Casa do Código e foi estruturado e criado com todo o carinho para que você possa aprender algo novo e acrescentar conhecimentos ao seu portfólio e à sua carreira.

A Casa do Código faz parte do Grupo Caelum, um grupo focado na educação e ensino de tecnologia, design e negócios.

Se você gosta de aprender, convidamos você a conhecer a Alura (www.alura.com.br), que é o braço de cursos online do Grupo. Acesse o site deles e veja as centenas de cursos disponíveis para você fazer da sua casa também, no seu computador. Muitos instrutores da Alura são também autores aqui da Casa do Código.

O mesmo vale para os cursos da Caelum (www.caelum.com.br), que é o lado de cursos presenciais, onde você pode aprender junto dos instrutores em tempo real e usando toda a infraestrutura fornecida pela empresa. Veja também as opções disponíveis lá.

ISBN

Impresso e PDF: 978-85-66250-07-7

EPUB: 978-85-5519-004-9

Você pode discutir sobre este livro no Fórum da Casa do Código: http://forum.casadocodigo.com.br/.

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

PREFÁCIO

A era pós-PC da computação pessoal chegou com tudo. Nos tablets e smartphones, a web, finalmente, tem a chance de se tornar verdadeiramente ubíqua e onipresente. Ainda estamos no começo dessa nova fase da tecnologia, mas o futuro é inegavelmente da web móvel e dos mais diversos aparelhos de navegação.

Essa nova perspectiva traz desafios importantes. Até ontem, Web significava um navegador instalado em um Desktop controlado por mouse e teclado em uma tela de tamanho confortável. Nossos sites foram construídos pensando nesse cenário.

Mas, agora, temos telas pequenas, touch screen, redes móveis e muitas outras diferenças. Precisamos de uma nova web. Uma web que suporte essa explosão de dispositivos e, mais ainda, esteja preparada para o futuro de dispositivos que ainda nem conseguimos antever.

O *Web Design Responsivo* é a chave para essa nova Web. É pensar em páginas que se adaptem a todo tipo de dispositivo e contexto de uso. É sair das limitações de um browser Desktop e seu tamanho previsível, e pensar em páginas com flexibilidade que suportem todo tamanho de tela, qualquer tipo de resolução, interfaces com touch ou mouse. Pensar responsivamente é repensar a web para o futuro.

Este livro é a ferramenta que você precisa para entender os desafios do Web Design Responsivo. O Tárcio mostra os pilares de uma interface responsiva, explora os aspectos técnicos do HTML e

CSS, e mostra ideias para uma interface móvel usável.

É um livro que recomendo para todo desenvolvedor Web que queira participar ativamente da nova Web. Espero que você aproveite a leitura como eu aproveitei.

E, quando terminar a leitura, não deixe de conferir também meu livro **A Web Mobile** (http://sergiolopes.org/livro-web-mobile/). Nele aprofundo em vários outros temas relacionados a dispositivos móveis e sites responsivos.

Sérgio Lopes - *@sergio_caelum*

Instrutor e desenvolvedor na Caelum

Sumário

Versão: 20.8.24

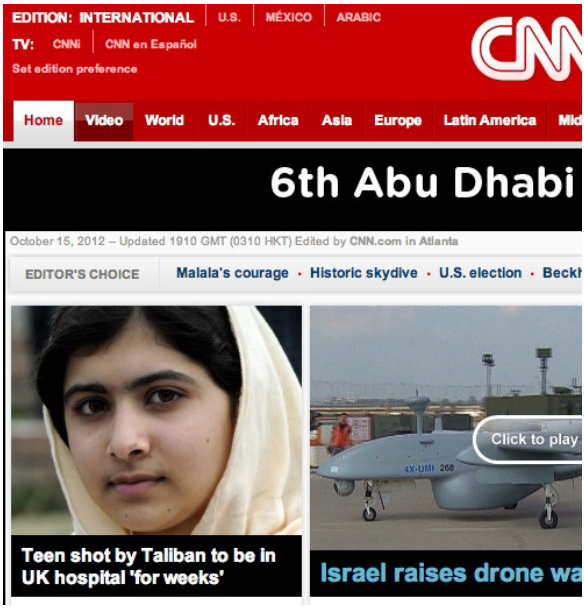
Capítulo 1

INTRODUÇÃO

É possível desenvolver uma só apresentação para um site, um só *web design*. Mais além: que esse design seja bem apresentado em quaisquer dispositivos e, conforme planejado, que se adapte aos diferentes meios em que este site é acessado. Sim, é possível, mas não é o que vemos por aí.

Repare no site da CNN em um *browser*, em 2012:

E agora o mesmo site, quando visto em um dispositivo que não um *desktop*:

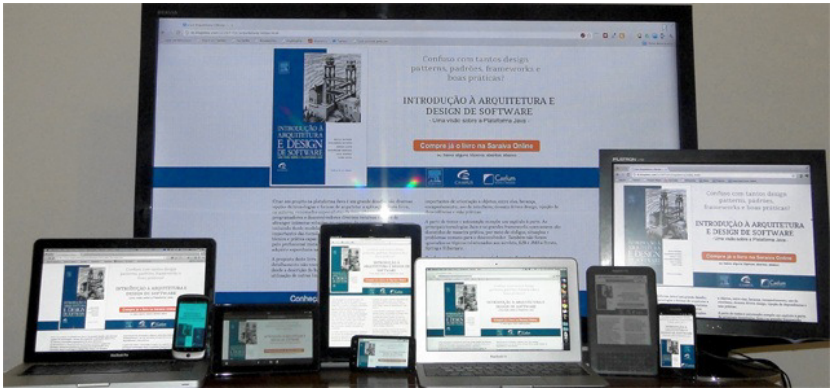


A página no dispositivo móvel aparece cortada e, para poder se ver a parte que está faltando, é necessário fazer o *scroll* horizontal. Já na versão aberta em um desktop, ela abre completamente. Assim, vemos que a página da CNN não se adaptou aos diferentes meios pelos quais ela pode ser acessada.

Repare no mesmo efeito no site da **Editora Casa do Código**:

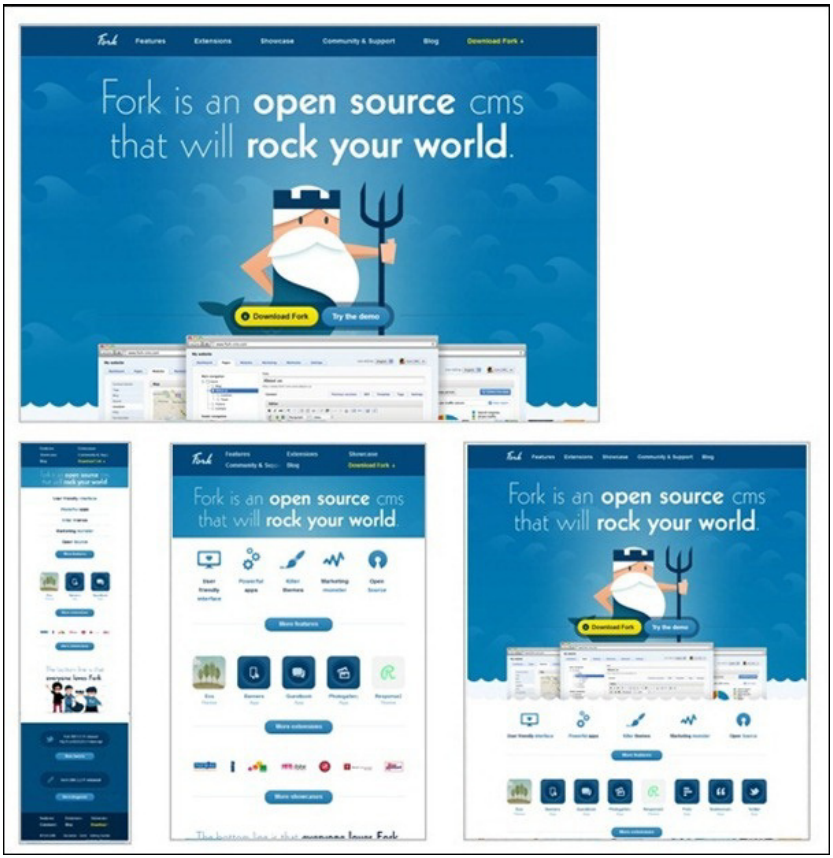
A mesma página pode ser visualizada da seguinte forma em dispositivo móvel:

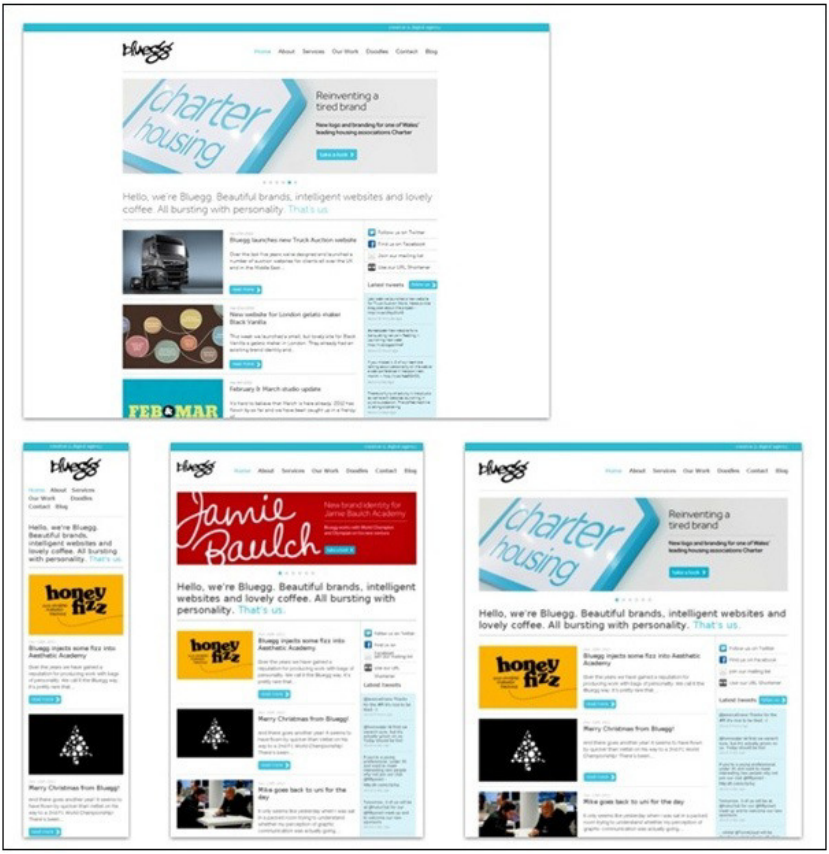
Para uma imagem mais impactante, veja o site do livro de arquitetura Java do pessoal da **Caelum** em diversos dispositivos:

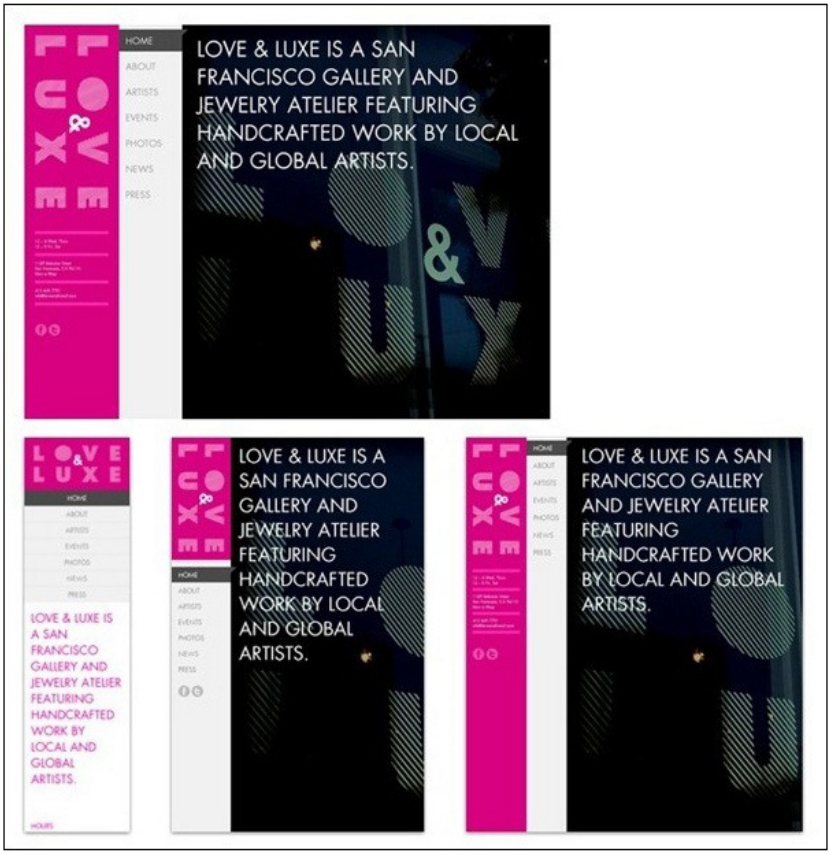


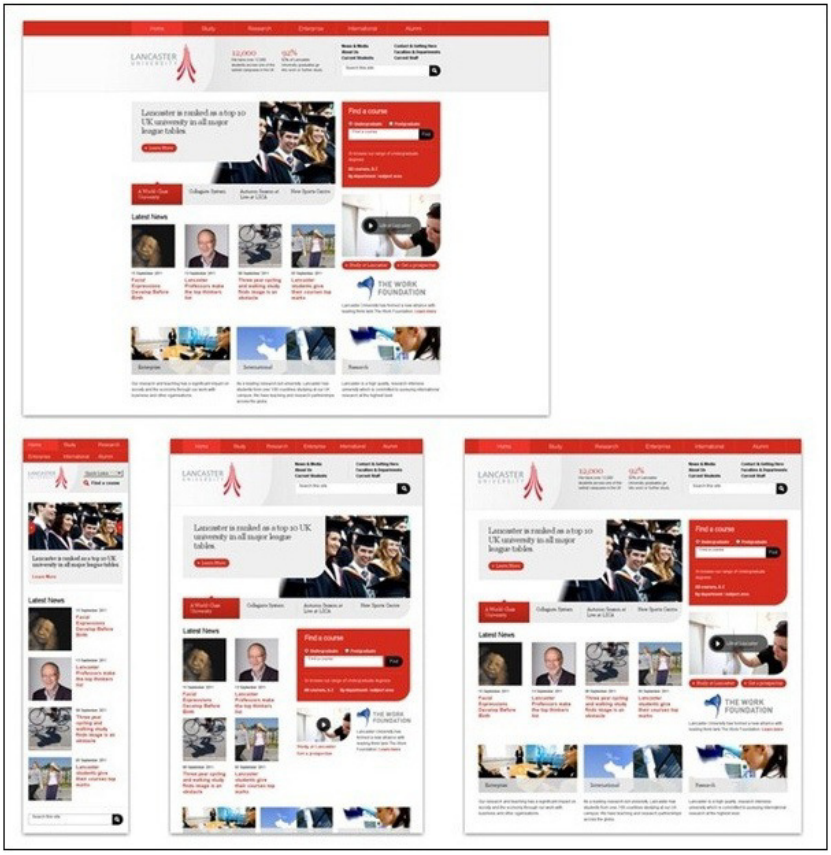
Veja outros exemplos incríveis de sites com essas

características que a revista **TripWire** reuniu:









Sites com estas características possuem **web design responsivo**.

Um site com web design responsivo — ou *responsive web design* — pode ser acessado de um PC, notebook, smartphone, tablet, TV, geladeira, banheira — sim, realmente existem geladeiras e banheiras que acessam a internet! -; em suma, de qualquer dispositivo com acesso à rede, independentemente de sua resolução, de sua capacidade de cores ou se é *touch* ou não. E, mesmo com essas diferenças dos dispositivos que podem acessar

seu site, ele continua bem apresentado — e sem quaisquer prejuízos de SEO (http://ow.ly/yynAV) —, inclusive com possibilidade de se alterar a ordem em que os conteúdos aparecem e, até mesmo, se determinados conteúdos serão ou não mostrados para "tal" ou "qual" dispositivo!

Diga "Não!" a mais versões específicas para celulares; basta de linguagens próprias para mobile; chega de subdomínios ou diretórios específicos para atender ao *público móvel*! Não existe uma "*web mobile*". **A web é única**, e web design responsivo é a única resposta para uma web única!

Não se trata de uma "moda" ou um *hype* de internet; nem de algo que chegou, que vai angariar alguns fãs e sumir na próxima estação. O web design responsivo é uma nova forma de **pensar a web** e, dentro de pouco tempo, será tão vital e importante aos desenvolvedores e à experiência do usuário quanto o próprio HTML ou CSS.

Ao terminar de ler este livro e fixar seus conteúdos, você fará parte do rol de profissionais de web capazes de desenvolver **sites responsivos** e poderá gritar, a plenos pulmões, que você também contribui para uma web e para um mundo melhor!

1.1 ESTATÍSTICAS DO MUNDO MOBILE

Não é somente porque usar essa tecnologia é algo divertido de se fazer. Existe uma forte base estatística que justifica por que se deve prestar muita atenção (e atuar) no mercado de sites multidispositivos.

Por exemplo, havia uma previsão de que as vendas de celulares

ultrapassariam as vendas combinadas de desktops e notebooks em 2012. Bem, a previsão estava errada: isso aconteceu em 2010 (http://ow.ly/GRyHP), mesmo ano em que o número de celulares e smartphones bateu a casa dos **200 milhões** no Brasil!

Segundo divulgado pela Mashable em um artigo de 2013 (http://ow.ly/yykYc), 17,4% do acesso mundial à web foi via dispositivos mobile (considerando somente dipositivos "de bolso", não tablets), representando um aumento considerável em relação ao ano anterior. Especialmente na América do Sul, a porcentagem mais que dobrou!

É interessante notar que, conforme é destacado em um artigo sobre o mercado mobile do Brasil em 2011 (http://ow.ly/bg7Wp), a venda de smartphones cresceu **179%** no Brasil, com destaque para as informações de que:

- 33% das pessoas já tinham smartphones;
- 20% com funções consideradas avançadas, como Wifi, GPS etc.;
- 41% acessava a internet;
- No último quadrimestre, o número de usuários mobile cresceu 40%.

Pode parecer incrível, mas, comparando as vendas de aparelhos móveis com a taxa de natalidade, os dispositivos mobile têm uma taxa de crescimento **4 vezes maior do que a da população mundial**!

E, partindo para o lado financeiro (http://ow.ly/awinf):

- O PayPal movimenta US$ 10 milhões por dia em pagamentos mobile;
- As vendas mundiais através de dispositivos móveis no eBay chegaram perto de US$ 2 bilhões em 2010 (são feitos cerca de 94 lances *por minuto*);
- Em dezembro de 2010, o número de usuários do aplicativo do Yelp estava na casa dos 3,2 milhões, e 35% das buscas feitas no site são através dele.

Na página da Smart Insights, *Mobile Marketing Statistics 2014* (http://ow.ly/yylp6), dentre as muitas informações valiosas, consta que visitas em nível mundial feitas por dispositivos móveis a *e-commerces*, comparando o mesmo período de 2012 e 2013, tiveram um aumento de quase 5% para smartphones e mais de 5% para tablets.

Levando em conta como é feito o uso do mobile pelas pessoas,

também há alguns números recolhidos pelo Google em solo estadunidense que merecem atenção (http://ow.ly/awkw7):

- 79% usam smartphone como auxílio na hora de fazer compras (70% dentro da loja);
- 54% para procurar o endereço de uma loja;
- 49% para comparar preços;
- 44% para ler *reviews* de produtos;
- 74% tomam a decisão da compra baseados em informações obtidas no smartphone;
- 35% dos que pesquisam no smartphone compram o produto através dele;
- 88% que encontram informações no smartphone tomam a iniciativa no mesmo dia;
- 71% dos que buscam algo em um smartphone o fazem após verem um anúncio;
- 79% dos anunciantes ainda não têm um site otimizado para mobile!

Segundo a CNN (http://ow.ly/yym7e), 2014 foi o ano em que as pessoas por lá usaram mais aplicativos em smartphones e tablets do que em PCs.

Em solo nacional, em 2013 foi o ano em que as vendas de tablets ultrapassaram as de desktops e notebooks pela primeira vez (http://ow.ly/yylXX)!

"Nosso Planeta Mobile: Brasil"

Em maio de 2012, a **Google** liberou no mundo todo uma pesquisa chamada *Nosso Planeta Mobile*. Felizmente, o Brasil também foi contemplado e um documento foi disponibilizado com os resultados obtidos (sendo este atualizado a cada ano). É altamente aconselhável que você tenha acesso e estude o relatório **Nosso Planeta Mobile: Brasil — Como entender o usuário de celular** (http://ow.ly/yymst). Já adiantando alguns números importantes do ano de 2013:

- A difusão dos smartphones atinge 26% da população e esses proprietários de smartphones dependem cada vez mais de seus dispositivos. 46% acessam a internet todos os dias e muitos nunca saem de casa sem ele;
- 90% (!) das pessoas usam o telefone durante outras atividades, como assistir TV;
- 89% dos usuários de smartphones procuram informações locais em seus telefones e 90% (!) tomam decisões em decorrência disso, como fazer uma compra ou entrar em contato com uma empresa;
- Anúncios para celular são vistos por 96% (!) das pessoas que usam smartphones, e 30% delas já fizeram

uma compra por este dispositivo;
- 76% dos usuários realizaram uma pesquisa em seus smartphones depois de visualizar um anúncio offline.

Por ainda ser algo recente, o web design responsivo ainda não foi amplamente adotado e, por enquanto, ainda não é um padrão *de fato*, razão pela qual pessoas que usam smartphones estão frustradas com a experiência web móvel (http://ow.ly/yymQI). Aí está a oportunidade de aprender mais sobre o assunto e destacar-se, provendo soluções web muito melhores que agradarão as pessoas que acessarem seu projeto web por meio de quaisquer dispositivos!

1.2 UMA QUESTÃO DE CONCEITO

Como visto, já é uma realidade que os mais diversos dispositivos mobile chegaram para ficar, estão assumindo seu lugar (e o de outros) e, daqui a não muito tempo, prometem ser o padrão para acesso à web! Já é hora de começar a olhar para o desktop com um olhar saudosista, agradecendo por tudo o que ele fez por todos nós até hoje, mas, ao mesmo tempo, já ir preparando-se para o que o futuro (breve) trará.

Em nosso dia a dia, já é possível perceber isso. Note que, nos lugares mais inusitados, as pessoas estão andando, não somente com seus smartphones ligados e em pleno uso, mas, também, com seus tablets, e-Readers e muitos outros dispositivos com nomes extravagantes.

Porém, não pense somente em dispositivos com displays menores do que os tamanhos de monitores convencionais. Ou

ainda não ouviu falar nas Smart TVs e/ou viu alguém acessando um site através de um videogame?

Em quaisquer dos casos citados, a situação ideal é que os sites desenvolvidos sejam bem apresentados (e até *otimizados*) conforme o *device* que o está acessando no momento. Será que as informações exibidas para quem conecta-se a um site de jogos por meio de seu videogame devem ser apresentadas exatamente da mesma maneira de quem realiza esse acesso usando um celular ou tablet? Obviamente, não!

Mas é aí que está o grande problema do desenvolvimento web "tradicional": os sites não são idealizados, desde sua concepção, para serem flexíveis e mostrados de forma adequada, seja lá por qual meio de acesso as pessoas os estejam acessando. Perceba que, mais do que uma questão técnica, estamos tratando de algo *conceitual*!

Para começar a pensar em web design responsivo, o primeiro passo é começar a mudar seus conceitos!

Qual a solução?

A solução (ou soluções) para que a web possa evoluir e os desenvolvedores possam atualizar seu *know-how* teórico e prático é, justamente, o que é tratado neste livro. Por meio dos conteúdos aqui apresentados, os profissionais de web poderão tomar conhecimento sobre técnicas como **layout fluido**, **imagens flexíveis**, **media queries** e muitas outras dicas úteis sobre o desenvolvimento de **web sites responsivos**!

Preparado?

1.3 MAS MEU SITE JÁ ESTÁ BOM PARA ANDROID E IPHONE!

Você pode ter um site focado para cada dispositivo, mas há diversas desvantagens nessa abordagem. E quando houver um novo dispositivo móvel com uma tela bem diferente? Quando precisar fazer uma alteração, terá de tomar cuidado com cada uma das várias versões do seu site.

Tomemos como exemplo a Wikipédia que, atualmente, redireciona para a página http://en.m.wikipedia.org/ se o acesso for mobile. Isso é chamado de estratégia do "m ponto" (`m.website.com`). Se você abri-la em desktop, a página flui e ocupa a tela toda. Fica "feio" e meio ruim de usar, mas é aceitável. Eles não usam *media queries* (que ainda serão vistas neste livro) e nem nada mais avançado para *responsivar*. Mas é responsivo? Ninguém sabe dizer ao certo, fica em uma área cinza.

Um outro conceito um pouco diferente é o de **One Web**: deve-se usar uma só URL para tudo, um sistema só e servir o mesmo conteúdo para todo mundo. O design responsivo costuma ser uma forma de implementar One Web com usabilidade adequada para todos.

Há ainda quem fale de uma outra técnica chamada **RESS — Responsive Design + Server Side Components**. Você serve a mesma URL e a mesma página, mas ajusta no *server-side* algumas coisas sabendo qual browser está sendo usado para o acesso. Faz-se *user-agent sniffing*, que é menos preciso, mas ajuda em algumas coisas, como decidir se um link para ligação direta no celular deve ser exibido.

Você também pode usar server-side para manter a mesma URL (One Web), mas servir páginas específicas (não responsivas). Ou seja, há combinações diversas dessas técnicas (one web + responsividade, one web sem responsividade, dois sites + responsividade etc.).

Este livro foca em **Web Design Responsivo + One Web**. Quer dizer, são explorados exemplos e cenários de sites únicos adaptando-se por meio de media queries, CSS flexível e recursos flexíveis (imagens, vídeos etc.). Esse cenário ninguém questiona se é responsivo ou não.

1.4 PARA QUEM É ESTE LIVRO?

Web design responsivo é um assunto que, principalmente no Brasil, ainda pode ser considerado como novidade e não é amplamente usado. Até os mais veteranos (e, talvez, principalmente eles) desconhecem suas técnicas e/ou, mesmo, seu conceito — gostaria de acreditar que não é o caso da maioria dos desenvolvedores, mas, caso seja, temos um trabalho de conscientização ainda maior a fazer! Por isso, querendo ou não, este livro destina-se a quem já possui alguns conhecimentos específicos na área de desenvolvimento web.

Ele é destinado a todo e qualquer profissional, estudante ou aspirante a desenvolvedor web que deseja começar ou aprofundar seus estudos em web design responsivo. O objetivo foi encontrar a melhor maneira de expor esse assunto de uma forma que possa ser considerada cronologicamente adequada e didaticamente satisfatória.

Portanto, caso queira ter seu primeiro contato com web design responsivo — ou, caso já o tenha tido, mas, por algum motivo, não tenha entendido muito bem como as coisas funcionam —, relaxe, prossiga com a leitura e tenha a certeza de que, ao final do processo, você será mais um conhecedor dos mistérios profundos do *responsive web design*!

1.5 PARA APROVEITAR MELHOR ESTE LIVRO

Para entender e absorver bem o conteúdo, é bom que você já esteja um pouco familiarizado com HTML e CSS. Saber um pouco de JavaScript é desejável, não para acompanhar os conteúdos principais e mais importantes do livro, mas, sim, para acompanhar o conteúdo complementar, dicas extras, o *plus* do design responsivo, que muitas vezes precisa de um bom JavaScript para ser implementado.

Mas fique tranquilo! Não é na maioria das técnicas que isso é preciso. E quando o for, tratam-se de materiais não-essenciais, sem os quais você ainda conseguirá entender como funciona o design responsivo para web.

Então, se você já é profissional do ramo e já passou (ou se encontra ou se interessa) pela área de front-end, não terá maiores problemas ao continuar com a leitura.

Se você é do back-end e não se interessa tanto pelo que acontece "lá na frente", acredito que, ainda assim, poderá aproveitar os conteúdos, dado que, geralmente, mesmo o profissional não sendo exclusivamente alocado a uma área de atuação deve, pelo menos, conhecer as "nuances" do trabalho de

outros colegas de equipe.

Caso você seja um praticante-militante de web design, profissional de usabilidade, UX e afins, então: ou você dá uma lida rápida em alguns tutoriais sobre HTML/CSS (caso já não saiba alguma coisa); ou, fique sabendo desde já, seu aproveitamento maior será sobre como planejar um site desde sua concepção. Para ser responsivo — principalmente, na filosofia *mobile first*, que será apresentada —, será obrigação **sua** (ou da equipe a qual você faça parte) pensar e apresentar as soluções visuais para os diferentes *breakpoints* do design (assunto que será visto mais à frente), uma vez que pode ser preciso ou acordado na equipe que "tal" ou "qual" projeto contará com web design responsivo.

Quer dizer, não importa muito em que área dentro da "linha de produção" de desenvolvimento web você se encontre no momento. O que deve ficar claro é que você precisa conhecer a marcação e sintaxe do HTML, sabendo quando aplicar e para que servem as diferentes tags da linguagem. Além disso, também deve conhecer CSS, seus seletores e as respectivas propriedades e possíveis valores; e um pouco de JavaScript, caso queira se aprofundar um pouco mais em web design responsivo e aprender algumas técnicas interessantes, também mostradas e explicadas neste livro (no caso, menos importante).

Capítulo 2

PRINCÍPIOS DE UM WEB DESIGN RESPONSIVO

O conceito de web design responsivo já foi apresentado: é aquele web design que *responde* a quaisquer dispositivos/resoluções e, devido a uma série de características técnicas bem específicas, é bem apresentado em qualquer um deles.

Chegou o momento de aprofundar um pouco mais em suas origens, raízes e inspirações e em como aplicar o *responsive web design*.

2.1 COMO SURGIU O WEB DESIGN RESPONSIVO

Existe um site muito, muito bom e tradicional na web, que é o **A List Apart**: http://www.alistapart.com/. Ele existe desde 1998 e, com o passar dos anos, nunca deixou de brindar seus leitores com artigos e livros de excelente qualidade.

Um dos seus escritores é **Ethan Marcotte** que, em meados de 2010, publicou um artigo intitulado *Responsive Web Design*, que mudaria para sempre a forma como se faz o design para web: http://www.alistapart.com/articles/responsive-web-design/.

Em uma tradução livre, este artigo começa assim:

"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 projetar (o web design) para essa flexibilidade".

E, no decorrer do artigo, Ethan explica seus conceitos e sugestões (usando tecnologia que já era existente na época de sua publicação) para que as páginas fossem projetadas utilizando o que ele chamou de **web design responsivo**. Até um site de exemplo foi mostrado no artigo (http://ow.ly/ate9K), o que deixou a comunidade de desenvolvedores em polvorosa!

Inspirado por conceitos de arquitetura e filosofia, sua proposta com a publicação daquele artigo foi mostrar um conjunto de técnicas que garantem responsividade a um web design.

Devido ao enorme feedback positivo e à ânsia da comunidade por maiores explicações e pormenores sobre o assunto, no ano seguinte foi lançado o livro que, por razões óbvias, levou o mesmo nome do artigo que revolucionara a maneira de se fazer design na web: **Responsive Web Design**.

E, como não poderia deixar de ser, muito da "inspiração" deste livro veio do que foi compartilhado por Ethan em seus escritos. Portanto, nada mais natural do que prestar homenagem ao homem que liberou para o mundo energias de conhecimento web profundo, baseando vários dos conteúdos que aqui constam em seus ensinamentos.

Thank you, Ethan!

2.2 A TRINCA TECNOLÓGICA DO WEB DESIGN RESPONSIVO

Para se conseguir implementar um design responsivo, 3 tecnologias principais (ou modos de aplicar essas tecnologias, se preferir) estão envolvidas:

1. Layout fluido;
2. Imagens e recursos flexíveis;
3. Media queries.

Desenvolver sites com **layouts fluidos**, ou seja, primar pela não especificação de medidas fixas no layout do projeto desde sua concepção, torna possível que haja uma "adaptação natural" e automática do que se apresenta na tela. Portanto, seja qual for a resolução do dispositivo que fez o acesso, evita-se barras de rolagem inconvenientes e/ou conteúdos "cortados", não exibidos em sua completude.

Mas de nada adiantaria se o conteúdo se moldasse às mais diferentes resoluções se as imagens (e outros recursos) do site também não se adaptassem e não fossem **flexíveis**, não é verdade? Então, por meio de técnicas variadas, é possível fazer com que os *assets* (recursos como imagens, vídeos etc.) da página também sejam flexíveis para garantir que a experiência do visitante prime pela excelência, independentemente do dispositivo que esteja sendo usado.

Para as mais diferentes resoluções dos aparelhos, as necessidades e desejos de uso do site podem se alterar. Afinal, uma pessoa não estaria errada ao imaginar que alguém que acessa uma página através de um celular, cuja tela pode não ter uma boa

resolução, não precisa de uma *sidebar*. O que esse usuário provavelmente está procurando são os conteúdos principais; então, nada mais justo do que não onerar a visita com a exibição de uma barra lateral. As opções presentes na sidebar poderiam, por exemplo, serem apresentadas no fim de todo o conteúdo.

É com as **media queries** que é possível ocultar, fazer aparecer e reposicionar elementos e interações conforme a resolução atual que está sendo usada no momento da visitação. Afinal, o site não precisa (e, na verdade, não deve) ter, exatamente, a mesma aparência e disposição de elementos em qualquer resolução. Um iPad, um celular Nokia e um monitor de 27 polegadas possuem espaços, resoluções e necessidades bem diferentes.

Sabendo usar essa *trinca tecnológica*, é possível criar designs responsivos para a web. É objeto deste livro fazer a apresentação e respectivas explicações (e integrações) de cada uma dessas principais tecnologias. Mas, antes, é interessante ter uma noção de alguns problemas comuns que trouxeram os desenvolvedores web até este ponto e conhecer alguns fatos notáveis sobre dispositivos, web e a relação entre eles.

2.3 RESOLUÇÕES DE TELA

Como se pôde ver quando foram apresentadas estatísticas do mundo mobile na seção *Estatísticas do mundo mobile* do capítulo anterior, os números envolvendo este novo marco tecnológico são surpreendentes! Esses números mostram também que a tendência é que mais e mais dispositivos surjam, muitos a cada ano. Então, juntando os já existentes com os que são lançados a cada período programado de tempo, temos um bocado de devices para prestar

atenção... Ou nem tanto?

Não é preciso uma extensa reflexão para constatar que é humanamente impossível prestar atenção e dar suporte a todo e qualquer dispositivo mobile existente e que venha a existir. Na verdade, nem é preciso que isso seja feito.

O atual (e não eficiente) paradigma de acesso multidispositivo

Sem essa ideia em mente para o desenvolvimento, mesmo antes de se começar a escrever o primeiro caractere de código, o que se faz é o seguinte: escolhe-se uma técnica qualquer para identificar qual é a resolução do dispositivo (ou o dispositivo, em si) e, a partir do resultado, carrega-se recursos condizentes (CSS, JavaScript, imagens etc.) para aquele device.

Algo que também não é difícil de acontecer é a pessoa ser redirecionada para um site à parte, feito para uma resolução ou dispositivo específico (um *site compatível*). Não é preciso escrever parágrafos infindáveis sobre os custos de produção e retrabalho envolvidos ao se usar essa abordagem, não é verdade? Sem contar que, muitas vezes, o que se busca é uma experiência parecida e não algo completamente diferente do site "normal".

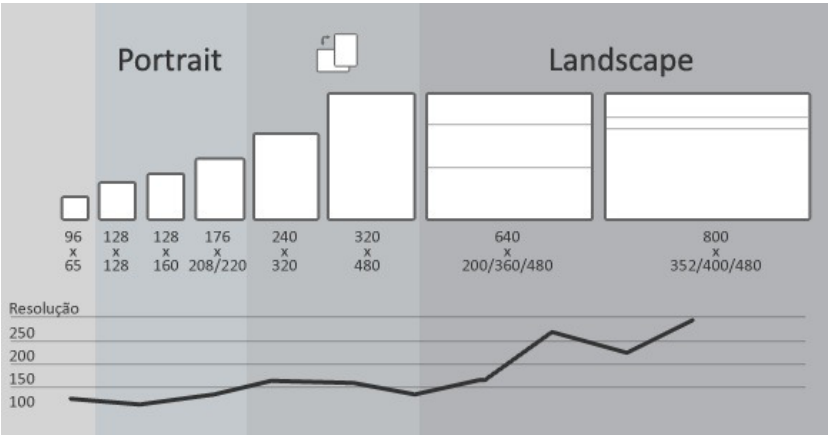
O equívoco comum é ainda não ter entendido que, na verdade, o que importa não é o tamanho físico da tela ou do dispositivo em si, mas sua **resolução**! Tendo a atenção voltada a elas, é possível desenvolver para uma gama incrível de dispositivos sem preocupar-se com quais são e/ou seu tamanho físico com uma facilidade muito maior.

Ou você quer ter de atualizar seus *scripts* de identificação de dispositivos sempre que o mais novo smartphone do momento for lançado no mês que vem? Claro que não! E, certamente, qualquer profissional da área de desenvolvimento web, passando pelo estagiário de HTML ao gestor do negócio, também não!

Como bem disse **Rik Schennink**, *"neste exato momento, pessoas estão realizando telefonemas segurando tablets em suas cabeças, enquanto outros estão jogando GTA em seus telefones até os dedos sangrarem. Há telefones que são tablets e tablets que são telefones, não há nenhuma maneira de determinar onde um telefone termina e um tablet começa"*.

Para ter-se uma ideia, veja uma pesquisa feita com aparelhos vendidos entre 2005 e 2008 que contemplou a análise de 400 devices diferentes (http://ow.ly/b72At) com intenção de levantar as principais resoluções de dispositivos usados para acesso à web, também levando em considerações as orientações desses dispositivos: *portrait* (retrato) e *landscape* (paisagem).

Constatações interessantes foram feitas, como por exemplo, a diferença de tamanho entre o menor e o maior dispositivo que participaram da amostra — que chega a ser mais de **23 vezes —**, e o aumento considerável de *ppi* (*pixels per inch* ou *densidade de pixels*) conforme o avanço das tecnologias!



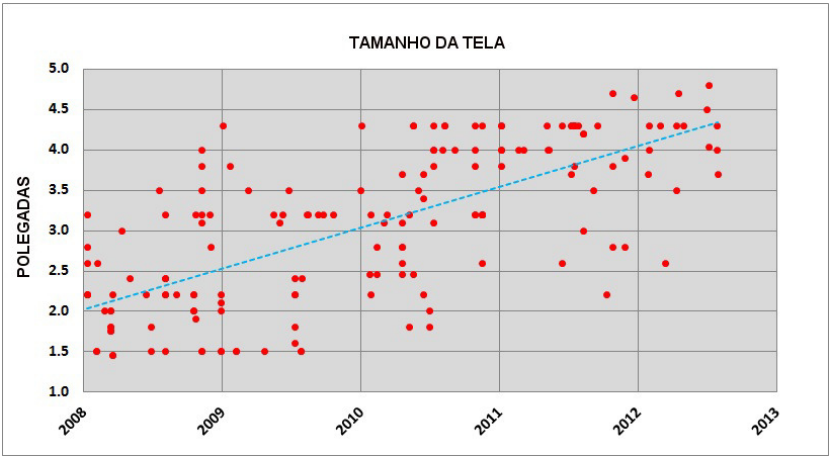
Para ficar mais claro e trazer o entendimento com exemplos mais concretos, veja o exemplo de alguns dispositivos para ter-se noção das diferenças de resolução que podem existir. Veja quais as resoluções específicas de alguns dos aparelhos comuns que podem ser encontrados em qualquer lugar. Para facilitar, o gráfico disponibilizado no blog **WebDev-il** (http://ow.ly/bazAV) faz uma separação lógica dos devices por Sistema Operacional.

Figura 2.2: Resolução do aparelho por Sistema Operacional

Perceba que, nesse gráfico, existem duas indicações para iPhone: uma é o retângulo que representa a resolução de 320x480 e a outra, que é a maior de todas, 640x960! A diferença é que o

iPhone 4 tem uma resolução bem superior, e o iPhone 5 aumentou ainda mais essa diversidade, com seus 640x1136 pixels.

Perceba que essas são resoluções que, apesar de mostrarem uma realidade de alguns anos atrás, não deixam de transmitir o conceito que se quer passar: o tamanho físico da tela não importa! Contudo, esses tamanhos aumentaram com o passar do tempo:



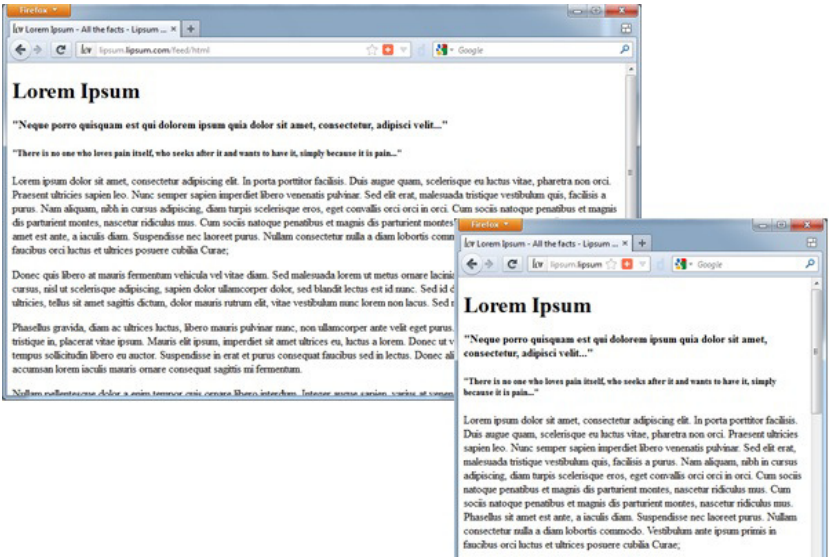
De qualquer maneira, o que importa não é o tamanho físico da tela ou do dispositivo em si, mas, sim, sua **resolução**. Fixou?

Capítulo 3

LAYOUT FLUIDO

Um **layout fluido** (ou, como originalmente chamado por Ethan Marcotte, **grid flexível**) é o primeiro passo para desvendar os mistérios do web design responsivo. Para conseguir-se um layout fluido em um projeto web, a principal medida a ser tomada é: **não usar medidas absolutas no CSS!**

Aliás, você, ao começar a codificar seus primeiros exemplos de HTML, já notou isso, mas aposto que esqueceu! Ou não se lembra de como aquele parágrafo cheio de "*Lorem ipsum...*" adaptava-se a qualquer tamanho de janela do navegador? E por quê? Porque não havia nenhuma especificação de largura! Simples assim.



Portanto, ao especificar tamanhos, espaçamentos, margens, *paddings*; enfim, ao estipular qualquer medida referente ao layout das páginas do site, é preciso usar valores relativos, como **porcentagens** e **ems**.

Antes de prosseguir, veja quais são os tipos de medidas em CSS.

3.1 TIPOS DE MEDIDA EM CSS

Embora existam outros tipos de medidas em CSS, os 4 principais são: **pixels**, **pontos**, **porcentagens** e **ems**.

- **Pixel (px):** é a unidade de medida fixa mais usada nas CSS. Comumente falando, um pixel é um ponto indivisível na tela de exibição de um dispositivo —

embora, mais recentemente, tenhamos vários tipos de pixel (http://ow.ly/atmIu). Não raramente, web designers preferem usar essa medida para fazer uma estrutura HTML/CSS em *pixel perfect*, não medindo esforços para estruturar seus documentos para que fiquem idênticos à imagem do design que lhes foi entregue.

- **Ponto (point):** pontos são tradicionalmente utilizados para CSS de impressão. Um ponto é igual a 1/72 polegadas. Assim como pixels, pontos são unidades de tamanho fixo.
- **Ems (em):** ele é uma unidade escalável. Quando se trata do tamanho da fonte, 1em é igual ao tamanho atual da fonte do elemento-pai. Por exemplo, se o tamanho da fonte do elemento é 12pt, 1em é igual a 12pt. *Ems* são escaláveis por natureza. 2em seria igual a 24pt; 0.5 seria 6pt e assim por diante.
- **Porcentagem (%):** a unidade porcento é muito parecida com a unidade *em*, mas possui algumas diferenças fundamentais, principalmente o fato de que o atual tamanho da fonte é igual a 100% (ou seja, 12pt = 100%). Durante seu uso, o texto permanece totalmente escalável para dispositivos móveis.

Como foi mostrado, a diferença básica entre essas medidas é que os dois primeiros são unidades de medida fixas e, os dois últimos, variáveis. Eles são relativos, escaláveis, adaptam-se e mantêm relações de tamanho com outros elementos de um documento — eles possuem um **contexto**.

Portanto, fica claro que usar *pts* ou *pxs* não é o adequado para

web design responsivo em um primeiro momento, já que, com a prática, experiência e/ou demanda do projeto, pode ser necessário especificar alguns valores fixos em determinadas situações. Isso deixa como "favoritas" as 2 opções restantes. Porém, qual a diferença entre usar ems ou porcentagens?

Ems x Porcento

Devido às características que foram analisadas previamente, parece que é a mesma coisa usar uma ou outra. Entretanto, existem diferenças.

O importante é saber que, quando falamos em layout, marcar unidades com porcento fornece uma exibição mais consistente e acessível para os visitantes. Quando as configurações de exibição do cliente alteram-se, as medidas marcadas com `%` mudam de maneira mais razoável, permitindo que a legibilidade seja preservada, bem como a acessibilidade e o design visual.

Portanto, apesar de ser possível usar qualquer um dos tipo de medidas relativas, existe uma espécie de consenso entre os desenvolvedores (e este "consenso" surgiu por meio de muitos testes e experiências com responsividade): **usar porcentagem para lidar com tamanhos no layout** (larguras, margens, espaçamentos etc.) e **usar ems para lidar com fontes**. `em` pode até ser usado fora de textos, mas vai ser sempre uma **medida relativa ao** `font-size` ; já o porcento é relativo ao `font-size` quando usada em `font-size` , mas, quando usada com outras medidas, é **relativa à largura do elemento-pai**.

Futuramente, será possível trabalhar amplamente com novos recursos, como as medidas **rem** (http://ow.ly/yyuAs), **vw**, **vh**

(http://ow.ly/yyuKt) e novas possibilidades de estruturação de layouts com Flexbox (http://ow.ly/yyuSe). Mas, por enquanto, façamos um bom uso do que temos como certeza.

3.2 A FÓRMULA MÁGICA DO WEB DESIGN RESPONSIVO

Quando um projeto tem por objetivo contar com design responsivo desde sua concepção, é importante que os desenvolvedores envolvidos pensem *de forma relativa*. Ou seja, em implementar esse layout valendo-se de medidas relativas no CSS. Relembrando, deve-se pensar em atribuir as medidas do layout em *porcentagem* e o tamanho de fontes em *em*.

Se na "antiga maneira de pensar" houvesse, por exemplo, um título com tamanho de fonte de 24px, então seria preciso converter essa medida em *em*. Neste ponto, você deveria estar se perguntando: "Mas como converter para *em* e continuar com a aparência de 24px?". E essa, meu caro colega de profissão, é a pergunta certa a se fazer!

Existe uma **fórmula** para realizar esse cálculo. Uma forma simples e rápida para realizar a conversão de tamanhos absolutos para tamanhos relativos em layouts para a web. E não é nada com que você deva se preocupar, ou sentir um arrependimento profundo ao imaginar que deveria ter prestado mais atenção às aulas de matemática da época do colégio. É uma fórmula tão simples que, por si só, contém a beleza e a elegância do web design responsivo:

Fórmula mágica do web design responsivo

```
Alvo / Contexto = Resultado
```

- **Alvo:** elemento-alvo com a medida atual;
- **Contexto:** onde o elemento-alvo está (baseado no elemento-pai);
- **Resultado:** o valor relativo que se está procurando.

Com essa fórmula rápida, é possível realizar cálculos simples para saber o resultado de conversões de medidas absolutas do CSS para *medidas relativas*. E isso vale tanto para o cálculo de **tamanho de fontes** (existe uma certa convenção de que o tamanho padrão de fontes em navegadores para desktop é de 16px) quanto para **medidas de layout**!

Então, voltando ao exemplo, agora fica simples descobrir qual a equivalência em `em` de 24px. Basta aplicar a fórmula: **24 / 16 = 1,5.**

Ou seja, ao pegar o alvo (o título) de `24px` e dividir pelo contexto — no caso, seu elemento-pai, que é `body`, e possui tamanho de fonte de 100%, ou seja, `16px` —, tem-se como resultado *1,5*. Como o assunto é tamanhos relativos de fontes em CSS, o resultado final é **`1.5em`**.

Imagine que exista um link dentro do título, que teria `11px` de tamanho no planejamento. É preciso converter isso para medida relativa. Aplicando a fórmula, `11 / 16 = 0,6875`, certo? **Errado!**

Já que o link se encontra dentro do título, **o contexto mudou!** Agora, o elemento-alvo não está mais no *contexto geral* da página (`body`). Então, para conseguir o resultado correto, a variável de contexto da fórmula muda: **11 / 24 = 0,45833333333333**

O elemento que em medidas absolutas deveria ter `11px` está dentro do contexto do título (que, em medidas absolutas, teria `24px`), que resulta em `0.45833333333333em` .

Ethan Marcotte sugere que, quando o resultado da divisão tiver tantos números depois da vírgula, se evita arredondá-lo e se dá preferência ao resultado com tantos números quanto sua calculadora consegue exibir. No entanto, caso queira usar um limite de 4 números, também funciona.

3.3 EXEMPLO DE UM LAYOUT FIXO

Como você já deve ter inferido, é possível aplicar a fórmula tanto para descobrir a equivalência relativa de tamanhos de fontes quanto para desenvolver layouts fluidos. Para um melhor entendimento, tome por base um modelo de página fictícia que ajudará bastante a fixar os conceitos que estão sendo tratados.

HTML de um layout fixo

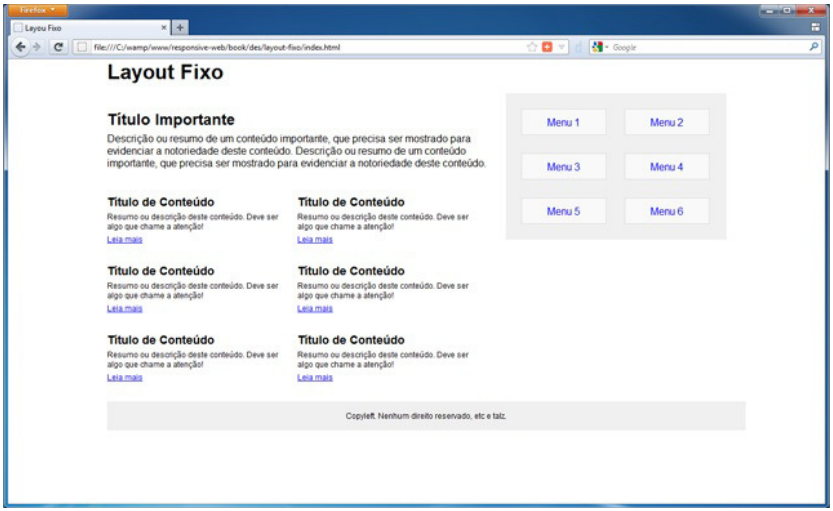
Veja um exemplo do que é muito comum de se encontrar no dia a dia da navegação na web: um site com **layout fixo**. A intenção é mostrar, através de um layout simples, a marcação de HTML e CSS que se encontra comumente na web.

HTML E CSS

O objetivo deste livro não é discutir teorias, técnicas ou condutas de escrita de HTML e CSS. Fogem do escopo da obra questões como espaçamentos, uso ou não de classes ou IDs, identação etc.

Se você quer conhecer mais, não deixe de ver o livro do Lucas Mazza, de HTML5 e CSS3, também lançado pela **Casa do Código**.

Então, imagine uma página de um site com **layout fixo** que se pareça com isso:



Seu HTML é simples — como todo HTML deve(ria) ser — e não apresenta nada diferente do convencional. Primeiramente,

tem-se toda a estrutura da página "englobada" por um contêiner. Essa é uma técnica muito comum:

```
<div class="container">
    [...]
</div>
```

Existem 3 divisões principais dentro do `div container` : um cabeçalho, o espaço para a inserção dos conteúdos principais e um rodapé. Novamente, sem nenhuma novidade para as técnicas mais básicas de estruturação em HTML:

```
<div class="container">
    <header class="header">
        [...]
    </header>

    <div class="content">
        [...]
    </div>

    <footer class="main-footer">
        [...]
    </footer>
</div>
```

Neste modelo de estrutura HTML, `header` e `footer` costumam sempre ter o mesmo conteúdo, independentemente da página do site que está sendo visitada no momento. Portanto, fica óbvio que os conteúdos principais do site ficam em `content` .

Dentro de `content` , também existem 2 "porções de conteúdo": a parte dos conteúdos principais, propriamente ditos, e uma barra lateral.

```
<div class="container">
    <div class="content">
        <div class="content-main">
            [...]
```

```
        </div>

        <aside class="content-sidebar">
            [...]
        </aside>
    </div>
</div>
```

Portanto, a estrutura desse site simples, demonstrada por meio dessa página-exemplo, basicamente teria a mudança de conteúdos acontecendo em `content-main`. Nela, há uma chamada para o conteúdo mais recente ou mais importante em dado momento (conhecida por muitos como `hero`), seguida de chamadas para outros conteúdos, cada uma mostrando seu respectivo título, um resumo/chamariz e um link para acessar a página na íntegra.

Como a estrutura foi idealizada para ser mantida e se repetir em todas as páginas do site, para os outros tipos de páginas, basta alterar os elementos HTML dentro de `content`.

No caso, essa página de exemplo foi estruturada com 6 dessas "chamadas", mas a estrutura permite que sejam feitas quantas sejam precisas (preferencialmente em quantidade par, para manter sua estrutura de apresentação).

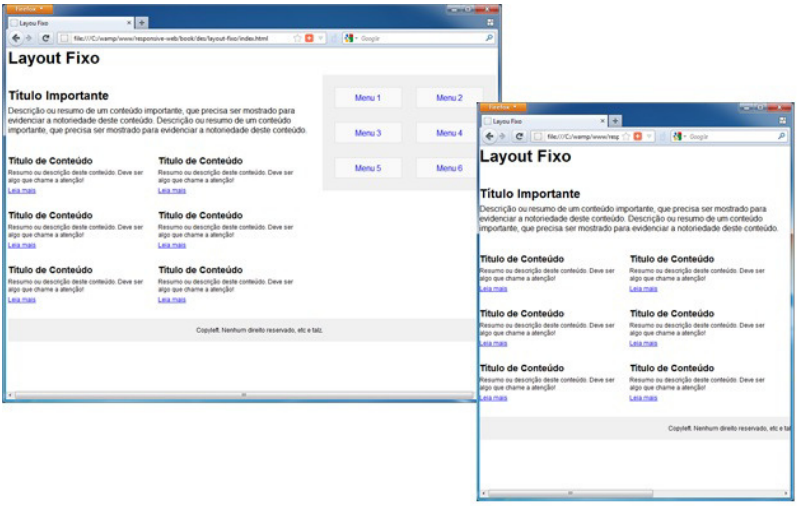
```
<div class="content">
    <div class="content-main">
        <article class="hero">
            <h2>
                Título Importante
            </h2>

            <p class="brief">
                Descrição ou resumo de um conteúdo
                importante, que precisa ser mostrado para
                evidenciar a notoriedade deste conteúdo.
            </p>
        </article>
```

```
        <div class="last-contents">
            <article class="last-content-call">
                <h2 class="secondary-title">
                    Título de Conteúdo
                </h2>

                <p class="brief">
                    Resumo ou
                    descrição deste conteúdo.
                    Deve ser algo que chame a atenção!
                </p>

                <a href="#">Leia mais</a>
            </article>

            <article class="last-content-call">
                <h2 class="secondary-title">
                    Título de Conteúdo
                </h2>

                <p class="brief">
                    Resumo ou
                    descrição deste conteúdo.
                    Deve ser algo que chame a atenção!
                </p>

                <a href="#">Leia mais</a>
            </article>

            [...]
        </div>
    </div>

    <aside class="content-sidebar">
        <nav class="main-nav">
            <ul>
                <li><a href="#">Menu 1</a></li>
                <li><a href="#">Menu 2</a></li>
                <li><a href="#">Menu 3</a></li>
                <li><a href="#">Menu 4</a></li>
                <li><a href="#">Menu 5</a></li>
                <li><a href="#">Menu 6</a></li>
            </ul>
        </nav>
    </aside>
```

```
</div>
```

O código completo pode ser visto neste link: https://gist.github.com/3630605.

Até então, tudo normal. Inclusive, quando alguém redimensiona a janela (ou, o que é mais comum, acessa-a usando uma resolução menor do que a mostrada no exemplo), o conteúdo continua no mesmo lugar e barras de rolagem do navegador aparecem. Enfim, tudo normal e como o esperado:



Quando alguém acessa através de um dispositivo móvel, principalmente com os existentes atualmente, o próprio device encarrega-se de fazer sua mágica para que todo o site seja apresentado na telinha como se aplicasse um *zoom out* nele todo.

Com isso, a pessoa tem uma visão geral dele (por o estar vendo "pequenininho") e, caso algo seja do seu interesse, por meio de um botão pressionado ou gesto feito na tela do dispositivo, consegue-se dar um *zoom in* e ver melhor o que quer.

A partir do momento que a pessoa já viu o que queria, ela ou vai "deslizando" o site para encontrar outros conteúdos do seu interesse, ou dá um *zoom out* para ter novamente a *visão topográfica*. Esse processo pode acontecer mais vezes, caso a pessoa não consiga encontrar a informação com rapidez.

Essa facilidade que os dispositivos móveis oferecem realmente facilita o processo de navegação, já que nem todos os sites estão tecnicamente preparados para apresentar seus conteúdos nas mais diversas resolução possíveis. Sim, de fato é um processo que, depois de algum tempo, é cansativo, repetitivo e, concordemos, não atende aos padrões e "gostos" de usabilidade um pouco mais rigorosos.

O ideal é que seu layout (e, logicamente, seu conteúdo) seja bem apresentado em todos os tipos de resolução. O desejável (e, muitas vezes e cada vez com mais frequência, o *esperado*) é que ele seja inteligente o bastante para apresentar o melhor conteúdo através da melhor maneira possível, não importando qual dispositivo e/ou qual resolução estejam sendo usados no momento do acesso.

CSS de um layout fixo

Esse HTML de exemplo deve ser acompanhado por um pouco de CSS. Como você verá, é um CSS que não tem absolutamente nada demais, servindo apenas para dar uma leve estilizada e

apresentar a página de um layout fixo de uma maneira um pouco mais agradável e organizada.

No exemplo, será usada a forma mais básica de *CSS reset*, servindo somente para ilustrar a necessidade de haver um esforço inicial na criação das folhas de estilo de uma padronização entre os diferentes navegadores.

Existem soluções mais interessantes, como http://necolas.github.io/normalize.css/, que padroniza estilos entre os navegadores em vez de usar regras para "zerar" valores. Mas, para fins de exemplo, um *reset* básico já basta. Além disso, será usada uma fonte comum não-serifada para os elementos.

```
* {
margin: 0;
padding: 0;
}

html {
font-family: Arial, Helvetica, sans-serif;
}
```

Começando a mexer nos elementos especificados no HTML, já é possível trabalhar com o `.container`, que é o *wrap* de toda a página. A ideia é ter um site centralizado, algo bastante comum nos dias de hoje. A largura "ideal" para tal modelo de página não existe; portanto, é aceitável arbitrar o que melhor se aprouver.

```
.container {
margin: 0 auto;
width: 960px;
}
```

Vale dizer que é possível também mexer no título principal dessa página, colocando um tamanho de fonte considerável para que não haja dúvidas quanto à sua importância.

```
h1 {
font-size: 32px;
}
```

A intenção é que haja uma padronização para elementos do tipo `.brief`. Portanto, não há mal algum em já estilizar este "elemento genérico" do site-exemplo.

```
.brief {
margin: 5px 0;
}
```

Os conteúdos principais do site ficam em `.content` que, por sua vez, abriga `.content-main` e `.content-sidebar`, respectivamente. Dentro de ambos, existem mais estilos para que o exemplo fique em uma estrutura interessante e funcional.

```
.content-main {
float: left;
width: 593px;
}

    .hero {
    margin: 25px 0;
    }

    .last-contents {
    font-size: 12px;
    }

        .last-content-call {
        float: left;
        margin: 15px 15px 15px 0;
        width: 280px;
        }

            .last-content-call .secondary-title {
            margin-bottom: 0;
            }

            .last-content-call .brief {
            margin: 5px;
```

```
            }

.content-sidebar {
background-color: #F0F0F0;
float: right;
padding: 10px;
width: 322px;
}

    .main-nav ul {
    list-style-type: none;
    }

        .main-nav li {
        background-color: #F9F9F9;
        float:left;
        margin: 15px;
        outline: 1px solid #DEDEDE;
        text-align: center;
        width: 130px;
        }

            .main-nav a {
            display: block;
            padding: 10px;
            text-decoration: none;
            }
```

Por fim, existe o `.main-footer` que, no momento, serve somente para exibir algumas informações de *copyleft*. Entretanto, nada impediria de conter, por exemplo, um menu secundário com links de menor importância que o menu principal, atualmente localizado na barra lateral.

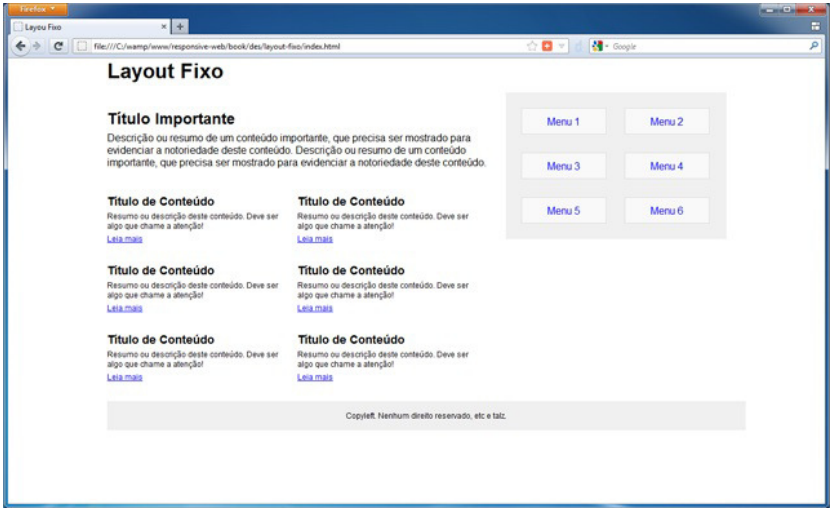
```
.main-footer {
background-color: #F0F0F0;
clear: both;
float: left;
font-size: 12px;
margin: 15px 0;
padding: 15px;
text-align: center;
```

```
width: 100%;
}
```

Como prometido, é um CSS bastante simples, sem implementações "polêmicas" e/ou técnicas mirabolantes para conseguir efeitos diversos. O objetivo deste livro não é esse.

> O código completo desse layout fixo pode ser encontrado em https://gist.github.com/3630828.

Com isso, não há mais nada a se fazer no arquivo `style.css` que, como você pode ter observado no código completo do exemplo, é referenciado e fará com que a camada de conteúdo receba sua devida estilização. Abrindo ambos os arquivos em um bom navegador, você será premiado com uma tela muita parecida com:



E o "esqueleto" da página de exemplo está terminado. Não foi nada demais, obviamente. Mas, neste ponto, você pode estar se perguntando: "E agora, como transformar esse layout fixo em um layout fluido?". E, novamente, essa é exatamente a pergunta certa a se fazer!

3.4 METATAG VIEWPORT

Você verá como converter um layout fixo em layout fluido. Porém, primeiro, deve ater-se a um dos mais importantes "detalhes" técnicos quando o assunto é web design responsivo: a **meta tag viewport**.

Meta tags

Teoricamente, se você está lendo este livro, já deve saber o que é uma **meta tag** HTML. De qualquer forma, uma breve revisão pode ser útil.

O termo *meta tag* é, na verdade, formado por duas palavrinhas que se juntam para dar seu significado. E, para entendê-lo, é preciso entender ambas as palavras separadamente:

- **Meta** é um prefixo grego que significa algo como "após", "que ultrapassa", "que engloba", "que está além". Ele não é um prefixo usado somente no desenvolvimento web; você já deve ter ouvido e lido palavras como "metabolismo" e "metamorfose". Esse prefixo "meta" refere-se à "coisa sobre a própria coisa" ou, em outras palavras, dados sobre dados, informação sobre a informação.

- **Tag** é uma palavra do inglês que significa "etiqueta" ou "rótulo". Tanto nos EUA como no Brasil ou em qualquer parte do mundo, etiquetas servem para identificar, nomear e marcar algo ou alguma coisa para que possa ser identificado e/ou corretamente catalogado.

Então, juntando esses dois termos, temos a formação de *meta tag*, que são *tags* que descrevem o documento web a qual pertencem ("informação sobre a informação"). As meta tags são para descrever informações sobre sites e páginas que as contém; informam sobre qual conteúdo está ali e mostram informações extras a respeito desse conteúdo.

Elas devem ser elementos-filho de `head` no HTML. Sua sintaxe obedece à seguinte convenção:

```
<meta name="nome-da-meta" content="conteudo-da-meta">
```

Então, para qualquer das meta tags existentes que podem ser usadas, a convenção para seu uso é essa: onde se informa, primeiramente; de qual meta se está inserindo; e, posteriormente, qual valor se dá.

No documento do W3C, *Visual formatting model* (http://ow.ly/baEmW), é explicado que:

"Agentes do usuário de mídia contínua em geral oferecem aos usuários uma viewport (uma janela ou outra área de visualização na tela) através da qual os usuários consultam um documento. Os agentes podem alterar o layout do documento quando a viewport é redimensionada. Quando a viewport é menor que a área da tela em que o documento é renderizado, o agente deve oferecer um

mecanismo de rolagem".

Ou seja, dentro do contexto de desenvolvimento web, *viewport* é a parte visível da página web que é renderizada pelos navegadores (desprezando painéis, barras de menu, plugins, barra de rolagem etc.).

O que é a meta tag viewport

Os navegadores mobile tentam exibir páginas web feitas somente para desktop (no momento, a imensa maioria de todas elas) ajustando, automaticamente, o *zoom* do display. Isso pode ser problemático para os sites que já foram planejados/otimizados para telas pequenas — todos os que você fará a partir da leitura deste livro, certo?

Felizmente, existe uma meta tag para contornar essa característica dos navegadores. É a *meta tag viewport*. Assim como qualquer outra, ela possui o formato:

```
<meta name="viewport" content="">
```

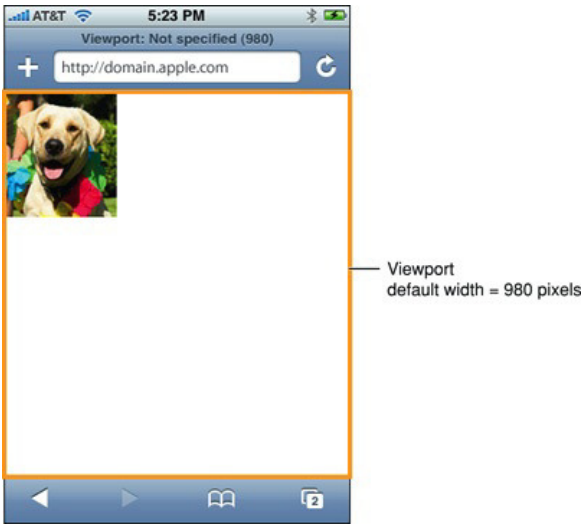
Sendo que, em `content`, é possível especificar uma diversidade de parâmetros e valores conforme o tipo de visualização que se configurar às páginas. Se preferir, pense que, com a meta tag viewport, é possível apresentar "resoluções personalizadas" aos visitantes para determinados dispositivos. Os principais e mais usados parâmetros de `content` são:

- **width**: define a largura da viewport;
- **height**: define a altura da viewport;
- **initial-scale**: define a escala inicial (zoom) da viewport.

É possível usar mais de um parâmetro, ou mesmo todos, como valor de `content` da meta tag viewport.

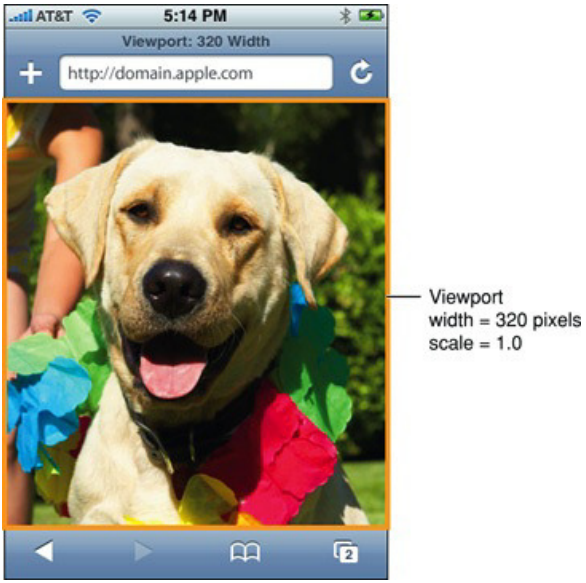
Exemplos de uso da meta tag viewport

A própria Apple, que originalmente propôs e colocou em uso a meta tag viewport, apresenta alguns exemplos que elucidam bastante a questão (http://ow.ly/baGbr). Veja uma imagem de `320x356px` renderizada, apresentada em um iPhone, usando as configurações padrão de viewport.



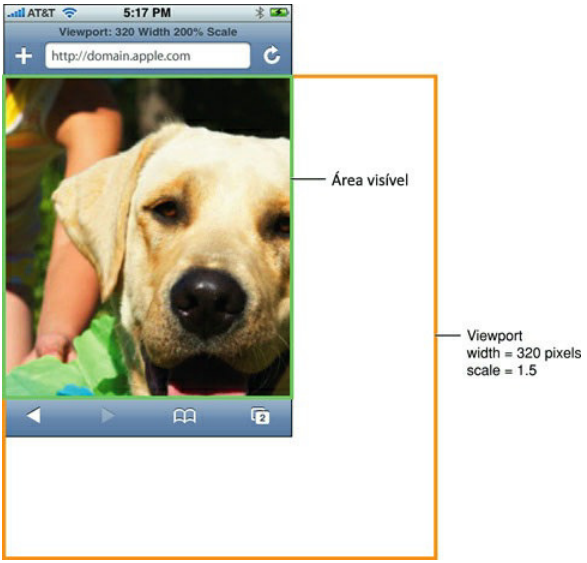
Agora, a página com o zoom na viewport sendo o mesmo do da própria imagem.

```
<meta name="viewport" content="width=320,initial-scale=1">
```



No entanto, como tratam-se de configurações que podem ser escolhidas conforme a necessidade do projeto, o zoom na viewport pode ser maior ou menor do que a área visível. Se o zoom é maior, então será possível deslizar a tela para ver mais da página.

```
<meta name="viewport" content="width=320,initial-scale=1.5">
```



E, como citado, também é possível definir um zoom menor para a viewport:

```
<meta name="viewport" content="width=320,initial-scale=0.5">
```

Nos dispositivos móveis, a pessoa também pode fazer *zoom in* e *zoom out* usando gestos. Quando isso acontece, ela não altera o tamanho da viewport; altera sua **escala**! Consequentemente, os gestos de *pan* e *zoom* não alteram o layout da página.

A configuração ideal de viewport

Observando a tabela anterior, é possível observar que diferentes configurações de viewport (e de zoom na viewport) produzem diferentes resultados e possibilidades de apresentação do site em condições controladas. Porém, há uma questão que não ajuda em nada quando pensamos em possibilidades de uso de `width` : para contemplar todos os dispositivos móveis, seria preciso conhecer a largura de cada um deles!

O iPhone, nos exemplos de tela apresentados, tem largura de `320px` . Mas, tomando como exemplo somente os dispositivos exemplificados no gráfico de resolução de aparelhos visto anteriormente (figura *Resolução do aparelho por Sistema Operacional* do capítulo anterior), seria muito trabalhoso identificar, marcar e fazer manutenções no código com todos os possíveis tamanhos.

Na verdade, existe uma forma de especificar *automaticamente* o tamanho do device. É como se houvesse uma variável que, por padrão, já contém a largura do dispositivo que está realizando o acesso naquele momento:

```
<meta name="viewport" content="width=device-width">
```

Isso indica ao navegador que o `width` da meta tag viewport é o tamanho da largura do dispositivo!

Em condições normais, também é interessante que, assim que a página renderize, não haja alterações na escala inicial. Por isso, juntamente com `width=device-width` da meta tag, é bom usar a `initial-scale` em `1`. Esta, para a maioria dos sites desenvolvidos segundo a filosofia do design responsivo, é a **configuração ideal de viewport**:

```
<meta name="viewport"
    content="width=device-width,initial-scale=1">
```

Valores possíveis da meta tag

A seguir, veja a tabela — originalmente exibida no Nokia Developer (http://ow.ly/baLZP) — com os parâmetros possíveis da meta tag viewport, exemplos de valores, respectivos valores-padrão e a descrição de cada um deles.

Parâmetro	Exemplos	Valores possíveis	Padrão	Descrição
width	width=device-width width=320	1 a 10000 pixels ou device-width	device-width	Largura da viewport (em pixels)
height	height=device-height height=640	1 a 10000 pixels ou device-height	device-height	Altura da viewport (em pixels)
initial-scale	initial-scale=0.5 initial-scale=2.0	0.1 to 10	1.0 (sem zoom)	Zoom para o primeiro carregamento da página (valores altos causam "zoom in"; valores baixos, "zoom out")
user-scalable	user-scalable=no	yes ou no	yes	Se a página permite zoom pelo usuário
minimum-scale	minimum-scale=0.5 minimum-scale=1	0.1 a 10	0.25 (ajuste pelo valor de width)	Limite do "zoom out" do usuário
maximum-scale	maximum-scale=1.5 maximum-scale=2.5	0.1 a 10	5 (ajuste pelo valor de width)	Limite do "zoom in" do usuário
target-densityDpi	target-densityDpi=high-dpi	device-dpi (sem ampliação), medium-dpi (zoom de 1.3) ou high-dpi (zoom de 1.5)	device-dpi	Modifica a densidade da tela, ampliando a página para uma maior resolução (DPI)

Lembre de tomar cuidado e prestar bastante atenção no uso combinado dos parâmetros possíveis para não causar efeitos desagradáveis e/ou não planejados. Por exemplo, ao especificar:

```
<meta name="viewport"
    content="width=device-width,initial-scale=1,
             maximum-scale=1">
```

Isso impossibilitaria a pessoa que está fazendo o acesso de dar *zoom* nas páginas do site. Se esse for realmente o objetivo que se quer alcançar, tudo bem; mas, repetindo: atenção ao usar parâmetros combinados!

3.5 CONVERTENDO UM LAYOUT FIXO EM FLUIDO

Fazer com que você saiba desenvolver layouts fluidos (ou grids flexíveis) é um dos maiores objetivo deste livro. Depois que os

conceitos aqui mostrados já estiverem *internalizados* e bastante prática tiver sido feita, você conseguirá imaginar com *foco* a construção de layouts fluidos, antes mesmo de colocar o dedo no teclado para digitar a primeira regra CSS!

Enquanto isso não acontece, há um exercício interessante que vai clarear bastante alguns dos principais conceitos sobre web design responsivo: converter o layout fixo mostrado anteriormente em um layout fluido. Como? Simplesmente aplicando a "fórmula mágica" do web design responsivo!

Como vamos mexer mais com as medidas especificadas anteriormente, não há necessidade de revisarmos todo o CSS mostrado. Em vez disso, atenha-se aos trechos que mais interessam ao exercício de conversão, começando com o `.container`.

A largura de `.container` é de `960px`. Obviamente, isso não ajuda em nada a montagem de um grid flexível, dado que a principal e primeira medida a ser tomada para se construir um é **usar somente medidas relativas no CSS**. Vamos revisar a fórmula: **Alvo / Contexto = Resultado**.

No caso, como trata-se do elemento `contêiner`, não existe um contexto que sirva como base. Então, da mesma forma que foi especificada uma largura para ele arbitrariamente, é possível fazer o mesmo com a largura com medida relativa que,
quando vista no browser, se pareça o máximo possível com o valor que se tem atualmente. Depois de alguns testes, chegou-se a:

```
.container {
    margin: 0 auto;
    width: 67.5%; /* +/- 960 */
}
```

Como pôde ser visto nesse sucinto trecho de código, trata-se de uma excelente prática. Quando trabalharmos com layouts fluidos, é necessário fazer um comentário na frente do valor convertido (ou construído) para não perder a noção do que se está fazendo. Fica a dica!

Em relação a esse valor de **67.5%** para o contêiner, foi totalmente opcional e servindo somente para ilustrar o site-exemplo. Didaticamente, configura-se como interessante preservar a característica de sites centralizados para a adaptação e mais fácil assimilação dos conceitos de web design responsivo. Entretanto, outras abordagens poderiam ter sido usadas:

1. Deixar o `width` em `960px` fixos e transformar todo o resto em porcentagem e `em`. Não tão eficiente, já que ficará com esta largura independentemente da tela ser maior ou menor doque `960px`. Não seria uma solução responsiva.
2. *Responsivar* para telas mobile menores do que `960px`, mas manter os `960px` para telas maiores. Usaria-se `max-width:960px` e continuaria igual para resoluções *desktop-like*. Porém, ao ser exibido em resoluções menores, ajustaria-se por não ter largura fixa.
3. A solução anterior é responsiva para telas pequenas, mas possui limite de 960px em telas maiores. Também é possível responsivar para todas, colocando `width:100%` no contêiner. Para projetos que precisem/requeiram de largura total em quaisquer resoluções, essa abordagem seria a ideal.

Dando prosseguimento, agora é preciso passar o tamanho da fonte de `h1` para medida relativa. A coisa está começando a ficar mais fácil, uma vez que, neste caso, já temos todas variáveis da

fórmula! Lembra-se de que existe um certo padrão entre os navegadores de colocar o tamanho inicial das fontes em `16px`? Isso é tudo o que é preciso no momento.

Quer dizer, `32` (**Alvo**) dividido por `16` (**Contexto**) é igual a `2` (**Resultado**). Lembre-se também daquela outra observação importante: quando se trata de medidas relativas para fontes, usa-se a unidade *em*; quando é para medidas de layout, **porcento**.

```
/* Antes */
h1 {
    font-size: 32px;
}

/* Depois */
h1 {
    font-size: 2em; /* 32 / 16 */
}
```

Depois, é hora de trabalhar com o próximo elemento do layout: `.content`. Usando a fórmula novamente: **15** dividido por **960** (a medida de nossa já finada largura fixa) é igual a **0.015625**.

Mas espere: ao se aplicar esse valor ínfimo ao layout, as coisas não funcionam como o esperado! Como estamos lidando com porcentagens, é preciso **multiplicar o valor por 100**. *Matemágica* pura!

```
/* Antes */
.content {
    margin: 15px 0;
}

/* Depois */
.content {
    margin: 1.5625% 0; /* 15 / 960 */
}
```

Finalmente, as margens fluidas estão devidamente configuradas e prontas para o que der e vier! Com isso em mente, o elemento `.content-main` não vai representar nenhum perigo. Entretanto, se você estiver realmente prestando atenção aos códigos de exemplo, poderá estar com uma pergunta intrigante em sua mente: "Onde está o *contexto*?".

> **Como calcular o contexto em elementos sem contexto imediato**
>
> Quando o contexto esperado não existe, isto é, o elemento-contêiner não apresenta largura definida, é preciso procurar no elemento imediatamente ascendente até que se encontre a definição de uma largura.

Ou seja, como o contexto de `.content-main`, que é o elemento `.content`, não possui declaração explícita de largura, é preciso verificar o próximo elemento-ascendente que, no caso, é `.container`. Este, sim, com declaração (prévia) de largura de `960px`. Por isso, neste caso, o contexto será de **960**:

```
/* Antes */
.content-main {
    float: left;
    /* Medida maior da Proporção Áurea aplicada ;-) */
    width: 593px;
}

/* Depois */
.content-main {
    float: left;
    /* 593 (.content-main) / 960 (.container) */
    width: 61.7708%;
```

```
}
```

O elemento `.hero` (e seu descendente `.brief`), dentro de `.content-main`, também precisa ter suas medidas convertidas. Algo interessante a se levar em conta é que o cálculo da relatividade de `.hero .brief` é `5 / 593 = 0.008431`.

Contudo, apesar de esta ser uma ciência (pretensiosamente) exata, há muito do **fator humano** envolvido. Portanto, quando estivermos desenvolvendo grids flexíveis e algum resultado, apesar de preciso, não for do agrado, não há nada de mais em (caso isso seja responsabilidade sua) arredondar este valor.

```
/* Antes */
.hero {
    margin: 25px 0;
}

.brief {
    margin: 5px 0;
}

/* Depois */
.hero {
    margin: 4.2158% 0; /* 25 / 593 */
}

.brief {
    margin: 1% 0; /* 5 / 593 */
    /* = 0.8431%, que, opcionalmente, pode ser arredondado*/
    /* para 1% */
}
```

Prosseguindo com a conversão dos outros elementos descendentes de `.content-main`:

```
/* Antes */
.last-contents {
    font-size: 12px;
}
```

```
    .last-content-call {
        float: left;
        margin: 15px 15px 15px 0;
        width: 280px;
    }

        .last-content-call .brief {
            margin: 5px;
        }

/* Depois */
.last-contents {
    font-size: .75em; /* 12 / 16 */
}

    .last-content-call {
        float: left;
        /* 15 / 593 (.content-main) */
        margin: 2.5295% 2.5295% 2.5295% 0;
        width: 47.2175%; /* 280 / 593 */
    }

        .last-content-call .brief {
            margin: 1.7857% 0; /* 5 / 280 */
        }
```

A conversão da barra lateral com todos seus descendentes fica assim:

```
/* Antes */
.content-sidebar {
    background-color: #F0F0F0;
    float: right;
    padding: 10px;
    width: 322px;
}

    .main-nav ul {
        list-style-type: none;
    }

        .main-nav li {
            background-color: #F9F9F9;
```

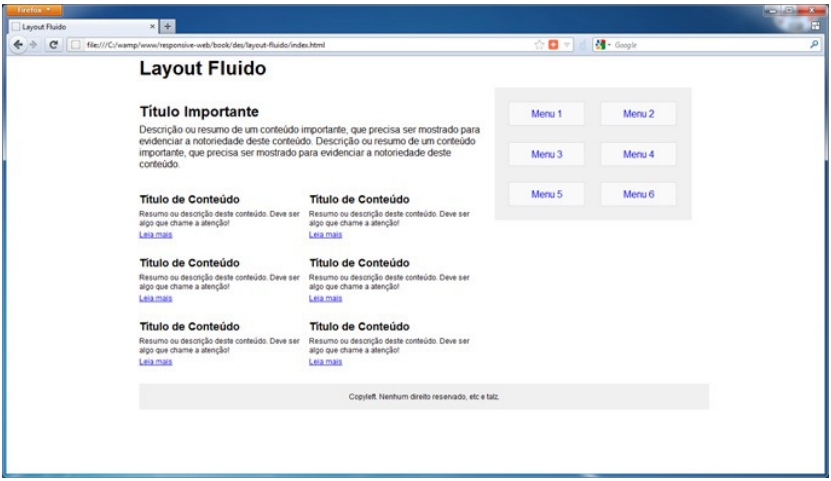
```
            float: left;
            margin: 15px;
            outline: 1px solid #DEDEDE;
            text-align: center;
            width: 130px;
        }

            .main-nav a {
                display: block;
                padding: 10px;
                text-decoration: none;
            }

/* Depois */
.content-sidebar {
    background-color: #F0F0F0;
    float: right;
    padding: 1.0416%; /* 10 / 960 */
    width: 33.5416%; /* 322 / 960 */
}

    .main-nav ul {
        list-style-type: none;
    }

        .main-nav li {
            background-color: #F9F9F9;
            float: left;
            margin: 4.6583%; /* 15 / 322 (.content-sidebar) */
            outline: 1px solid #DEDEDE;
            text-align: center;
            width: 40.3726%; /* 130 / 322 */
        }

            .main-nav a {
                display: block;
                padding: 7.6923%; /* 10 / 130 */
                text-decoration: none;
            }
```

Por fim, o rodapé:

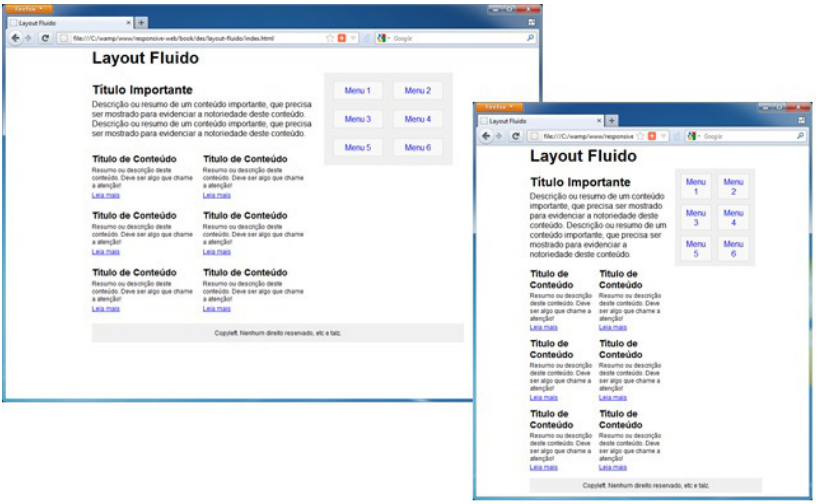
```
/* Antes */
.main-footer {
```

```
    background-color: #F0F0F0;
    clear: both;
    float:left;
    font-size: 12px;
    margin: 15px 0;
    padding: 15px;
    text-align: center;
    width: 100%;
}

/* Depois */
.main-footer {
    background-color: #F0F0F0;
    float:left;
    font-size: .75em; /* 12 / 16 */
    margin: 1.5625% 0; /* 15 / 960 */
    padding: 1.5625%; /* 15 / 960 */
    text-align: center;
    width: 100%;
}
```

Ao pegar o CSS convertido e abri-lo como estilização do HTML que já temos, em um primeiro momento não dá para notar nada de mais, parecendo que nenhuma mudança aconteceu. Com exceção de alguns poucos pixels para lá ou para cá, a aparência é a mesma do layout fixo.



Mas, ao redimensionarmos a janela, os caminhos da responsividade para a web começam a serem revelados:



Perceba que, ao redimensionarmos a janela do browser (e,

claro, pessoas acessarem a página usando resoluções diferentes), o que acontece é que a apresentação está agradavelmente padronizada e **responsiva**!

Com isso, você dá o primeiro grande passo para a construção de layouts responsivos para a web!

Mas, apesar de já ser um enorme passo em sua jornada, o desenvolvimento de um site responsivo não se dá somente a partir de um grid flexível. Como mostrado, ainda existem mais princípios de um web design responsivo a serem explorados: recursos flexíveis e media queries.

Capítulo 4

IMAGENS E RECURSOS FLEXÍVEIS

4.1 CSS PARA IMAGENS FLEXÍVEIS

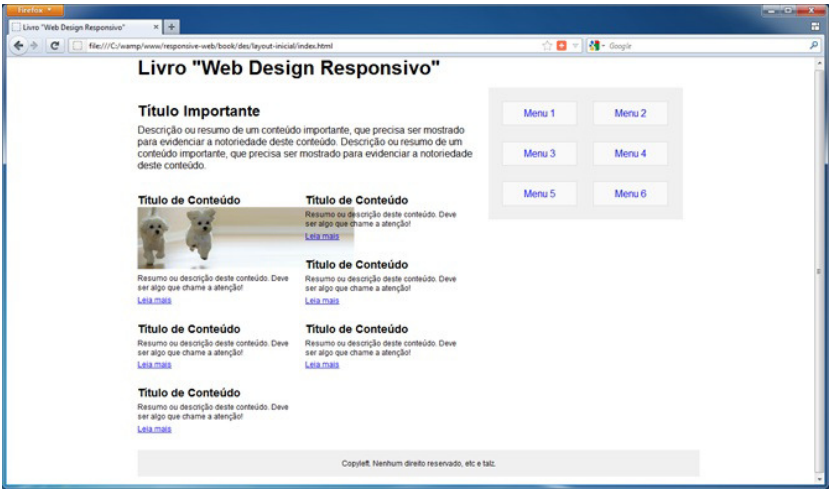
Continuando com o desenvolvimento do HTML de exemplo, suponha que, agora, foi decidido que cada `.last-content-call` terá uma imagem para ilustrar seu conteúdo logo abaixo do título. Como o projeto já está em um viés de desenvolvimento por meio da filosofia de um web design responsivo, é preciso encontrar uma solução que faça com que as imagens se adequem ao espaço devido e sejam bem apresentadas, independentemente da resolução.

Para começar, veja como fica o exemplo com uma imagem colocada na primeira chamada. O HTML muda para:

```
<article class="last-content-call">
    <h2 class="secondary-title">
        Título de Conteúdo
    </h2>

    <img src="cute-dogs.jpg" width="350">

    <p class="brief">
        Resumo ou
        descrição deste conteúdo.
        Deve ser algo que chame a atenção!
    </p>
```

```
    <a href="#">Leia mais</a>
</article>
```

E o resultado é:



Veja que coisa: a imagem, corretamente marcada com seu `width` no HTML possui 350px de largura. Acontece que, seja qual for a resolução em que a página estiver sendo vista, essa quebra acontecerá.

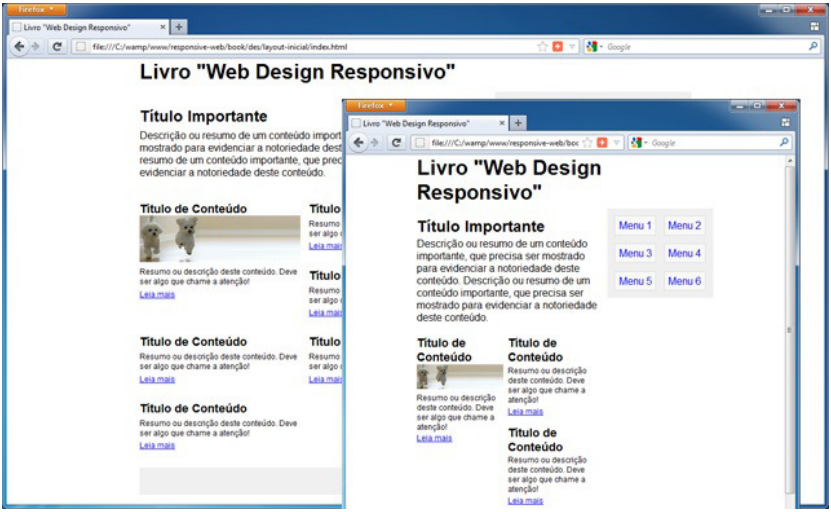
Pode parecer incrível, mas, para conseguir um efeito interessante de imagens flexíveis, só é preciso um pouquinho de CSS. Nada de regras extravagantes, nem de propriedades recém-lançadas anteontem que nenhum navegador ainda suporta; é o bom e velho CSS, com propriedades familiares a qualquer um com pouco tempo de estudos em folhas de estilo. Só é preciso acrescentar no início do CSS:

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

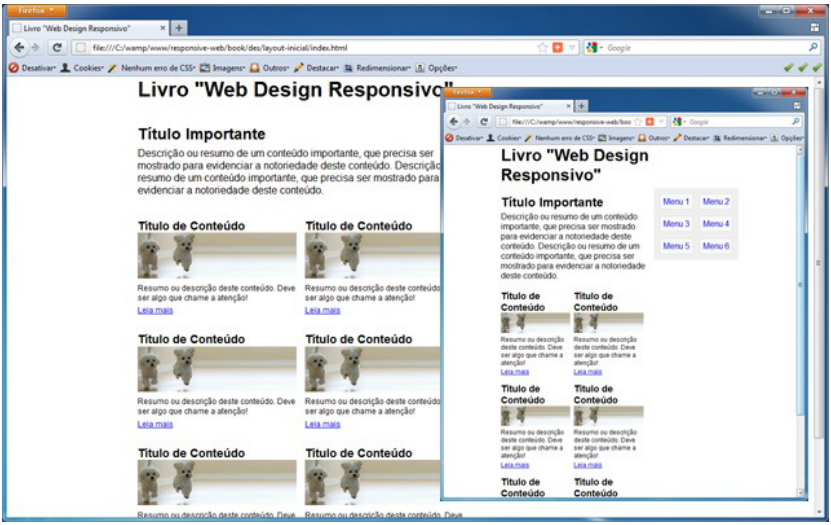
Segundo a documentação sobre `max-width` do W3C (http://ow.ly/bcKdv), essa propriedade CSS permite restringir larguras de conteúdo dentro de um determinado intervalo. No caso, o "intervalo" é 100%. Ou seja, a regra genérica de CSS significa que as imagens podem ter qualquer largura até no máximo 100%. Mas "100%" de quê? Do elemento em que estão

contidas!

No caso da imagem dos cachorrinhos usada, ela é filha de `.last-content-call`, portanto, a regra vale para a largura deste elemento ou, no caso de sua falta, para a largura do elemento ascendente e assim por diante. Depois de aplicada a regra `max-width: 100%;`, as coisas já começam a ficar mais interessantes:



Com essa regra em uso, já é possível complementar o layout fluido e ver como ele fica com todos os `.last-content-call` e com suas respectivas imagens:



Uma observação importante é que, caso seja preciso preocupar-se com o IE8 na implementação de imagens responsivas, é preciso colocar `width:auto` em imagens dentro de elementos com `float` sem declaração explícita de largura (http://ow.ly/cO3fP).

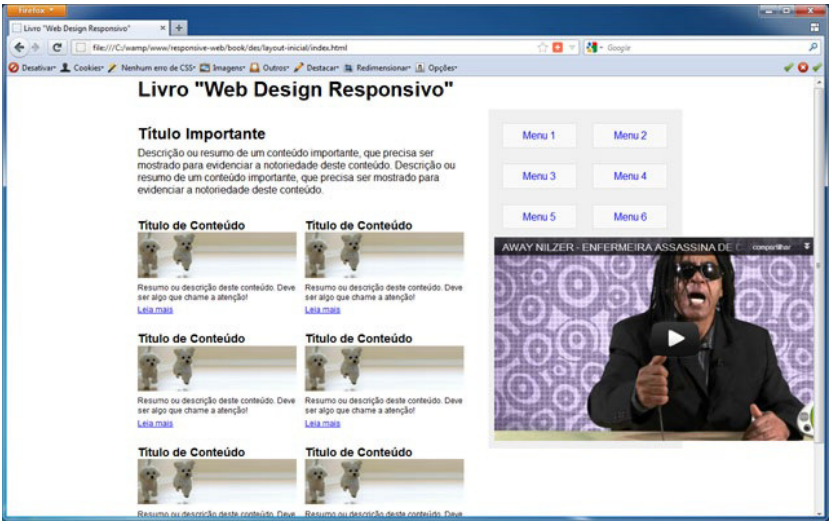
4.2 CSS PARA OUTROS RECURSOS FLEXÍVEIS

Mais uma vez, uma requisição foi feita à equipe de desenvolvimento do site. Dessa vez, será preciso que um vídeo seja sempre carregado na barra lateral, logo abaixo do menu principal. O `aside`, então, muda para o seguinte:

```
<aside class="content-sidebar">
    <nav class="main-nav">
        <ul>
            <li><a href="#">Menu 1</a></li>
            <li><a href="#">Menu 2</a></li>
            <li><a href="#">Menu 3</a></li>
```

```
            <li><a href="#">Menu 4</a></li>
            <li><a href="#">Menu 5</a></li>
            <li><a href="#">Menu 6</a></li>
        </ul>
    </nav>

    <iframe src="http://www.youtube.com/embed/1YLKyVwExOs"
     width="560" height="315" frameborder="0" allowfullscreen>
    </iframe>
</aside>
```

O vídeo escolhido não importa; o fato é que o código foi copiado diretamente do recurso de compartilhamento do YouTube (mas poderia ter sido qualquer outro serviço de compartilhamento de vídeos). Do jeito que de lá veio, cá está. Ao observarmos o resultado, é possível perceber que algo não saiu como deveria.

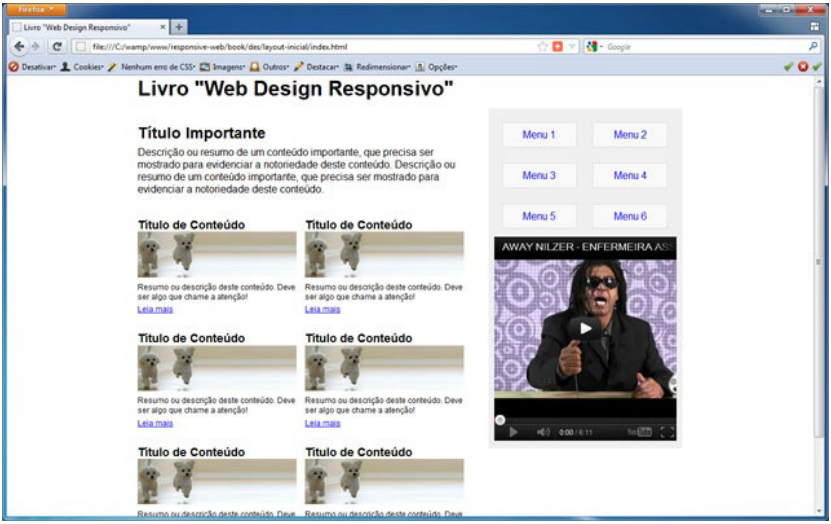


O *embedded* está com o tamanho totalmente fora do aceitável e está quebrando o layout. No caso, ele não pode extrapolar os

limites do `aside` e deve aparecer corretamente em qualquer resolução e dispositivo em que o site for acessado.

Pode parecer incrível, mas a mesma regra que aplicamos anteriormente para imagens também pode ser aplicada para outros tipos de mídia e recursos, incluindo `iframe`, `object`, `embed` e `video`! Logo, basta atualizar para:

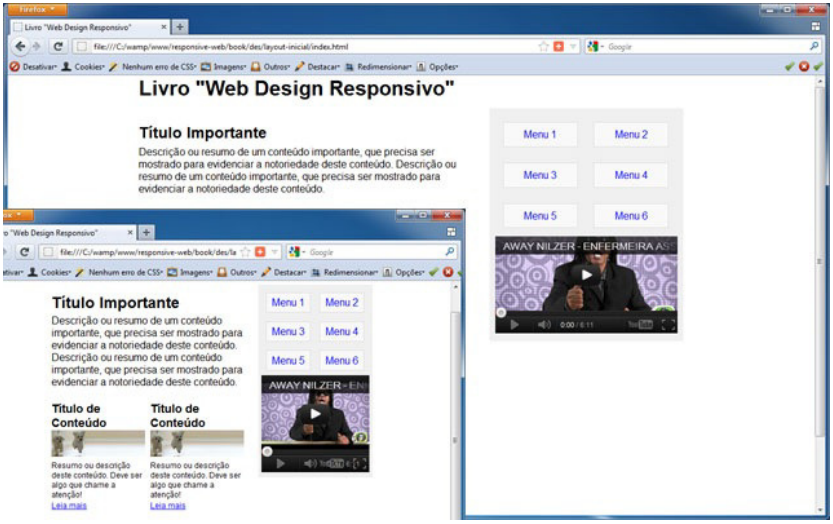
```
img,
iframe,
object,
embed,
video {
    max-width: 100%;
}
```



Assim, como o esperado, o HTML contempla boa apresentação/posicionamento em qualquer resolução. Em outras palavras, agora possui um **vídeo flexível**!

É possível, ainda, incrementar essa regra. Nos testes realizados, a responsividade dos recursos não é comprometida de nenhuma maneira; o acréscimo facultativo só altera um pouco como os elementos são apresentados:

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



Sem muitas alterações no comportamento dos elementos, é possível mudar o valor da propriedade `height` de `auto` para `100%`. A decisão da mudança fica ao critério e/ou necessidades do projeto.

O YouTube coloca, automaticamente, tarjas pretas em cima e

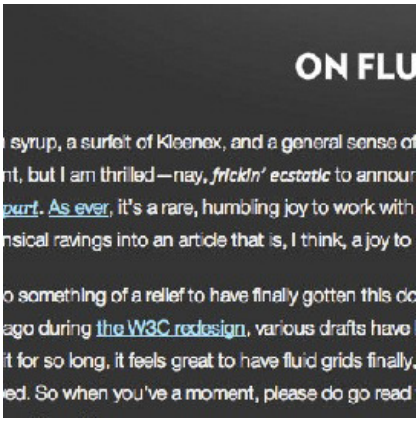
em baixo do vídeo ao se redimensionar, preservando seu *ratio* original. Entretanto, caso seja preciso usar vídeos de outras fontes (ou até mesmo vídeos com hospedagem própria), existem algumas soluções:

- **Creating Intrinsic Ratios for Video:** artigo muito interessante de **Thierry Koblentz** que apresenta uma técnica que permite aos navegadores determinarem as dimensões de vídeos com base na largura do elemento que estão contidos (http://ow.ly/eqra4).
- **FitVids.js:** plugin jQuery simplíssimo de usar que dá a qualidade da responsividade a vídeos incorporados em qualquer página (http://ow.ly/eqrpN).
- **Fluid Width Video:** artigo de **Chris Coyier** que aborda a inserção de vídeos com as tags `iframe`, `object` e `embed` (http://ow.ly/eqrT9).

Exceção à velharia: imagens redimensionadas com qualidade em IE6~7

Aproveitem a exceção do que acabou de ser devidamente enfatizado: uma solução de como melhorar a qualidade de imagens redimensionadas e exibidas em ambiente Windows (antes do 7), sobretudo com uso de IE6~7.

Acontece que, usando a web sob estas condições, imagens redimensionadas ficam com a qualidade muito comprometida. Com a devida referência a Ethan Marcotte, reproduzo a imagem de exemplo em que o guru do web design responsivo mostra um exemplo dessa questão.



Existem várias boas tentativas por parte da comunidade para solucionar isso. Inclusive, o próprio Ethan desenvolveu um JavaScript para tratar essas situações.

Basicamente, o script varre todo o documento, troca as imagens pelo famoso pixel transparente de 1px quadrado e aplica o filtro proprietário da Microsoft `AlphaImageLoader` em cada uma. A partir disso, sempre que a janela do navegador for redimensionada, o script recalcula e adequada proporcionalmente a largura e a altura das imagens, redimensionando o pixel *spacer*. A seguir, o script de Ethan.

```
var imgSizer = {
    Config : {
        imgCache : []
        ,spacer : "/path/to/your/spacer.gif"
    }

    ,collate : function(aScope) {
        var isOldIE =
            (document.all && !window.opera
            && !window.XDomainRequest)
            ? 1 : 0;
```

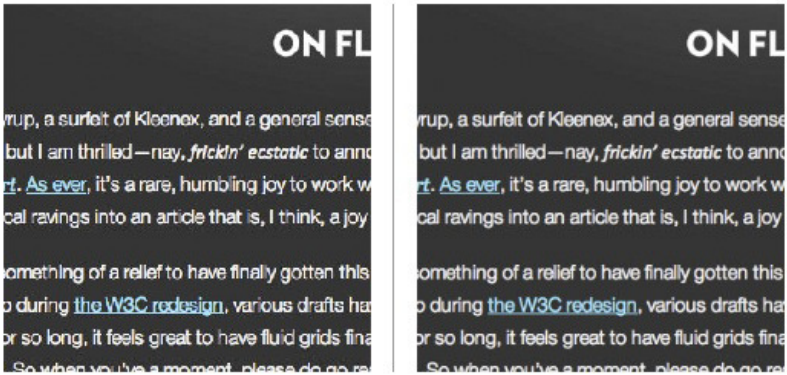
```
    if (isOldIE && document.getElementsByTagName) {
        var c = imgSizer;
        var imgCache = c.Config.imgCache;

        var images = (aScope && aScope.length) ? aScope :
        document.getElementsByTagName("img");
        for (var i = 0; i < images.length; i++) {
            images[i].origWidth = images[i].offsetWidth;
            images[i].origHeight = images[i].offsetHeight;

            imgCache.push(images[i]);
            c.ieAlpha(images[i]);
            images[i].style.width = "100%";
        }

        if (imgCache.length) {
            c.resize(function() {
                for (var i = 0; i < imgCache.length; i++) {
                 var ratio = (imgCache[i].offsetWidth /
                 imgCache[i].origWidth);
                 imgCache[i].style.height =
                     (imgCache[i].origHeight * ratio) + "px";
                }
            });
        }
    }
}

,ieAlpha : function(img) {
    var c = imgSizer;
    if (img.oldSrc) {
        img.src = img.oldSrc;
    }
    var src = img.src;
    img.style.width = img.offsetWidth + "px";
    img.style.height = img.offsetHeight + "px";
    img.style.filter =
        "progid:DXImageTransform.Microsoft.AlphaImageLoader
    (src='" + src + "', sizingMethod='scale')"
    img.oldSrc = src;
    img.src = c.Config.spacer;
}

// Ghettomodified version of Simon Willison's addLoadEvent()
// http://simonwillison.net/2004/May/26/addLoadEvent/
```

```
    ,resize : function(func) {
        var oldonresize = window.onresize;
        if (typeof window.onresize != 'function') {
            window.onresize = func;
        } else {
            window.onresize = function() {
                if (oldonresize) {
                    oldonresize();
                }
                func();
            }
        }
    }
}
```

Por meio de um comentário condicional, podemos fazer o script entrar em ação somente quando o browser for Internet Explorer:

```
<!--[if IE]>
    <script src="imgsizer.js"></script>
<![endif]-->
```

Depois da sua implementação, a diferença na qualidade das imagens renderizadas sob as condições expostas é nítida:



Frequentemente, a partir de sugestões e casos da comunidade, o script é revisado e atualizado. Portanto, passe a acompanhar o artigo em que, originalmente, Ethan disponibilizou-o: *Fluid Images* (http://ow.ly/bg5hO). Aproveite também para conhecer mais detalhes e pormenores de como a técnica foi desenvolvida e como funciona.

4.3 O PROBLEMA DE IMAGENS EM LAYOUTS FLUIDOS

Com as técnicas apresentadas para imagens flexíveis em web designs responsivos, é possível fazer um bom número de testes para
fixar os conhecimentos, pois isso já é a "base" a respeito de gráficos para a responsividade na web. Entretanto, surge uma
questão (se não, **a** questão) sobre apresentar imagens em qualquer resolução: **peso** e **tamanho**!

Quando o acesso é *tradicional*, feito via desktop ou notebook, é justificável supor que o usuário está vendo a página através de um monitor de tamanho relativamente confortável. Isso, juntamente com o fato de que a velocidade média de conexão à internet no Brasil é de 512Kb a 2Mb (http://ow.ly/bg78j), não traz maiores problemas para a pessoa que navega em sites com imagens.

Já quando o acesso segue a tendência crescente de acesso mobile, tem-se questões e problemas seríssimos. Esses aparelhos costumam ter uma tela menor e, muitas vezes, resoluções não muito boas. Além disso, a velocidade da conexão móvel é inferior a de outros meios. Logo, é preciso pensar sobre como servir imagens adequadamente aos visitantes, levando em conta seu **peso**

(tamanho em KB) e seu **tamanho** (quantidade de pixels).

Segundo as técnicas mostradas até então, uma mesma imagem é apresentada seja para qual resolução for. E, conforme o caso, ela é redimensionada para adaptar-se à realidade da resolução do dispositivo que está sendo usado. Isso revela um ponto crítico: o **mesmo arquivo** que é apresentado em uma boa velocidade de conexão, vista em uma tela de tamanho considerável, também é servido e renderizado em dispositivos móveis!

Mesmo que sem querer, pessoas utilizando dispositivos mobile que acessam a página serão penalizadas por não disporem de um display maior e/ou uma conexão mais veloz. O cidadão acessando seu site pseudo-otimizado para dispositivos móveis será obrigado a esperar carregar aquela imagem de 1500px, pesando 80Kb. Ele a verá sendo lentamente renderizada, já redimensionada (e distorcida), demorando muito mais tempo para terminar de ser baixada e exibida em sua conexão 3G do que ele demora para tomar a decisão de procurar o que precisa no site do concorrente.

Se o acesso é feito via mobile, é mais lógico (e melhor para quem acessa) que uma versão reduzida da imagem, de menor peso e tamanho, seja enviada a fim de aprimorar a experiência da visita. Afinal, **71%** das pessoas esperam que o acesso a um site móvel seja tão rápido quanto no desktop (http://ow.ly/bg7Wp)!

4.4 TÉCNICAS PARA IMAGENS FLEXÍVEIS EM WEB DESIGNS RESPONSIVOS

Tendo visto o básico necessário para apresentar imagens flexíveis em designs responsivos para a web, cabe ressaltar que

existem diversas técnicas para que essa entrega de imagens seja a mais adequada possível (dentro da perspectiva de quem desenvolveu a respectiva técnica, evidentemente). A seguir, é possível conhecer uma série de técnicas para imagens flexíveis; como sempre, não existe "a melhor"; mas, sim, a que mais se enquadra ao seu estilo de desenvolvimento, dentro de um projeto específico.

Riloadr

Riloadr autointitula-se um independente framework carregador de imagens responsivas. Ele realmente possui várias boas características. Só para citar algumas:

- Relativamente fácil de aprender e usar;
- Oferece bastante controle sobre como as imagens responsivas vão se comportar;
- Opção de carregamento sob demanda (*lazy load*) das imagens;
- Suporta browsers com JavaScript desabilitado (ou sem JavaScript);
- Não faz múltiplas requisições para a mesma imagem.

A ideia principal do Riloadr é deixar o atributo `src` completamente fora das tags de imagem, substituindo-o pelo atributo `data-src` . Por meio da mágica do JavaScript, é possível obter resultados satisfatórios com marcação do tipo:

```
<img data-src="pessoa.jpg">
<noscript>
    <img src="img/small/pessoa.jpg">
</noscript>
```

Realmente, o Riloadr apresenta uma das técnicas mais eficientes para imagens responsivas ou, pelo menos, é uma das que mais oferecem opções de personalização e customizações para esse objetivo. Para ver a documentação completa e exemplos de uso, visite o repositório oficial do Riloadr (http://ow.ly/cO3HZ).

jQuery Picture

jQuery Picture é um plugin para jQuery (de somente 2KB) para adicionar suporte a imagens responsivas nos layouts. Ele suporta o elemento `<figure>` (usando alguns `data-*`) e até no elemento proposto `<picture>` (http://ow.ly/cO5lY). Um exemplo de uso seria:

```
<figure class="responsive" data-media="assets/img/small.png"
    data-media440="assets/img/medium.png"
    data-media600="assets/img/large.png">
    <noscript>
        <img src="assets/img/large.png" alt="">
    </noscript>
</figure>
```

A melhor forma enxuta de iniciar plugins com jQuery:

```
$(function(){
    $('figure.responsive').picture();
})
```

A boa notícia é que, além de ser bastante simples e fácil de se usar, ele tem suporte pelos melhores navegadores e pelo Internet Explorer a partir da versão 9. Para maiores informações, dê uma olhada no respositório oficial do jQuery Picture (http://ow.ly/cO4OW).

Picturefill

Picturefill é uma abordagem de imagens responsivas que é possível usar atualmente sem maiores complicações (em browsers com suporte a media queries, assunto que será abordado futuramente no capítulo *Media Queries*). Para tanto, ele exige uma marcação diferente da usual para imagens:

```
<div data-picture data-alt="Descrição da imagem">
    <div data-src="small.jpg"></div>

    <div data-src="medium.jpg" data-media="(min-width: 400px)">
    </div>

    <div data-src="large.jpg" data-media="(min-width: 800px)">
    </div>

    <div data-src="extralarge.jpg"
         data-media="(min-width: 1000px)">
    </div>

    <!-- Conteúdo para JavaScript desabilitado/não suportado -->
    <noscript>
      <img src="assets/img/small.jpg" alt="Descrição da imagem">
    </noscript>
</div>
```

Apesar da marcação incomum (e do trabalho de manutenção que ela acarreta), a proposta é bem interessante. Para mais detalhes a respeito, basta acessar o repositório do Picturefill (http://ow.ly/cO5En).

Adaptive Images

Adaptive Images, de Matt Wilcox (http://ow.ly/cNXkD), é uma das técnicas que, na opinião do criador (e de milhares de pessoas pelo mundo que a utilizam), mais faz sentido. Adaptive Images detecta o tamanho da tela do visitante e automaticamente cria, faz cache e entrega versões redimensionadas de imagens

referenciadas em tags `<img>` , apropriadas ao dispositivo que está sendo usado.

A técnica envolve o uso de PHP, JavaScript e um pouco de conteúdo em seu arquivo `.htaccess` (lembra-se do RESS?). Além de ser bastante eficiente por servir diferentes tamanhos da mesma imagem dependendo do dispositivo usado para a visualização do site, um outro ponto forte é que nenhuma alteração na marcação HTML se faz necessária.

Para instruções mais detalhadas e como instalar e usar a técnica, visite o site oficial da Adaptive Images (http://ow.ly/cNVks).

Qual técnica para imagens responsivas usar?

Essas foram somente algumas recomendações de técnicas para imagens responsivas possíveis. Existem, literalmente, dezenas delas e, a cada dia que passa, mais pessoas desenvolvem novas, fazem propostas de padrões de marcação e tentam desenvolver soluções eficientes para a questão de imagens responsivas para web.

Dentre tantas técnicas diferentes, fica a pergunta: qual delas usar? Como já comentado, não existe uma resposta única para essa questão; é preciso fazer uma análise do projeto em que se está envolvido para que a decisão seja tomada. Para ajudar nesta decisão, alguma considerações devem ser feitas:

- É importante que funcione sem JavaScript?
- Sendo JavaScript, pode ser um plugin ou deve ser uma solução "nativa"?
- Pode-se usar marcação especial?

- Semântica é essencial?
- As páginas precisam ser validadas no validador do W3C?
- Tem-se acesso a arquivos `server side` e `.htaccess`?
- Deve-se primar pela performance?

Respondendo a essas perguntas e tendo em mente que as pessoas esperam que o acesso a um site móvel seja tão rápido quanto no desktop (http://ow.ly/bg7Wp), é possível escolher uma solução que seja a melhor para um projeto específico que se esteja desenvolvendo no momento.

Existe uma planilha pública com uma compilação de algumas soluções possíveis para a questão das imagens responsivas. É interessante que você dê uma olhada e, quem sabe, já encontre alguma que se adeque a seu projeto atual: http://ow.ly/cO758.

Nas palavras de Ethan Marcotte

Para fechar com chave de ouro essa questão importantíssima sobre imagens responsivas, cabe a você que leia (pesquise e reflita) sobre as palavras do próprio Ethan Marcotte em uma entrevista à .NET Magazine (http://ow.ly/cSCTR). Quando o entrevistador Oliver Lindberg disse: "*Eu quero saber qual a abordagem mais prática e robusta para exibição de imagens responsivas em vários dispositivos*", sua resposta foi:

"Bem, há muita atividade em toda a área de imagens responsivas agora. Detecção de largura de banda é um ponto cego para todos nós se estamos trabalhando no front-end ou back-end. `navigator.connection` *não está amplamente implementado ou*

(eu diria) é específico o suficiente para ser útil. E todos nós estamos assistindo a Device APIs Working Group (http://ow.ly/cSDta) com interesse.

Dito isto, eu estou realmente animado com a discussão [do elemento] "picture", que tem sido habilmente conduzida por Mat Marquis (http://ow.ly/cSEky) no W3C community group for responsive images (http://ow.ly/cSEc0). E a solução 'Picturefill', de Scott Jehl, significa que podemos começar a experimentar isso hoje.

Claro que ainda não temos sequer uma especificação para um novo padrão de marcação, muito menos uma solução nativa funcional. Então, talvez a curto prazo, precisemos confiar em negociação de conteúdo server-side — embora, se esse for o caso, espero que haja uma mentalidade mais "mobile first", à Brett Jankord's Categorizr (http://ow.ly/cSFcA).

Mas eu pergunto se há outra opção disponível para nós: por que não podemos perguntar aos nossos usuários o que eles preferem? Eles estão acostumados a tomar decisões "cosméticas" de UI sempre que visitam o Gmail; ou a escolher a qualidade de vídeo no YouTube. Cada vez que clicam em um link "móvel" ou "desktop" em seus telefones, eles estão optando por experiências e conjuntos de recursos que são radicalmente diferentes um do outro. Então, eu estou querendo saber se há uma solução mais "matizada" aqui. Poderíamos pedir aos nossos usuários para, bem, dizer-nos que tipo de decisões de largura de banda tomar?"

O futuro

A maioria das soluções apresentadas anteriormente possuem algum ponto falho; afinal, assim como o próprio web design

responsivo, as técnicas que lhe são inerentes ainda estão no início, não estão "maduras" e ainda não se chegou a uma solução "perfeita" ou "infalível". A boa notícia é que já existem propostas no W3C para tentar preencher essa lacuna, com o atributo `srcset` no HTML (http://ow.ly/eqwpe) e a propriedade `image-set` para CSS (http://ow.ly/eqwjs).

Espere só mais algum tempo e usar imagens em web design responsivo será tão simples quanto escrever algumas linhas de HTML ou CSS!

4.5 IMAGENS EM ALTA RESOLUÇÃO

Devido à tecnologia atual, somente alguns dispositivos possuem displays de alta definição, capazes de apresentar corretamente imagens de altíssima resolução. No capítulo com referências de recursos para web design responsivo (capítulo *Continuando seus estudos*), você encontrará soluções para servi-las para dispositivos que possuem *retina display*; o que é bem interessante. Mas é preciso ter em mente que a tendência é que mais aparelhos comecem a apresentar uma tela com essa característica.

Portanto, é importante ter conhecimentos de alguns pontos cruciais para aqueles projetos em que imagens de alta resolução estão contempladas pelo planejamento. Adam Bradley apresenta diretrizes de muito interesse sobre a questão (http://ow.ly/e0tkE) que, independentemente de qual solução opte-se por usar, devem ser levadas em consideração:

- **Não fazer requisição da mesma imagens várias**

vezes: quando o navegador vê pela primeira vez um elemento `img` no DOM, ele imediatamente faz uma requisição ao servidor para esta imagem (mesmo antes de o DOM estar pronto). O atributo `src` deve ser alterado antes mesmo dele passar pelo elemento de imagem, ou já será tarde e a requisição já terá sido feita. Além disso, se vários pedidos estavam sendo feitos para uma imagem, isso não só retarda o navegador e impacta na performance, mas também aumenta o "estresse" adicional sobre os servidores.

- **Não usar cookie nem o elemento `<base>` :** não se deve usar por causa do *look-ahead parser* dos navegadores que já disparam as requisições das `<img>` antes mesmo de o JavaScript entrar em ação. Ou seja, o download da imagem começa antes de o código que especifica o `cookie` ou `base` rodar.
- **Manipulação mínima do DOM:** manipular o DOM requer tempo e retarda a atuação do navegador. Portanto, a solução usada deve fazer tão poucas atualizações nele quanto possível, para garantir que todos os dispositivos, especialmente os de baixa capacidade de processamento, possam lidar com essa manipulação.
- **Imagens devem continuar visíveis sem JavaScript:** se o JavaScript não estiver habilitado, o navegador ainda deve ser capaz de interpretar e renderizar todas as imagens da página em seu tamanho padrão.
- **Não apresentar imagens em alta resolução para todos:** a maioria dos usuários não têm displays de alta resolução; isso é fato. Ter servidores servindo imagens

maiores para todos os usuários adiciona uma "pressão extra" nos servidores, clientes, largura de banda etc. Só apresente imagens de alta resolução para aqueles que realmente podem valer-se desse recurso.

Como dito, essas são somente diretrizes, não regras rígidas e imutáveis de procedimentos. Então, sempre que for possível, levando em consideração a parte técnica e de planejamento, seguir essas dicas trará benefícios aos projetos que se valem de imagens de alta qualidade.

4.6 TIPOS DE IMAGEM PARA WEB

Foram (e continuam sendo) desenvolvidas técnicas que visam sanar o problema de apresentar uma imagem que melhor condiz ao device que a pessoa usa ao visitar um site. Porém, um passo anterior a isso é saber escolher qual **tipo de imagem** usar em qual situação.

Atualmente, são três principais formatos de imagem usados na web — GIF, JPG e PNG. Conhecer as diferentes características desses formatos mais comuns para poder optar conscientemente entre um ou outro, conforme a necessidade, é o primeiro passo rumo à otimização de um site — não somente os de web design responsivo!

GIF

GIF quer dizer *Graphics Interchange Format* (formato de intercâmbio de gráficos). Imagens nesse formato são conhecidas como *indexed*, dada a recomendação de se usar até 256 cores (8

bits) *indexadas* em uma paleta — embora, tecnicamente, seja possível mais (http://ow.ly/yyrhS).

Elas provêm imagens de pequeno peso, bastante usadas e ideais para o ambiente online. São utilizadas, principalmente, para ícones, ilustrações, logos, imagens "chapadas" e qualquer outra necessidade que possa ser suprida com essa limitação de cores.

GIF é um formato sem perda; o que significa que, quando você modifica e salva a imagem, não há mudança na sua qualidade. Ela suporta "transparência binária"; ou seja, pixels em uma imagem GIF, ou são totalmente transparentes, ou totalmente opacos.

Além disso, o formato passou a suportar animações depois de uma revisão poucos anos depois de seu lançamento, o que consolidou seu sucesso à época da, então novidade e certamente contribuiu, juntamente com suas outras boas características, para a "consagração" do formato e amplo uso na web.

Devido às suas características, o uso do formato GIF já foi indicado para:

- Logos;

- Ícones;
- Desenhos;
- Ilustrações simples;
- Imagens com poucas cores.

Atualmente, devido ao advento do PNG 8 (que será visto a seguir), o uso dele deve dar-se mais para imagens animadas.

JPG (ou JPEG)

JPEG é o acrônimo de *Joint Photographic Experts Group* e é um dos mais usados no mundo, não somente em ambientes virtuais, mas também por câmeras digitais e dispositivos diversos de captura de imagem. Esse índice de uso não é em vão, já que o formato de 24 bits permite o uso de até **16 milhões de cores**, o que garante imagens de boa excelente qualidade.

Apesar de o JPG poder ter milhões de cores e ainda conseguir uma excelente taxa de compressão, diferentemente do GIF, é um formato **com perda**. Ou seja, cada vez que se salva o arquivo, a compressão é refeita e um pouco de qualidade se perde.

A vantagem é que é possível escolher o nível de qualidade das imagens salvas em JPG, conforme a necessidade. A mesma imagem com uma definição excepcional que pode servir para impressão de um banner, após ser salva com qualidade inferior, também pode servir para exibição e transferência na web.

Infelizmente, o formato não aceita transparência. Devido às características mostradas, seu uso mais indicado é:

- Fotografias;
- Imagens com muitas cores;
- Imagens de alta complexidade;
- Aliar qualidade e peso do arquivo;
- Uso da mesma imagem em vários meios de acesso (ajustando a qualidade).

PNG

PNG, ou *Portable Network Graphics*, é o mais recente formato de imagem dos mais comuns usados na web. Com o objetivo de suprir determinadas restrições (técnicas e legais) do GIF, o formato trouxe consigo características desejáveis do GIF e JPG.

O formato pode vir como **PNG 8** bits ou **PNG 24** bits, cada um tendo suas características próprias e servindo para determinados propósitos. PNG-8 pega muito do GIF, sendo um formato sem perda, provendo suporte a 256 cores e transparência binária (sendo que consegue gerar arquivos de peso menor do que ele).

PNG-24 junta "o melhor dos mundos", já que é um formato sem perda de qualidade, aceita 16 milhões de cores e aceita diferentes níveis de transparência! Obviamente tudo isso acarreta em uma imagem de peso maior.

Entre PNG8 ou PNG24, suas recomendações de uso são, respectivamente, iguais a dos formatos GIF e JPG. Lembrando de que, para imagens complexas ou de muitas cores que exigem transparência, PNG24 é o ideal.

SVG

SVG é uma linguagem de marcação para descrever gráficos bidimensionais e imagens. Segundo a Wikipédia (http://ow.ly/ev4Ba):

"SVG é a abreviatura de Scalable Vector Graphics, que pode ser traduzido do inglês como gráficos vetoriais escaláveis. Trata-se de uma linguagem XML para descrever de forma vetorial desenhos e gráficos bidimensionais, quer de forma estática, quer dinâmica ou animada. Uma das principais características dos gráficos vetoriais, é que não perdem qualidade ao serem ampliados".

Preste atenção especial na parte sobre não perder qualidade ao ser ampliado! Exatamente o que é preciso ao lidar com gráficos que devem ser bem apresentados em diversas resoluções diferentes, não é verdade?

Por ser independente de resolução, SVG é ótimo para a apresentação de gráficos na web. Ela, sendo uma descrição XML para gráficos e imagens, é capaz de trabalhar bem com formas, caminhos, cores, gradientes e fontes. Filtros e animação também podem ser definidos. Uma vez criadas, imagens SVG podem ser usadas em qualquer lugar, escala e resolução.

A grande vantagem de usar SVG em web designs responsivos é que, apesar de existirem diversas técnicas para imagens flexíveis (seção *Técnicas para imagens flexíveis em web designs responsivos*), não é preciso se preocupar em usar versões de tamanhos diferentes de uma mesma imagem, uma vez que, conforme explicado, trata-se de um formato vetorial, que pode ser redimensionado à vontade sem que sua qualidade seja comprometida.

Você vai conseguir apresentar bem SVG nativamente em diversos navegadores, como:

- Chrome 4+;
- Opera 9.5+;

- Firefox 4+;
- Safari 4+;
- Internet Explorer 9+.

Icon Fonts

Icon Fonts, na verdade, não é um formato de imagem. Como sugere o próprio nome, são gráficos (geralmente, ícones) que são fontes! Trata-se de uma técnica inteligentíssima para prover "imagens" que são bem apresentadas em quaisquer resoluções sem a necessidade de ser preciso diferentes versões, em tamanhos diferentes, da mesma imagem ou gráfico.

Por exemplo, quando é preciso usar vários ícones em um site, geralmente um arquivo é feito e usado como *sprite*. Usando Icon Fonts, você incorpora em suas páginas, por meio da propriedade `@font-face` de CSS (http://ow.ly/ev7lV), fontes "especiais", em que cada caractere é, na verdade, um ícone!

E como, apesar de parecer-se com uma imagem, trata-se de uma fonte, além do conveniente de poder-se aumentar e diminuir à vontade seu tamanho sem perda de qualidade, também é possível aplicar todas as propriedades CSS relativas a fontes, o que dá uma flexibilidade incrível na hora da apresentação e efeitos!

Para se ter uma noção, dê uma olhada no *Icon Fonts are Awesome* (http://ow.ly/ev7Sj), que é um demo de Icon Font quer permite alterar, em tempo real, algumas propriedades CSS para testes, tais como tamanho; cor e posisionamento; e formato de sombra.

Projetos de renome, como Twitter Bootstrap, valem-se dessa

técnica para conferir mais dinamismo e modernidade em seus recursos. Provavelmente, você deveria fazer o mesmo! Aqui vão algumas referências:

- **Glyphicons:** excelentes *sets* de Icon Fonts gratuitos (http://ow.ly/ev9us);
- **Font Squirrel @font-face Generator:** tenha certeza de que suas fontes serão incorporadas e visualizadas corretamente em qualquer navegador (http://ow.ly/ev9mX);
- **Free Icon Fonts for Web User Interfaces:** coletânea incrível de vários sites que disponibilizam Icon Fonts (http://ow.ly/ev9Jb).

Breves considerações sobre formatos de imagem

Em um primeiro momento, pode parecer um tanto quanto inútil abordar os formatos de imagens mais comuns usados na web em um livro sobre web design responsivo. Entretanto, pensando melhor, é muito importante saber escolher conscientemente qual é o melhor formato para as diferentes necessidades dos projetos.

Usando imagens adequadas e otimizadas para cada situação (por exemplo, JPG para fotografias, PNG para ícones etc.), é possível tirar melhor proveito das características técnicas das imagens. E ao não usar formatos inadequados, muitos recursos — em diversos sentidos da palavra — são poupados.

Você deveria passar também as figuras do site em otimizadores de imagens (que diminui seu peso, sem alterar sua qualidade), já que, com menos KB para ser baixado no momento do acesso, o tempo total de carregamento é menor. Existem várias boas

ferramentas para isso, como jpegtran (http://ow.ly/dxSyx), JPEGmini (http://ow.ly/dxSte), TinyPNG (http://ow.ly/dxSpB), OptiPNG (http://ow.ly/dxSmr), e Smush.it (http://ow.ly/dxSjc).

Outra dica importantíssima é o uso de sprites (http://ow.ly/dxRdy), uma técnica para combinação de imagens menores em uma maior. Para mostrar a imagem correta, é preciso ajustar os valores da propriedade CSS `background-position`. Combinando várias desta forma, é possível reduzir os pedidos HTTP, e isso é um importantíssimo passo rumo à melhora de performance e desempenho de um site.

É possível fazer sprites manualmente, mas há ferramentas online para isso, como CSS Sprite Generator (http://ow.ly/dxRt7). Ele dá a opção de fazer upload de imagens para serem combinadas em um único sprite, gerando o código CSS (os valores de `background-position`) para renderizá-las.

Prover imagens otimizadas para cada contexto/necessidade é um passo importante para a otimização geral de web sites. Existem muitas outras técnicas para conseguir os melhores resultados de cada formato apresentado; estas que extrapolam o objetivo deste livro — veja uma planilha completa de técnicas com seus prós e contras: http://ow.ly/yyspZ.

Porém, certamente, elas devem ser de interesse para qualquer um que procure sempre prover melhores experiências aos visitantes de seus sites, garantindo que seus projetos sejam rápidos e superotimizados.

O futuro reserva deleitáveis possibilidades, como o formato de imagem Webp (http://ow.ly/yyrJY) e novas características da própria HTML, como o atributo **srcset** (http://ow.ly/yys02) e elemento **picture** (http://ow.ly/yys5k). O próprio HTTP também evolui, o que também muda bastante coisa (http://ow.ly/yyseQ).

Mas, enquanto todas essas maravilhas ainda não são uma realidade de fato, procuremos trabalhar e fazer o melhor possível com as ferramentas e recursos que temos *hoje*.

Capítulo 5

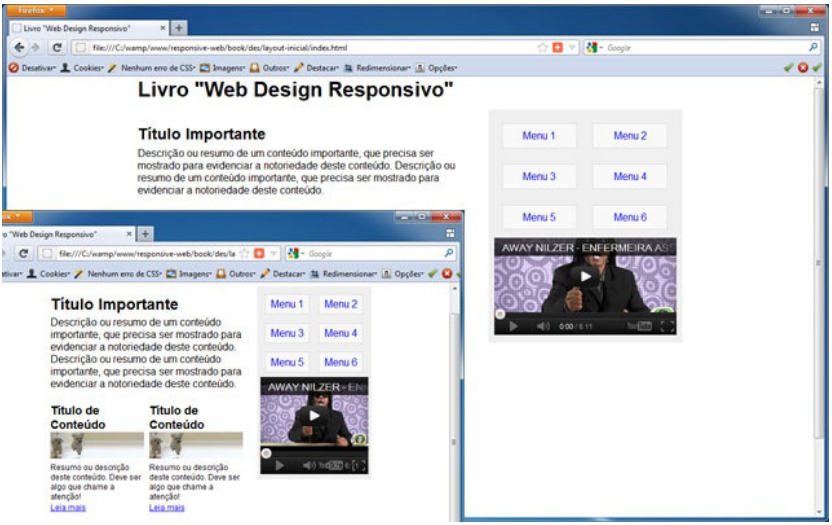
MEDIA QUERIES

No capítulo anterior, foi apresentada a técnica essencial para obter-se imagens (e outros recursos) responsivas na web.

Como foi citado no capítulo *Princípios de um web design responsivo*, quando foi abordada a trinca tecnológica do web design responsivo (seção *A trinca tecnológica do web design responsivo*), são três tecnologias necessárias para tal. Já foram abordados o layout fluido e as imagens (e outros recursos) flexíveis. E agora, finalmente, as **media queries**, uma peça-chave no quebra-cabeça da responsividade na web.

5.1 PRIMEIRO, OS MEDIA TYPES

Até o ponto em que o site-exemplo está, este já conta com layout flexível e a largura de seus elementos é ajustada conforme o tamanho da viewport do navegador muda (ou é diferente o tamanho do dispositivo que realiza o acesso).



Porém, quando o assunto é desenvolver sites de qualidade que se adaptam a todo e qualquer dispositivo, é preciso ir além. É preciso que o site não somente tenha seus elementos flexíveis, mas também que eles possam variar de posição, de tamanho e escondam-se ou apareçam, conforme a necessidade. **É preciso que o design se ajuste!**

É exatamente neste ponto que entram as *media queries*. Mas, antes de adentrarmos em seu fascinante estudo, é interessante entender um pouco sobre **media types**.

Media types é uma recomendação da W3C (http://ow.ly/cQ3MF) desde o CSS2. Com elas, é possível apresentar o site de maneira diferente, dependendo da mídia (*media*). É possível uma apresentação diferenciada (por meio de folhas de estilo) quando a página está sendo vista de um projetor, que pode ser diferentes de quando se usa uma impressora, um

sintetizador de voz, uma TV, dentre outros. Veja a lista resumidamente:

- **all:** a folha de estilo serve para todos os dipositivos;
- **braille:** para dar *feedback* quando se usa algum dispositivo tátil;
- **embossed:** impressoras em braille paginadas;
- **handheld:** dispositivos móveis (comumente com tela pequena e largura de banda limitada);
- **print:** para material paginado e para documentos visualizados na tela no modo *Visualização de impressão*;
- **projection:** destinado a apresentações projetadas como, por exemplo, projetores;
- **screen:** destinado, principalmente, para telas coloridas de computador;
- **speech:** para sintetizadores de voz;
- **tty:** dipositivos de grade fixa para exibição de caracteres, com o teletypes, terminais e alguns outros;
- **tv:** aparelho tipo TV (baixa resolução, cores, *scroll* limitado e som).

A ideia é que, ao ser acessada, a página web seja renderizada conforme o estilo mais apropriado ao dispositivo que está sendo usado para isso. Ou seja, se a pessoa está usando um *handheld*, o estilo especificado para este dispositivo, em específico, entra em ação. Caso seja através de uma TV, o estilo apropriado à TVs é usado, e assim por diante.

Veja um exemplo de implementação de media types na `<head>` de uma página (a especificação da `media` deve ser em

minúsculo):

```
<link rel="stylesheet" type="text/css" media="print"
                                       href="print_style.css">
```

Quer dizer, caso a tag `link` tenha o atributo de media type especificando `print`, somente quando alguém quiser imprimir o(s) documento(s) que leva(m) essa tag é que `print_style.css` entrará em ação; para os demais tipos de mídia, não.

Outro exemplo seria caso se quisesse usar a mesma folha de estilo para diferentes medias:

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

Neste caso, a(s) página(s) somente seria(m) renderizada(s) valendo-se de `print_handheld_style.css`, se o dispositivo fosse um *handheld*, ou no caso de a pessoa querer imprimi-la(s).

Também é possível, dentro de um mesmo arquivo de estilos, especificar regras para diferentes medias assim. E, obviamente, as regras que não ficam dentro de algum bloco `@media` são aplicadas a todos os dispositivos.

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

@media screen {
    body {
        font-size: 13px;
    }
}

@media screen, print {
    body {
```

```
        line-height: 1.2;
    }
}
```

Porém, com a evolução das tecnologias de desenvolvimento web, as diferenças entre os tipos de devices, antes tão discrepantes, começaram a desaparecer. Como escreveu Diego Eis em seu artigo sobre media queries (http://ow.ly/cQ85G):

"Cada vez mais surgem dispositivos de diversos tamanhos com hardwares bem parecidos com os desktops. Isso faz com que a navegação destes aparelhos tenha uma experiência muito próxima de desktop. Um exemplo atual é o iPhone. Sua tela tem boa qualidade e seu navegador renderiza as páginas como um navegador normal de desktop. Logo, não tem motivo para prepararmos um layout e um CSS com media type `handheld` *para o iPhone. Apesar de ele ser um* `handheld` *, ele não trabalha como um. Contudo, ele também não trabalha como um desktop. Mesmo a renderização do MobileSafari sendo dêntica a de um desktop, o comportamento do usuário e a forma de navegação são diferentes. Logo, temos um meio termo. Não podemos disponibilizar um CSS para* `HANDHELD` *, nem um CSS totalmente* `SCREEN` *".*

A solução? **Media queries**.

5.2 MEDIA QUERIES

Quer dizer, começou a não ficar tão clara a diferença entre os vários tipos de dispositivos e suas capacidades, deixando os media types em uma posição de não atenderem mais às necessidades de desenvolvimento sozinhos, já que muitos acabaram ficando em uma zona cinzenta de medias, tendo características tanto de

handheld quanto de desktop, como o citado iPhone.

A solução? Media queries. Como muito bem argumentou Sérgio Lopes em seu artigo sobre media queries (http://ow.ly/cQ8Di):

"Melhor do que separar os dispositivos em desktop (`screen`) e mobile (`handheld`), os novos media queries permitem que foquemos principalmente no tamanho da tela onde vamos exibir o conteúdo".

Ou seja, media queries — que já é uma recomendação do W3C desde junho de 2012 (http://ow.ly/cUjIw) — tornaram possível a evolução para um novo nível de especificação de estilos para web sites. Por exemplo, é possível indicar um CSS somente para dispositivos com tela de até `320px` :

```
<link rel="stylesheet" media="screen and (max-width: 320px)"
                       href="320.css">
```

Caso se queira usar várias declarações em somente um arquivo, faria-se assim:

```
@media screen and (min-width: 320px) {
    body {
        font-size: 80%;
    }
}

@media screen and (min-width: 480px) {
    body {
        font-size: 90%;
    }
}
```

Porém, não somente isso. Existe uma gama extensa de parâmetros de media queries.

5.3 PARÂMETROS PARA TRABALHAR COM MEDIA QUERIES

Seguindo alguma das sintaxes apresentadas anteriormente, é possível especificar o uso de estilos separados por arquivo ou blocos dentro do mesmo arquivo, usando um ou vários dos parâmetros a seguir. Importante ressaltar que sua maioria pode ser prefixado com `min-` ou `max-` , que representam, respectivamente, "maior ou igual a" e "menor que ou igual a" (assim como foi mostrado no exemplo anterior).

Veja alguns desses parâmetros a seguir.

aspect-ratio

Ele descreve a proporção da área de exibição do navegador usado. O valor é composto de dois números inteiros positivos, separados por um caractere de barra (`/`). Isso representa a proporção de pixels na horizontal (primeiro termo) para pixels na vertical (segundo termo).

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, para aplicar regras com a janela do navegador em widescreen de proporção `16:9` :

```
@media screen and (aspect-ratio: 16/9) {
    [...]
}
```

color

Ele indica o número de bits por componente de cor do dispositivo. Se ele não é um dispositivo de cor, o valor é igual a 0. Se os componentes de cor têm diferentes números de bits, o menor número é usado.

Por exemplo, se um display utiliza 5 bits para azul e vermelho, e 6 bits para verde, ele usará 5 bits. Se o dispositivo usa cores indexadas, o número mínimo de bits por componente de cor na tabela de cores é usado.

- **Valor:** ;
- **Media:** visual;
- **Aceita min/max:** sim.

Por exemplo, para especificar regras para dispositivos coloridos, seria:

```
@media all and (color) {
    [...]
}
```

Ou, para estilos que somente serão aplicados em dispositivos com, pelo menos, 4 bits de cor:

```
@media all and (min-color: 4) {
    [...]
}
```

Ou, ainda, com no máximo 16 bits de cores:

```
@media (max-color: 16) {
    [...]
}
```

color-index

Ele descreve o número de entradas na tabela de cores do

dispositivo. Se ele não possuir tabela de cores, então o valor é 0.

- **Valor:** ;
- **Media:** visual;
- **Aceita min/max:** sim.

Por exemplo, para especificar que algumas regras serão aplicadas a todos os dispositivos com cores indexadas:

```
@media all and (color-index) {
    [...]
}
```

Dispositivos com, pelo menos, 2 cores indexadas:

```
@media (min-color-index: 2) {
    [...]
}
```

Ou incorporar uma folha de estilos para, somente, dispositivos com menos de 256 cores indexadas:

```
<link rel="stylesheet" media="all and (max-color-index: 256)"
href="http://foo.bar.com/stylesheet.css">
```

device-aspect-ratio

Ele descreve a proporção do display do dispositivo. O valor é composto de dois números inteiros positivos, separados por um caractere de barra (`/`). Isso representa a proporção de pixels na horizontal (primeiro termo) para pixels na vertical (segundo termo).

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, para aplicar regras em dispositivos widescreen:

```
@media screen and (device-aspect-ratio: 16/9),
screen and (device-aspect-ratio: 16/10) {
    [...]
}
```

device-height

Ele descreve a altura do display do aparelho em pixels. Para meios contínuos, é a altura da tela; para meios paginados, é a altura do tamanho da folha de cada página.

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, aplicar uma folha de estilo a um documento quando visualizado em uma tela menor que 480px de altura:

```
<link rel="stylesheet"
      media="screen and (max-device-height: 379px)"
      href="style.css">
```

device-width

Ele descreve a largura do display do aparelho em pixels. Para meios contínuos, é a largura da tela; para meios paginados, é a largura do tamanho da folha de cada página. Obviamente, comprimento não pode ser um valor negativo.

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, aplicar uma folha de estilo a um documento quando visualizado em uma tela menor que 800px de largura:

```
<link rel="stylesheet"
      media="screen and (max-device-width: 799px)"
      href="style.css">
```

grid

Ele determina se trata-se de um dispositivo de grade ou um de mapa de bits (bitmap). Se ele é baseado em grade (tal como um terminal TTY ou um visor do telefone, com apenas uma fonte), o valor é 1; caso contrário, é 0.

- **Valor:** ;
- **Media:** todas;
- **Aceita min/max:** não.

Por exemplo, para aplicar algumas regras a todos dispositivos assim:

```
@media (grid) {
    [...]
}
```

Para aplicar um estilo em dispositivos móveis com display de 15 caracteres ou menos:

```
@media handheld and (grid) and (max-width: 15em) {
    [...]
}
```

Note que a unidade `em` tem um significado especial para dispositivos assim. Já que a largura exata de um `em` não pode ser determinada, assume-se que 1em é a largura de uma célula da grade horizontal e a altura de uma célula vertical.

height

Altura, em pixels, da janela do navegador sendo usado. Para meios contínuos, é a altura da viewport, incluindo o tamanho da barra de rolagem; para meios paginados, é a largura da "page box" da impressora. O comprimento não pode ser um valor negativo.

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, especificar algumas regras CSS para dispositivos com a altura da janela do browser em 300px:

```
@media screen and @media (height: 300px) {
    [...]
}
```

Ou com a altura menor que 600px:

```
@media screen and @media (max-height: 599px) {
    [...]
}
```

monochrome

Ele indica o número de bits por pixel em dispositivos monocromáticos (escala de cinza). Se ele não for monocromático, o valor é zero.

- **Valor:** ;
- **Media:** visual;
- **Aceita min/max:** sim.

Por exemplo, para aplicar estilos em qualquer dispositivo

monocromático:

```
@media all and (monochrome) {
    [...]
}
```

Para aplicar estilos em qualquer dispositivo monocromático com, pelo menos, 2 bits por pixel:

```
@media all and (min-monochrome: 2) {
    [...]
}
```

orientation

Ele indica se o dispositivo está em modo "paisagem" (*landscape*), quando a tela mais larga que alta; ou "retrato" (*portrait*), quando a tela é mais alta que larga.

- **Valor:** landscape | portrait;
- **Media:** visual;
- **Aceita min/max:** não.

Então, para aplicar-se regras para dispositivos que estão no modo paisagem, basta usar:

```
@media all and (orientation:landscape) {
    [...]
}
```

resolution

Ele trata sobre a resolução (densidade de pixels) do dispositivo, que pode ser especificada em pontos por polegada (dpi) ou por centímetro (dpcm).

- **Valor:** ;

- **Media:** bitmap;
- **Aceita min/max:** sim.

Por exemplo:

```
@media (resolution: 72dpi) {
    [...]
}

@media (min-resolution: 118dpcm) {
    [...]
}
```

Uma `<resolução>` sem algum dos prefixos `min-` ou `max-` nunca vai atuar em dispositivos com pixels não-quadrados (*non-square pixels*).

scan

Ele trata sobre o processo de escaneamento que um dispositivo do tipo TV pode fazer.

- **Valor:** progressive | interlace;
- **Media:** tv;
- **Aceita min/max:** no.

Por exemplo, para aplicar uma folha de estilo apenas para televisões com escaneamento progressivo:

```
@media tv and (scan: progressive) {
    [...]
}
```

width

Largura, em pixels, da janela do navegador sendo usado. Para

meios contínuos, é a largura da viewport, incluindo o tamanho da barra de rolagem; para meios paginados, é a da "page box" da impressora. O comprimento não pode ser um valor negativo.

- **Valor:** ;
- **Media:** visual, tátil;
- **Aceita min/max:** sim.

Por exemplo, especificar algumas regras CSS para dispositivos com a largura da janela do navegador em 640px:

```
@media screen and @media (width: 640px) {
    [...]
}
```

Ou com a largura menor que 768px:

```
@media screen and @media (max-width: 767px) {
    [...]
}
```

5.4 OPERADORES LÓGICOS

Apesar de alguns dos **operadores lógicos** de media queries já terem sidos mostrados em exemplos anteriores, para ficar claro sobre sua existência, possibilidades e utilidade, conforme o W3C (http://ow.ly/cQVfk), os Operadores Lógicos de media queries são:

- `and` e `or` ;
- `not` ;
- `only` .

and e or

Várias media queries podem ser combinadas em uma lista

(separada por vírgulas) de consultas. Se uma ou mais das consultas são verdadeiras, toda a lista é verdadeira; do contrário, é falsa. Na sintaxe de media queries, a palavra `and` expressa o operador lógico **AND** e a `vírgula` expressa o operador lógico **OR**.

Por exemplo:

```
/* Telas e impressões coloridas */
@media screen and (color), print and (color) {
    [...]
}
```

not

O operador lógico **NOT** é indicado pela palavra `not`. A presença da palavra `not` no início da consulta nega o resultado; ou seja, se a consulta de mídia for verdadeira, sem o `not` ela torna-se falsa (e vice-versa).

Os navegadores que suportam somente media types não reconhecerão essa palavra-chave e a folha de estilo associada, portanto, não será aplicada. Exemplo:

```
<link rel="stylesheet" media="not screen and (color)"
                       href="style.css">
```

only

A palavra-chave `only` pode ser usada para esconder as folhas de estilo de agentes de usuário mais antigos. Eles devem começar a processar as consultas de mídia com `only` como se a palavra-chave `only` não estivesse presente.

```
<link rel="stylesheet" media="only screen and (color)"
                       href="example.css">
```

5.5 VÁ QUEBRANDO SEU DESIGN EM BREAKPOINTS BEM PENSADOS

Apesar de existir mais de uma dezena de parâmetros para media queries, a verdade é que somente alguns poucos são usados. Isso ocorre não pela falta de cuidado ou competência no desenvolvimento, mas principalmente devido ao fato de que as necessidades atuais da maioria dos sites não requerem o uso de muitos dos parâmetros. De fato, a maioria dos usados atualmente dizem respeito à largura e altura do display e à orientação do dispositivo.

Como foi visto anteriormente, é possível declarar regras CSS específicas conforme o tamanho do dispositivo ou browser, por meio dos parâmetros `device-width` e `width` (e seus prefixos). Isso traz uma discussão importante sobre quais são os **breakpoints** (ou "pontos de interrupção") mais eficientes. Quer dizer, os pontos em que se devem tratar o web design de forma diferenciada para que a experiência de quem está visitando o site seja a melhor e mais eficiente possível.

Qual é o momento de parar de escrever regras CSS para uma resolução e começar para outra? Se estamos escrevendo estilos que atendem a diversos tipos de devices, em qual momento, especificamente, é o "limite da resolução" de um dispositivo e início de outro? Em que momento é preciso que uma sidebar seja posicionada abaixo do conteúdo principal? Quando a largura do dispositivo obrigado ao usuário *scrollar*, por exemplo. Isso definiria um breakpoint!

Então, para ficar claro, breakpoints são os delimitadores das

regras de CSS para atenderem às diferentes especificações (feitas por você mesmo, ao usar os parâmetros de media queries). Ou seja, os pontos em que devemos utilizar estilos diferentes para tratar o web design de forma diferenciada, fazendo a experiência de quem está visitando o site seja a mais proveitosa possível.

Diferentes abordagens sobre breakpoints

Quando os estudos de web design responsivo estavam bem no início, alguns desenvolvedores propuseram o uso de breakpoints conforme a resolução que determinados dispositivos tinham. Por exemplo, `@media only screen and (min-device-width : 320px) and (max-device-width : 480px)` para smartphones; `@media only screen and (min-width : 1224px)` para notebooks e desktops, e assim por diante.

Mal isso começou e já foi constatado que essa abordagem não funciona muito bem, já que, conforme sua evolução, é praticamente impossível determinar qual device estamos usando, conforme a resolução no momento da visita ao site.

Portanto, não existe consenso sobre quais seriam os melhores ou mais eficientes breakpoints em web design responsivo. Para dizer a verdade, existe bastante discrepância de opiniões; alguns chegando a afirmar que não seria preciso a especificação de breakpoints, mas, sim, o desenvolvimento de layouts que se adaptariam a quaisquer resoluções.

Para os que pensam de uma maneira diferentes, a preferência é valer-se desse recurso para alterar a disposição do layout conforme o dispositivo (ou largura do browser) que se esteja usando. Para esses casos, existem posições diferentes sobre quais seriam os

melhores (ou preferencialmente usados).

Realmente, não é possível especificar um conjunto de breakpoints que sejam os mais eficientes. Cada projeto web tem suas próprias peculiaridades e pede uma implementação diferenciada dos demais. Alguns tentaram chegar a um ponto de partida genérico, especificando breakpoints-chave, que podem facilitar o início de qualquer projeto. Veja alguns dos mais conhecidos a seguir.

320 and up

Perceba que o uso de media queries dá-se de forma crescente. Ou seja, primeiro contemplando estilos para dipositivos com resoluções menores com `480px` e `600px` (geralmente, smartphones e netbooks), para depois abordar os de maior resolução, como os tradicionais PCs e dispositivos com alta resolução (por exemplo, as Smart TVs mais atuais). Para saber mais, acesse: http://ow.ly/cSIXX.

```
@media print { }

@media only screen and (min-width: 480px) { ... }

@media only screen and (min-width: 600px) { ... }

@media only screen and (min-width: 768px) { ... }

@media only screen and (min-width: 992px) { ... }

@media only screen and (min-width: 1382px) { ... }

@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
    only screen and (min--moz-device-pixel-ratio: 1.5),
    only screen and (min-device-pixel-ratio: 1.5)
{ ... }
```

Less framework

Os próprios comentários do *Less framework* já ajudam bastante a identificar para quais tipos de navegação as folhas de estilo foram projetadas. Para saber mais, acesse: http://ow.ly/cSQIb.

```
/* Tablet Layout */
@media only screen and (min-width: 768px)
                   and (max-width: 991px) { ... }

/* Mobile Layout */
@media only screen and (max-width: 767px) { ... }

/* Layout largo de mobile */
@media only screen and (min-width: 480px)
                   and (max-width: 767px) { ... }

/* Retina display */
@media
only screen and (-webkit-min-device-pixel-ratio: 2),
only screen and (min-device-pixel-ratio: 2) { ... }
```

Os desenvolvedores definiram os breakpoints no que chamaram de `Tablet Layout`, `Mobile Layout`, `Layout largo de mobile` e `Retina display` (regras estas, obviamente, somente para os poucos dispositivos que temos hoje que possuem essa característica).

Skeleton

O *Skeleton* é bem específico ao propor breakpoints para dispositivos levando em consideração, inclusive, a orientação usada no momento (landscape ou portrait). Quer dizer, um planejamento da experiência do usuário mais detalhado e bem feito que, se bem usado, tem maiores chances de promover uma boa experiência aos visitantes e garantir uma boa dose de "agradabilidade". Para saber mais, acesse: http://ow.ly/cSQ78.

```
/* Menor que 960 */
@media only screen and (max-width: 959px) { ... }

/* Tablet Portrait ao padrão 960 */
@media only screen and (min-width: 768px)
                   and (max-width: 959px) { ... }

/* Todos tamanhos de mobile */
@media only screen and (max-width: 767px) { ... }

/* Mobile em landscape a tablet Portrait */
@media only screen and (min-width: 480px)
                   and (max-width: 767px) { ... }

/* Mobile em portrait a mobile em landscape */
@media only screen and (max-width: 479px) { ... }
```

Twitter Bootstrap

O *Twitter bootstrap* também teve o cuidado de projetar para vários dispositivos usando as variações de orientação. Apesar de conter menos breakpoints que o Skeleton, dá conta do recado tranquilamente (lembrando de que isso é o que vem por padrão e, seja lá em qual deles você esteja se inspirando, pode acrescentar ou retirar quantos breakpoints quiser). Para saber mais, acesse: http://ow.ly/cSLyw.

```
/* 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) { ... }
```

5.6 GERENCIAMENTO DE ERROS

Foram vistos quais os tipos possíveis de parâmetros para media queries e como é possível, por meio de operadores lógicos, refinar consultas a mídias específicas. Mas, e quando as coisas dão errado? E se, por acaso, ocorrer algum erro de sintaxe ou algum parâmetro inexistente for especificado?

Tipos de mídia desconhecidos serão avaliados como `false`. Efetivamente, eles são tratados de forma idêntica para os tipos conhecidos de mídia que não correspondem ao tipo de mídia do dispositivo. Por exemplo, a consulta de mídia `unknown` vai ser avaliada como `false`, a não ser que `unknown` seja um tipo de mídia suportada.

Parâmetros de media queries desconhecidos

Os agentes de usuário representarão uma consulta de mídia como `not all` quando algum dos seus parâmetros não for conhecido. Por exemplo:

```
<link rel="stylesheet" media="screen and (max-weight: 3kg)
                                    and (color), (color)"
href="style.css">
```

A primeira consulta será representada como `not all` e avaliada como falsa; a segunda, como se a primeira não tivesse sido especificada.

Veja um outro exemplo que resultará em `not all` e tente compreender o motivo:

```
@media (min-orientation:portrait) {
    [...]
}
```

Adivinhou? O parâmetro `orientation` não admite prefixo `min-`!

Valores desconhecidos em parâmetros de media queries

Tal como acontece com parâmetros desconhecidos, agentes de usuário interpretam como `not all` quando um dos *valores* de algum parâmetro especificado não for conhecido. Por exemplo, essas duas consultas seriam interpretadas como `not all`:

```
@media (color: 20example) {
    [...]
}

@media (min-width: -100px) {
    [...]
}
```

Media queries mal formadas (erros de sintaxe)

Os agentes de usuário conseguem lidar com símbolos inesperados encontrados ao analisar uma consulta de mídia. Eles leem até o final, observando correspondência de pares de `()`, `[]`, `{}`, `""` e `''`, e lidando corretamente com escapes. Consultas de mídia com *tokens* inesperados são interpretadas como `not all`.

```
/* Aplicado somente a dispositivos "speech" */
@media (example, all,), speech {
    [...]
}

/* Aplicado somente a dispositivos "screen" */
@media &test, screen {
    [...]
}
```

```
/* Sem espaço entre "and" e "(" */
@media all and(color) {
    [...]
}

/* Ponto e vírgula?! */
@media test;,all {
    [...]
}
```

5.7 USO CONSCIENTE DE MEDIA QUERIES

Muita coisa foi vista sobre media queries. Evidentemente, não se espera que, em uma primeira lida de todo esse conteúdo, tudo seja entendido e absorvido. Na verdade, a melhor maneira para aprender e fixar tudo isso é colocando alguns projetos em prática!

Mas, mesmo antes de começar a mexer com as fantásticas media queries, é preciso levar em consideração algumas questões muito importantes sobre seu uso, tais como: se usar arquivos externos ou não; a melhor maneira de usar breakpoints; se é melhor começar pelos tamanhos menores ou maiores etc.

Há algum conteúdo sobre isso circulando na web e, felizmente, é possível encontrar algumas considerações importantes no blog do zomig (http://ow.ly/cTV4J). Vamos ver algumas delas.

Media queries em arquivos separados ou em um único arquivo

Segundo foi visto anteriormente, é possível realizar consultas de mídia chamando folhas de estilo em separado, ou realizar a diferenciação de estilos usando breakpoints dentro de um mesmo arquivo:

```
<link rel="stylesheet" type="text/css"
                        media="screen and (min-width:480px)"
href="480.css">
<link rel="stylesheet" type="text/css"
                        media="screen and (min-width:768px)"
href="768.css">
```

```
@media only screen and (min-width: 480px) { ... }

@media only screen and (min-width: 768px) { ... }
```

Mas qual é a melhor maneira de usar? Chamar vários arquivos externos, um para cada breakpoint, ou somente um arquivo com todos breakpoints especificados? Na verdade, a resposta não é tão simples e há algumas considerações a esse respeito.

Quando chamamos vários arquivos diferentes, é importante ser levado em conta os browsers que não oferecem suporte a media queries, pois eles não baixarão os arquivos. Também, mesmo quando há um arquivo que não será usado para algum tipo de media, este é baixado de qualquer maneira.

Por exemplo, um iPad vai baixar o arquivo com `(max-width:700px)`. Sobre essa questão, existem prós e contras.

Prós em se usar 1 arquivo:

- Somente 1 requisição HTTP;
- Mais difícil de se esquecer de atualizar.

Contras em se usar 1 arquivo:

- Tamanho do arquivo fica maior;
- A medida que o projeto cresce, a manutenção fica mais difícil;
- Uso de soluções JavaScript para funcionar em IE8

para baixo.

Prós em se usar vários arquivos:

- Para navegadores que não suportam, o arquivo padrão é menor;
- Organização/manutenção ficam melhores.

Contras em se usar vários arquivos:

- Várias requisições HTTP;
- Mais fácil de esquecer de atualizar algo.

Como o desenvolvimento para web não é feito de tecnologias funcionando em separado e levando em conta que várias requisições HTTP acabam sendo mais lentas do que somente um arquivo de tamanho maior, a recomendação é que seja usada *somente uma folha de estilos*. Vale lembrar de que não há uma resposta única, apenas uma recomendação. Sempre dependerá muito da sua situação.

Media queries sobrepostas ou empilhadas

Apesar de não serem termos oficiais dentro do web design responsivo, "sobrepostas" e "empilhadas" traduzem bem as possibilidades e situações que podem acontecer quando estamos desenvolvendo sob a filosofia do *responsive web design*. Então, independentemente de usarmos um ou vários arquivos de estilo, o fato é que é possível, por meio dos breakpoints, **sobrepor** ou **empilhar** as regras usadas.

Veja este exemplo:

```
@media all and (min-width:500px) {
    body {
        background: blue;
        font-family: serif;
    }
}

@media all and (min-width:700px) {
    body {
        background: red;
        color: #FFF;
    }
}
```

Perceba que as especificações de media não são mutuamente exclusivas; quer dizer, se a viewport no momento do acesso for 800px, ambas as regras entrarão em ação. Se isso for feito dessa maneira, sem a devida consciência de qual efeito quer se obter, o que conseguiremos, no final, é muita dor de cabeça e horas a mais de manutenção do arquivo. Não há problema em fazer isso, desde que essa prática seja algo consciente!

Também não é por que agora estamos usando media queries no arquivo que as regras mais elementares de CSS deixam de valer. Então, a **cascata** e a **especificidade** ainda estão presentes! Supondo um acesso com viewport de 800px, em um código como esse, a página será renderizada com fundo azul em vez de vermelho:

```
@media all and (min-width:700px) {
    body {
        background: red;
        color: white;
    }
}

@media all and (min-width:500px) {
    body {
        background: blue;
        font-family: serif;
```

```
    }
}
```

Para fins de organização, manutenção e facilidade de entendimento da lógica dos estilos que compõem o projeto, geralmente é mais fácil empilhar os breakpoints e separar as regras devidamente, como em:

```
@media all and (min-width:500px) and (max-width:699px) {
    body {
        background: blue;
        font-family: serif;
    }
}

@media all and (min-width:700px) {
    body {
        background: red;
        color: white;
        font-family: serif;
    }
}
```

Veja que, para que seja aplicada uma família de fonte serifada (`font-family: serif`), foi preciso repetir o par propriedade/valor, uma vez que as regras para um breakpoint específico não entram em ação para outro. Para as regras genéricas, que devem entrar em ação sempre, o recomendado é deixar fora de um declaração de media.

Prós da sobreposição:

- Tamanho de arquivo menor (não é preciso repetir regras);
- Fácil de atualizar regras compartilhada (já que são declaradas só uma vez).

Contras da sobreposição:

- Arquivos maiores, que tendem a ser mais complexos e de difícil manutenção;
- Imagens substituídas/escondidas em partes posteriores do código continuam sendo baixadas por navegadores baseados em WebKit.

Prós do empilhamento:

- Browsers somente baixam imagens condizentes ao breakpoint atual.

Contras do empilhamento:

- Arquivos maiores, já que algumas regras são repetidas em vários breakpoints;
- Em atualizações, é mais fácil de esquecer de atualizar regras repetidas em alguns breakpoints.

Sobrepor estilos pode deixar o código mais enxuto, mas sua manutenção e entendimento ficam mais complicados. Empilhar deixa o código maior, mas mais simples (embora algumas vezes repetido). Quando os estilos entre os breakpoints são bastante diferentes (site muito diferente para mobile do que para desktop, por exemplo), é aconselhado empilhar; do contrário (casos mais comuns), sobrepor.

Versões antigas do IE: comentários condicionais ou JavaScript?

Para versões do Internet Explorer anteriores à 9, não há suporte para media queries (por essa você já esperava!). Para a maioria dos projetos no Brasil, ainda é importante dar suporte

para, pelo menos, IE7+. É preciso, então, fazer isso de alguma maneira. Para tanto, é possível usar comentários condicionais ou alguma solução JavaScript.

Em relação aos comentários, no próprio blog da Microsoft é possível encontrar soluções para especificar estilos para Windows Phone (http://ow.ly/cTZ9i). Um exemplo seria:

```
<link rel="stylesheet" media="all" href="global.css">
<link rel="stylesheet" media="all and (max-width: 700px)"
                       href="mobile.css">
<!--[if IEMobile 7]>
    <link rel="stylesheet" media="all" href="mobile.css">
<![endif]-->
```

Ou:

```
<link rel="stylesheet" media="all" href="global.css">
<link rel="stylesheet" media="all and (min-width: 700px)"
                       href="desktop.css">

<!--[if (lt IE 9)&(!IEMobile 7)]>
    <link rel="stylesheet" media="all" href="desktop.css">
<![endif]-->
```

Quanto a soluções JavaScript, existem muitas, mas algumas das mais conhecidas e usadas são:

- `Respond` : suporte somente os parâmetros `min-width` e `max-width` , mas é bastante eficiente no que se propõe.
- `css3-mediaqueries.js` : é mais pesado e mais lento do que o outro, mas suporta mais parâmetros de media queries.

Não se esqueça que, para que os arquivos não sejam executados também em navegadores de qualidade, é preciso condicionar sua

chamada:

```
<!--[if (lt IE 9)&(!IEMobile 7)]>
    <script src="assets/js/respond.min.js"></script>
<![endif]-->
```

Prós de comentários condicionais:

- Funciona sem JavaScript;
- Todos os prós de media queries em vários arquivos.

Contras de comentários condicionais:

- Usa somente 1 estilo para IEs antigos;
- Todos os contras de usar media queries em vários arquivos.

Prós de soluções JavaScript:

- Vários breakpoints para IE (inclusive mudando em tempo real);
- Todos os prós em se usar 1 só arquivo para media queries.

Contras de soluções JavaScript:

- Não funciona quando o JavaScript está desabilitado;
- Todos os contras em se usar 1 só arquivo para media queries.

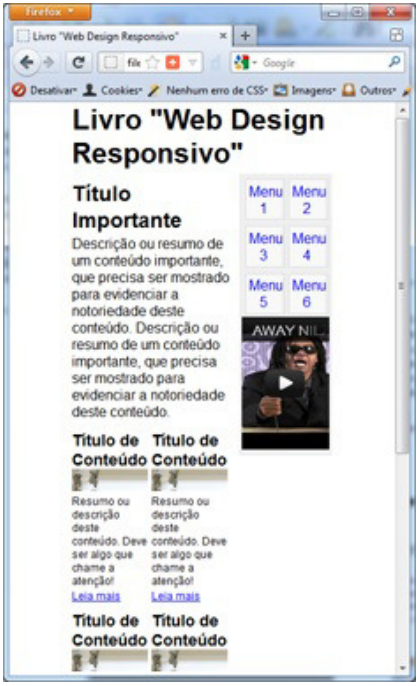
No Brasil, infelizmente, a porcentagem de uso de IE abaixo do 9 ainda é alta. Usar soluções JavaScript para o problema de não suporte a media queries em IE8 para baixo certamente torna a vida do desenvolvedor mais fácil, e a maioria usa (ou deveria usar) algum script de alerta para atualização do navegador quando se

trata de uma visita a partir dos antigos Internet Explorers! Conheça bem seu público para saber quais navegadores são usados.

5.8 MEDIA QUERIES NA PRÁTICA

É hora da aplicação prática de alguns dos conceitos de media queries vistos! Como você deve se lembrar, o site de exemplo já conta com muitas características de um site responsivo, tendo imagens (e vídeos) flexíveis e a maioria das áreas de conteúdo possuem largura variável, conforme a resolução da viewport.

Entretanto, ainda não há aplicação de media queries. Então, o site fica "espremido" quando visto em determinadas resoluções.

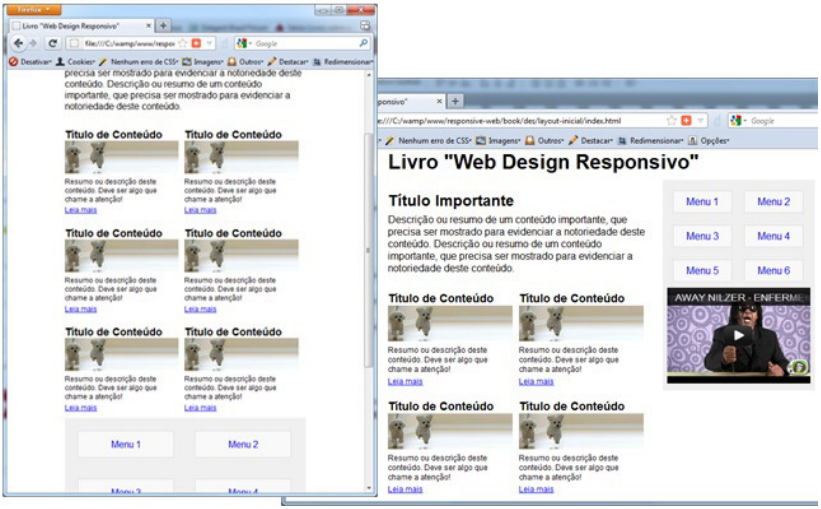


Se nada fosse feito, ainda assim seria uma solução parcialmente aceitável, já que o conteúdo ainda seria completamente acessível, independente da resolução. Entretanto, como já foi mostrado, por meio de media queries, nós podemos planejar melhor como será a experiência dos visitantes, alterando (exibindo ou ocultando) a disposição dos elementos, conforme a experiência que se queira oferecer.

Por exemplo, quando a resolução do navegador no momento da visita for de largura de até 640px, o site não será apresentado em colunas, mas, sim, com o conteúdo principal vindo em primeiro lugar (lógico) e a sidebar embaixo. Para tanto, é preciso acrescentar algumas linhas de CSS ao final do atual arquivo do site de exemplo, criando um breakpoint que contemple resoluções de até `640px`:

```
@media all and (max-width: 640px) {
    .content-main {
        float: none;
    }

    .content-sidebar {
        float: none;
    }
}
```

Com isso, sempre que a resolução do navegador do dipositivo usado for menor ou igual a 640px, a divisão de coluna principal e coluna secundária não mais existirá no site, sendo apresentado em uma só sequência.



Na verdade, o fato de o menu de navegação principal estar sendo apresentado em duas colunas pode ser um contratempo para quem acessa usando uma resolução bem pequena. É possível resolver inserindo o seguinte trecho de código antes da media querie mostrada anteriormente:

```
@media all and (max-width: 320px) {
    .main-nav li {
        float: none;
        width: auto;
    }
}
```

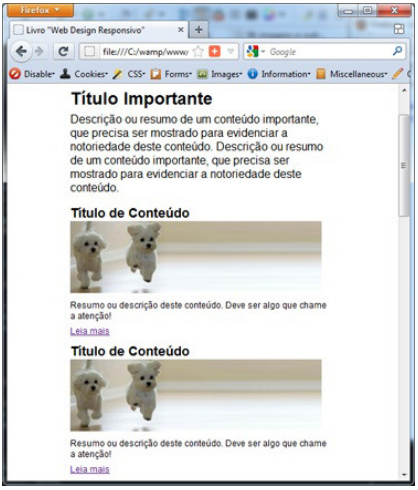
Código que garantirá a seguinte renderização:

Pensando bem, talvez o fato de as notícias serem apresentadas em duas colunas também não contribua para uma boa experiência dos visitantes. É possível alterar isso no breakpoint para resolução de até 640px, que fica assim:

```
@media all and (max-width: 640px) {
    .content-main {
        float: none;
    }

        .last-content-call {
            float: none;
```

```
            width: auto;
        }

    .content-sidebar {
        float: none;
    }
}
```

Isso resultará em:



Como você verá depois que fizer alguns projetos e praticar bastante, o mais difícil não é codificar sites responsivos. O verdadeiro desafio é fazer um bom planejamento de cada um dos sites para diferentes *ranges* de resoluções para garantir que, em cada um deles, o visitante tenha uma boa experiência de uso, independentemente de qual dispositivo esteja usando.

Será que a melhor forma de vencer esse desafio é encarar o mobile em primeiro lugar?

Capítulo 6

TÓPICOS DE WEB MOBILE

O caminho traçado por meio das técnicas para se fazer um bom **web design responsivo** foi emocionante! Foram vistas muitas das estatísticas que provam que o desenvolvimento responsivo não é uma mera tendência, como sugerem alguns.

Depois, vimos os três pilares do web design responsivo: layout fluido, imagens (e outros recursos) flexíveis e media queries. Sabendo usar essa trinca poderosa, certamente você está preparado tecnicamente para lidar com web design responsivo.

Mas, como foi alertado no fim do capítulo anterior, o desafio maior não é dominar sua parte técnica, mas é saber *como*, *quando*, *se*, *onde* e ser capaz de explicar todos os *porquês* de se usar uma ou outra técnica responsiva.

6.1 O QUE É MOBILE FIRST?

Uma abordagem importante para se ter um bom web design responsivo é **Mobile First**! Mobile First (ou *Móvel Primeiro*), criada por **Luke Wroblewski** (http://ow.ly/dPqs2), é uma metodologia de desenvolvimento web que preconiza o dever de **primeiro planejar para dispositivos móveis** e, só depois, *aumentar* os possíveis dispositivos até se chegar ao desktop (e

além). Em termos mais simples, **do menor para o maior**.

Atualmente, já temos muita diversidade de dispositivos, desde os móveis (phones, smartphones, tablets), passando pelos especializados (eReaders, TVs etc.), até os mais tradicionais (desktops, laptops, netbooks). E um dos desafios do desenvolvimento web moderno (e profissional) é dar suporte a todos eles e aos que ainda estão por vir.

É por isso que essa abordagem é tão eficiente: começar com mobile e projetar com melhoramento progressivo (*progressive enhancement*) permite abranger todos os devices. Qualquer dispositivo com acesso a web será capaz de acessar o site e ter uma experiência funcional. Em seguida, é possível permitir que essa experiência seja melhorada e otimizada para o contexto do dispositivo que está sendo utilizado, usando deteção de recurso, carregamento condicional de scripts, media queries (capítulo anterior) e outras técnicas.

Projetar, levando em conta Mobile First, requer uma revisão profunda e fundamental de uma página e, mais importante, exige uma **revisão mental**. Não se trata de uma solução rápida e milagrosa; pelo contrário, Mobile First demanda um planejamento cuidadoso, tempo e uma execução séria e com disciplina — você já deve ter imaginado depois das últimas palavras, que é algo difícil.

Pode parecer assustador no começo, mas depois que você internaliza sua filosofia, as recompensas adquiridas são enormes! Em vez de ter de criar interações totalmente novas toda vez que um novo dispositivo sai, basta otimizar a experiência para o novo contexto sem ter de reinventar a roda (e virar madrugadas) para isso.

6.2 POR QUE MOBILE FIRST?

É absolutamente comum que aconteça um profundo e quase sintomático estranhamento quando temos o primeiro contato com a abordagem Mobile First. Afinal, antes dessa história toda de web design responsivo, o jeito "normal" de se elaborar designs para web era começar um *wireframe* de 960px e, somente quando o design já estivesse pronto, começar a pensar em outros pontos do projeto.

Depois que tivemos contato com o *responsive design*, o próximo passo seria reduzir as propostas de layout quando o de 960px estivesse pronto. Por que, então, fazer o contrário? Por que pensar em focar o desenvolvimento para telas pequenas, touch e outras restrições, antes mesmo da versão desktop? Há várias vantagens, tais como as que Luke Wroblewski mostra em seu excelente livro, *Mobile First* (http://ow.ly/dmxcd), que são:

- Estar preparado para o crescimento explosivo e novas oportunidades mobile emergentes (85% dos telefones vendidos em 2011 vieram com browser);
- Obriga você a se concentrar e focar nas principais características de seus produtos e sistemas (o que fazer quando se perde 80% do espaço em tela?);
- Permite-lhe proporcionar experiências inovadoras por meio de novas tecnologias (geolocalização, eventos de toque etc.).

A principal, talvez, seja o fato de a abordagem Mobile First forçá-lo a se **focar** no que é mais importante e essencial dentre as funcionalidades de seu projeto web. Afinal, quando se trata de um acesso feito por meio de um dispositivo móvel, as condições de uso

e as necessidades são bem diferentes do acesso feito por meio de outros tipos de dispositivos.

Não que seja necessário extirpar os conteúdos e funcionalidades que serão apresentados no desktop. Mas é uma abordagem que *obriga* a dar prioridade ao mais difícil, ao mais limitado.

Veja alguns pontos interessantes que podem elucidar sobre o porquê de Mobile First ser a abordagem mais eficiente para um site responsivo.

6.3 NOTAS GERAIS SOBRE MOBILE FIRST

De maneira geral, as pessoas que usam dispositivos móveis ficam com "um olho no gato e outro no peixe"; ou seja, estão realizando outras atividades, enquanto usam seus devices. Isso diz muita coisa sobre as diferenças de planejamento para mobile e em relação ao **foco** e **objetividade** necessários.

Existem determinados perfis de uso mobile: Pesquisar; Explorar/Divertir; Check in/Status; e Editar/Criar. Esses perfis (que serão mais bem explicados mais à frente) fazem parte de seu ponto de partida para a definição de sua estratégia mobile, baseado em qual destes seu público-alvo primário encaixa-se e o que seu site/sistema deve proporcionar.

Três dos pontos mais importantes em relação aos sites para dispositivos móveis são: **foco no conteúdo**, **foco no conteúdo** e **foco no conteúdo**. Se conseguir atender aos 3, praticamente metade de seu trabalho estará feito!

As pessoas precisam conseguir acessar diferentes áreas de um site de maneira fácil e intuitiva. Portanto, seja qual for o perfil em que se enquadre seu público, prover um sistema de navegação igualmente focado e intuitivo é sempre uma boa pedida.

Ao tratar-se de experiências móveis, a performance tem importância ainda maior, já que as pessoas esperam que sites mobile sejam tão rápidos quanto os vistos no desktop (ou mais!). A performance e otimização na web, em especial no contexto móvel, é fator decisivo de sucesso e crescimento de um negócio online. Em seu artigo *14 fatos sobre performance Web e otimizações* (http://ow.ly/e0Coh), Sérgio Lopes lembrou bem de alguns fatos a esse respeito:

- Para a Amazon, 1 segundo a mais no carregamento da página custa 1,6 bilhão de dólares por ano;
- O tráfego do Yahoo! aumenta 9% a cada 400 milissegundos de melhora na velocidade;
- Ao cortar 2,2s da landing page do Firefox, a Mozilla aumentou o número de downloads em 15%;
- Um experimento do Google aumentou o carregamento de 0,4s para 0,9s e o tráfego de buscas caiu 20%;
- A Microsoft mostrou que 2s a mais de latência no Bing diminuíam o faturamento em 4,3%;
- Uma diminuição de 6s para 1,2s na página do Shopzilla aumentou as vendas em 12%;
- 50% dos usuários mobile abandonam um site se ele não abre em até 10s (e 3 em cada 5 não voltam mais);
- Desde 2010, o Google considera a velocidade de carregamento dos sites no ranking de buscas.

Por último, caso você ainda esteja pensando que ganhará seus visitantes apostando todas suas fichas em design — aqui, *design* sendo usado no sentido que consta no dicionário, **um conjunto de técnicas e de concepções estéticas aplicadas à representação visual de uma ideia ou mensagem** —, então você ainda não entendeu muita coisa sobre projetar para mobile. Na verdade, ainda tem muito a aprender sobre a web em si!

É evidente que web design é imprescindível e um bom trabalho nesse campo do desenvolvimento sempre garantiu uma melhora na experiência do usuário e no uso do site. Entretanto, não é algo a ser considerado *primariamente*.

Excetuando-se raríssimos casos, web sites não são feitos de design, mas de **conteúdo**! Por isso, aposte na *simplicidade* quando projetar para dispositivos móveis. É sério!

6.4 COMO AS PESSOAS USAM DISPOSITIVOS MÓVEIS?

Perfis de uso

Não é preciso ser um grande observador para saber que as pessoas utilizam dispositivos móveis de maneiras diferentes das que usam dispositivos mais "tradicionais". Porém, já que, neste livro, estamos tratando de conhecimentos técnicos específicos, o mero fato de saber sobre esse comportamento diferente não contribui muito.

É preciso mais detalhes sobre ele, um conhecimento mais aprofundado sobre como as pessoas usam esses dispositivos e, a

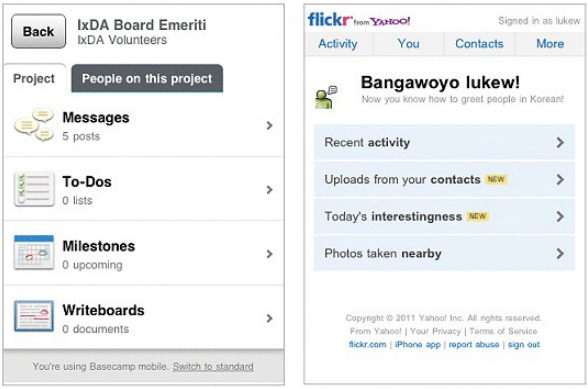
partir disso, projetar e planejar melhor a experiência mobile para cada um dos projetos web de que você fará parte.

Saiba, então, que existem determinados **perfis** quando o assunto é comportamento em dispositivos móveis. Por meio de uma série de estudos e observações, foram encontrados quatro principais perfis de uso nos dispositivos móveis:

- **Pesquisar (urgente, local):** pessoas que precisam de uma informação, seja ela qual for, de maneira urgente (informação esta frequentemente relacionada à localização física atual da pessoa). Por exemplo: "Do ponto em que estou agora, onde posso encontrar farmácias próximas?".
- **Explorar / Divertir (entediado, local):** pessoas que têm algum tempo ocioso e querem aproveitar para procurar alguma forma de distração ou diversão. Por exemplo: "Enquanto espero nesta fila, vou procurar informações sobre os últimos lançamentos do cinema".
- **Check in / Status (repetição, microtarefas):** para os mais "antenados" e/ou que precisam ficar atualizados caso informações que lhes são importantes sejam alteradas. Exemplo: "Preciso saber imediatamente se o status do meu pedido foi atualizado".
- **Editar / Criar (mudança urgente, microtarefas):** quando é preciso realizar uma tarefa importante que tenha de ser feita o mais rápido possível. Por exemplo, ajustar uma informação incorreta que uma pessoa tenha publicado em seu blog.

Então, por meio deles, é possível conhecer melhor e saber como as pessoas fazem uso de seus dispositivos. Conhecer esses perfis de uso permite que um melhor planejamento de seu projeto web seja feito, sempre com o objetivo de focar em seu público-alvo para proporcionar uma melhor experiência de uso do site ou sistema web, independentemente do device que ele está utilizando.

Dois exemplos interessantes são os do Basecamp e Flickr que, por meio de uma série de estudos e análises de estatísticas, conseguiram mapear quais as ações mais importantes que as pessoas mais usam frequentemente. A partir disso, projetaram a experiência mobile de seus sites para focar quase que unicamente nas tarefas que as pessoas precisam e querem realizar.



Decisão acertadíssima, pois, como mostrado nos perfis de uso anterior, o público dos serviços mostrados como exemplo precisam de **agilidade** e **foco** nas tarefas que pretendem executar!

Hora e lugar

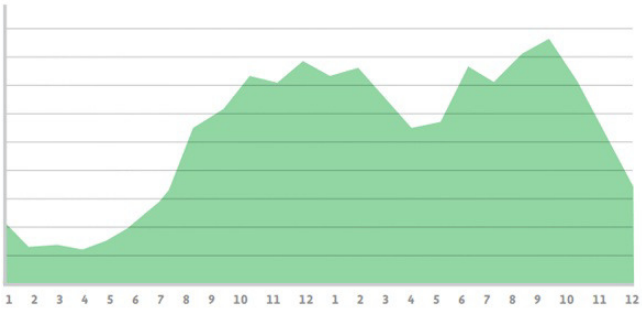
Além desses perfis de uso de dispositivos móveis, também foi possível avaliar outras informações importantes, como hora e lugar em que as pessoas costumam usar seus smartphones (http://ow.ly/dnNuz):

- 84% usam em casa;
- 80% usam em tempos variados de inatividade durante o dia;
- 74% usam enquanto esperam em filas ou por algum compromisso;
- 69% usam enquanto estão fazendo compras;
- 64% usam no trabalho;
- 62% usam enquanto estão assistindo TV;
- 47% usam enquanto se deslocam.

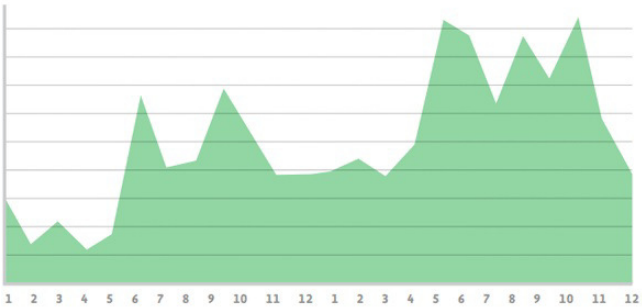
Estatísticas interessantíssimas, já que, para grande parcela dos desenvolvedores que têm contato com web design responsivo e Mobile First pela primeira vez, desenvolver para dispositivos móveis remete a sites para executivos de empresas, investidores, instituições financeiras e afins.

As estatísticas sobre horários de uso também são de grande importância. Afinal, é mais do que evidente o fato de que as pessoas usam seus diversos devices, de modo diferente e em horários variados. Tomando por base uma série de estatísticas fornecidas pelo Pocket (http://ow.ly/dnOT0) — serviço web que permite salvar artigos do interesse para que possam ser lidos posterioremente — em dispositivos Apple, é possível ter uma boa noção do que se está querendo dizer.

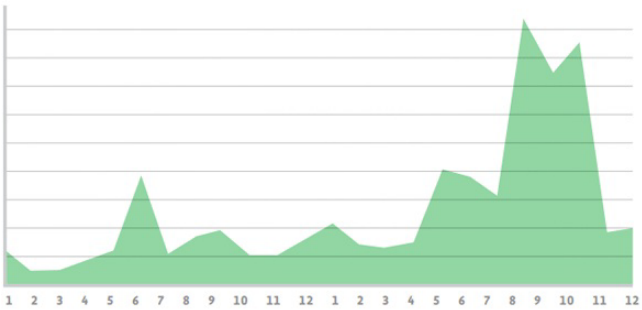
Artigos lidos no computador

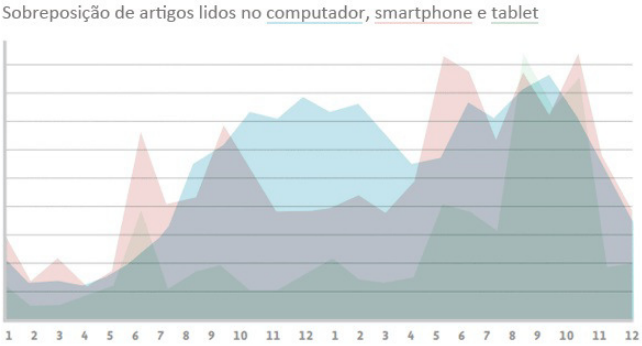


Artigos lidos no smartphone



Artigos lidos no tablet





No smartphone, por exemplo, há picos em:

- **6am:** bem cedo, na hora do café da manhã;
- **9am:** início do expediente para muitas pessoas;
- **5pm — 6pm:** fim do expediente, volta para a casa;
- **8pm — 10pm:** horário nobre (no sofá), hora de ir para a cama.

Já os maiores picos de tablet acontecem em horários considerados mais descontraídos e sem maiores compromissos e afazeres. **8pm — 10pm**, depois de um dia de trabalho inteiro, provavelmente já tomou-se banho, o jantar já foi servido e as crianças estão distraídas. Em suma, é um momento mais tranquilo.

Com base nas informações sobre hora e lugar em que as pessoas mais usam seus dispositivos móveis, é possível fazer um planejamento mais acurado sobre a estratégia do produto ou serviço online que se queira criar ou aprimorar.

Pense no seguinte: quais seriam os horários mais indicados para atualizações de gráficos e notícias para um público-alvo de investidores? Ou qual seriam os melhores horários para

atualizações naquele seu superaplicativo de fofocas sobre celebridades?

6.5 O CONTEÚDO É O REI

Certamente você já deve ter escutado ou lido a célebre frase: *O conteúdo é o rei*, mesmo antes de ter contato com essa história toda de *web design responsivo* e *mobile first*. De tanto repetida e divulgada na internet, já não se conhece sua autoria. De nada adiantaria o site mais bonito e usável do mundo se o conteúdo fosse inútil, vazio e/ou confuso.

Quando o assunto é desenvolvimento web com foco em dispositivos móveis, é preciso ter atenção redobrada quanto a isso. Mais do que nunca, agora é necessário dar ênfase total ao conteúdo e tarefas mais comuns que os frequentadores do site ou sistema fazem ou podem querer fazer. Quando o acesso é móvel, não há espaço para *firulas*, elementos que não agreguem nada à experiência de uso, menus inúteis ou quaisquer outros tipos de complicadores!

Na verdade, essa máxima deve ser levada em conta no desenvolvimento de qualquer site; mas, em mobile, esteja realmente atento a isso! Seus visitantes realmente vão querer ver o que seu site tem a oferecer nos primeiros segundos de sua visita. E caso encontrem algo diferente do conteúdo que estão procurando ou querendo, tenha certeza de que a taxa de rejeição de seu site aumentará.

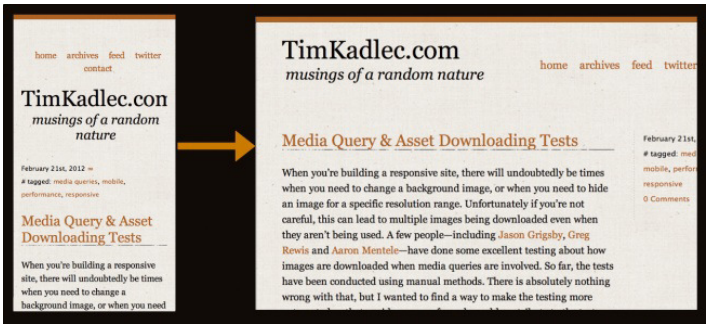
6.6 PADRÕES DE NAVEGAÇÃO MOBILE

Tudo bem, a esta altura você já deve ter internalizado a importância maior do conteúdo em seus sites. Depois disso, é preciso garantir que o esquema de navegação também esteja bem projetado, pronto para atender às necessidades dos visitantes prontamente, de maneira eficiente, agradável e sem dificuldades.

Apesar de o web design responsivo ser algo relativamente novo, alguns padrões de navegação já foram sistematizados. Brad Frost (http://ow.ly/dy56A) fez um ótimo trabalho em organizar a maioria deles, elencando prós e contras de cada abordagem. Organização esta apresentada de maneira resumida à seguir.

Top nav

Uma das soluções mais fáceis de implementar para a navegação é simplesmente mantê-la no topo. Por causa de sua facilidade de realização, essa abordagem é encontrada em muitos (talvez a maioria) dos sites responsivos.



Prós:

- Fácil de implementar;
- Não é preciso JavaScript;

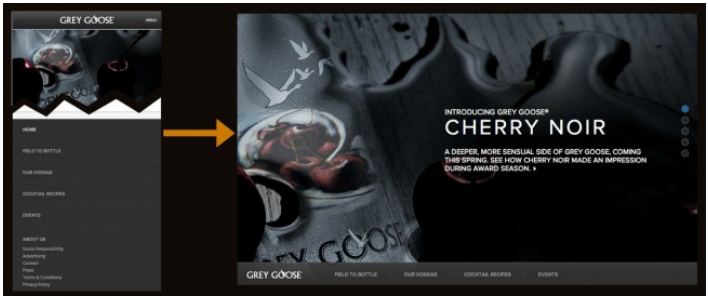
- Sem necessidade de *malabarismos* CSS.

Contras:

- Pode ocasionar problemas de altura;
- Não escalável;
- Pode ocasionar problemas com os links muito próximos.

Âncora no rodapé

Esta solução mantém a lista de navegação no rodapé do site, enquanto o cabeçalho contém um link de âncora simples apontando para a navegação no rodapé.



Prós:

- Fácil de implementar;
- Não é preciso JavaScript;
- Sem necessidade de *malabarismos* CSS;
- Único botão no cabeçalho.

Contras:

- A navegação por âncora pode desorientar algumas pessoas;
- Alguns podem considerar a solução não elegante.

Menu de seleção

Uma maneira de lidar com menus de navegação é transformá-los em uma lista de links em um menu de seleção para telas menores. Isso evita os problemas da abordagem *Top nav*, e é uma maneira inteligente de economizar espaço em tela.

Prós:

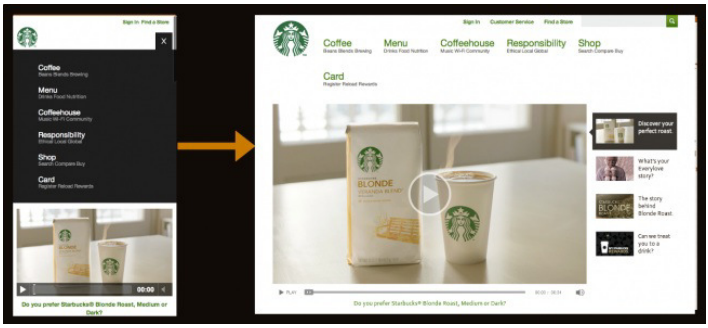
- Não ocupa espaço em tela;
- Mantém toda a interação no cabeçalho;
- Facilmente reconhecível.

Contras:

- Difícil de estilizar elementos `select` ;
- Potencialmente confuso;
- Submenus podem parecer estranhos;
- Há necessidade de JavaScript.

Alternância

Essa abordagem é semelhante à *Âncora no rodapé*, mas, em vez de saltar para baixo, para uma âncora na parte inferior da página, o menu abre no próprio local. É uma abordagem de boa aparência e é relativamente fácil de implementar.



Prós:

- Mantém a interação em um só local;
- Elegante;
- Facilmente escalável.

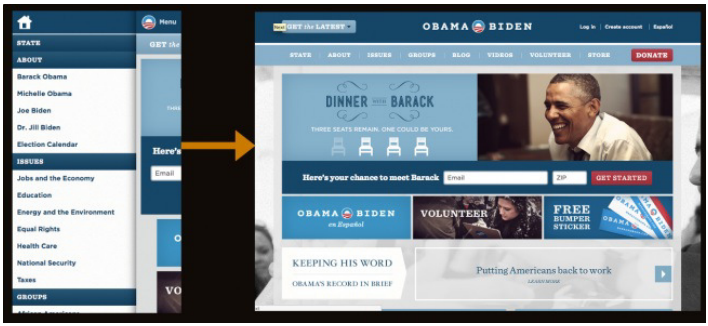
Contras:

- Performance (ao realizar a animação do slide);
- Precisa de JavaScript.

Slide à esquerda

O menu de navegação é acessado por um ícone do menu que, quando acessado, exibe as opções do menu por meio de uma animação de slide que começa à esquerda, movendo o conteúdo

principal para a direita.



Prós:

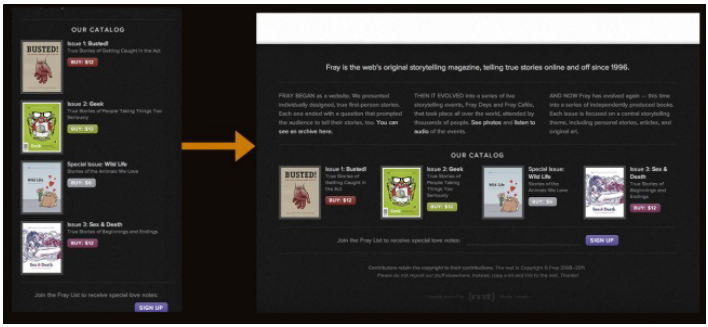
- Não ocupa espaço em tela;
- Mantém a interação em um só local;
- Boa aparência.

Contras:

- Técnica relativamente avançada;
- Não escala muito bem;
- Potencialmente confuso.

Somente no rodapé

Essa navegação é semelhante à *Top nav*, só que sem a âncora no cabeçalho.



Prós:

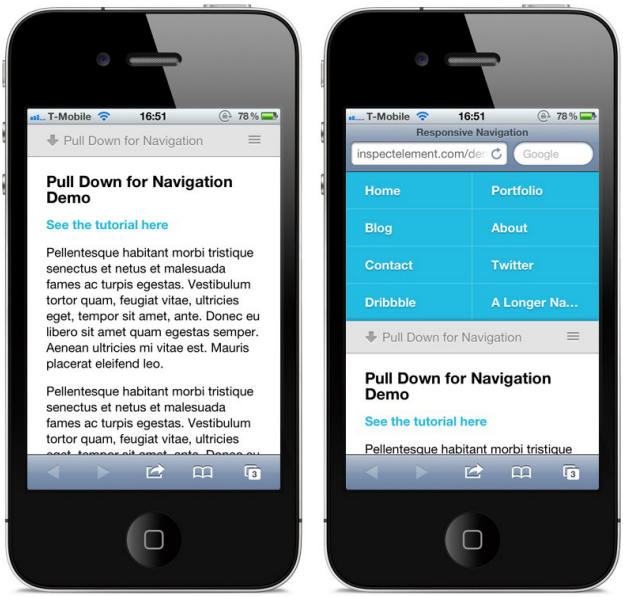
- Não ocupa espaço no cabeçalho.

Contras:

- Difícil de encontrar e acessar o menu (deve-se percorrer todo o conteúdo).

Navegação Pull Down

Revela o menu de navegação em um movimento de slide, que empurra o restante do conteúdo para baixo.



Prós:

- *Sexy*! ;-)
- O movimento de slide *top-down* já é uma convenção em smartphones.

Contras:

- Técnica relativamente avançada;
- Precisa de instrução de uso (geralmente, um label).

Considerações sobre padrões de navegação mobile

Foram apresentadas algumas das abordagens mais comuns quando falamos em sistemas de navegação para sites que se valem de web design responsivo. Certamente, não foram apresentadas

todas e, como o assunto é novo, sugestões de melhorias e novas abordagens serão desenvolvidas, propostas e discutidas. O importante é tentar chegar o mais próximo possível do objetivo de aliar navegação acessível, espaço em tela e facilidade de uso (o que não é uma tarefa simples).

Caso esteja interessado em como é possível implementar cada uma dessas abordagens, é indicado o artigo *Responsive menus: enhancing navigation on mobile websites* (http://ow.ly/dyb4x).

6.7 10 PRINCÍPIOS DE DESIGN PARA INTERFACES MOBILE

O consultor de mobile Jonathan Stark (e o nome Stark, por si só, já diz muito sobre sua capacidade) elaborou 10 princípios de design de interface móvel (http://ow.ly/dJnk8), que são bastante conhecidos. Eles serão apresentados a seguir, de forma sucinta.

Primeiro: mentalidade móvel

Devido às diferenças entre mobile e desktop, é imperativo que você faça uma imersão na mentalidade móvel antes de começar.

- **Seja focado:** mais não é melhor. Edite suas features impiedosamente. Você vai ter de deixar algumas coisas para fora.
- **Seja único:** saiba o que faz com que seu aplicativo seja diferente e amplifique isso. Há muitos peixes no mar de aplicativos móveis. Se não há nada de especial em sua aplicação, por que alguém iria querê-la?
- **Seja encantador:** dispositivos móveis são

intensamente pessoais. Eles são nossos companheiros constantes. Aplicativos que são amigáveis, confiáveis e divertidos são uma delícia de usar e as pessoas vão tornar-se muito ligadas à experiência.

- **Seja atencioso:** desenvolvedores de mobile muitas vezes concentram-se no que seria divertido desenvolver, no seu próprio modelo mental do aplicativo ou seus objetivos pessoais. Estes são bons lugares para começar, mas você tem de colocar-se no lugar de seus usuários se você espera criar experiências envolventes.

Segundo: contexto móvel

A imagem do profissional ocupado, correndo pelo aeroporto com um saco na mão e um smartphone no outro é o que muitas pessoas têm em mente quando se pensa sobre o contexto móvel. Certamente, este é um contexto; mas não é o único.

Para começar a colocarmo-nos na pele das pessoas que usam os sites e apps que desenvolvemos, é preciso considerar 3 grandes contextos móveis: entediado, ocupado e perdido.

- **Entediado:** há muitas pessoas usando seus smartphones no sofá de casa. Nesse contexto, as experiências imersivas e deliciosas, voltadas para uma sessão de maior tempo de uso, são uma grande possibilidade. Ainda assim, interrupções são altamente prováveis. Então, tenha certeza de que seu aplicativo pode continuar de onde o usuário parou. Exemplos: Facebook, Twitter, Angry Birds, navegador

web.

- **Ocupado:** este é o cenário "atravessando o aeroporto". A capacidade de realizar microtarefas de forma rápida e confiável com uma mão em um ambiente agitado é crítica. Lembre-se de que o usuário terá a "visão de túnel" nesse contexto, então objetivos claros e design arrojado são importantes. Exemplos: TripIt, e-mail, calendário, internet banking.
- **Perdido:** os usuários que estão no trânsito, em ambientes desconhecidos ou em ambientes familiares, mas interessados em algo ao redor, enquadram-se nessa categoria. Nesse contexto, a conectividade e a bateria são grandes preocupações, então você deve oferecer algum nível de suporte offline e ser econômico com a utilização de geolocalização e outros gargalos de bateria. Exemplos: Mapas, Yelp, Foursquare.

Para maiores informações, dados e estatísticas a esse respeito, relembre sobre os perfis de uso de dispositivos móveis vistos anteriormente.

Terceiro: orientações gerais

Diferentes aplicações exigem diferentes abordagens, design e técnicas. Dito isso, a natureza inerente de um dispositivo touchscreen de bolso sugere várias diretrizes globais, ou seja, as coisas que sempre importam.

- **Responsividade:** capacidade de resposta é absolutamente crítica. Se o usuário faz algo, sua

aplicação deve reconhecer a interação instantaneamente. Note-se que a capacidade de resposta e velocidade não são a mesma coisa. Tudo bem se certas operações levarem tempo. Apenas certifique-se de que o usuário saiba que as coisas estão funcionando.

- **Polidez:** polidez é extremamente valiosa. Em virtude da natureza de "companheiro constante" da relação com os smartphones, preste muita atenção nos pequenos detalhes. Isso será notado e apreciado!
- **Dedos:** com o advento das interfaces touchscreen, todo mundo está sempre falando "toque nisso" e "toque naquilo". Na realidade, é para o polegar que precisamos projetar. A menos que o usuário esteja interagindo com seu smartphone com as duas mãos, é quase impossível a interação com outro dedo qualquer e, mesmo quando se mexe com as duas mãos, o mais provável é digitar com os dois polegares. Polegares são o padrão.
- **Alvos:** devido a uma série de estudos sobre o tamanho e forma do polegar, parece que o número mágico para os elementos de UI amigáveis ao polegar é de 44 a 50 pixels. Existem muitas exceções, mas esta é uma regra geral. Você também deve estar consciente sobre onde colocar um target em relação aos outros (caso existam). Por exemplo, colocar o botão "Apagar" (*backspace*) ao lado do botão de envio em um aplicativo SMS seria uma má ideia.
- **Conteúdo:** a revolução das interfaces de toque é que elas permitem interagir diretamente com os

conteúdos. Isso remove abstrações (como mouse e trackpad) e está mais de acordo com a forma como nossos cérebros funcionam. Uma criança de 2 anos consegue usar um tablet sem dificuldade, mas um laptop já é um mistério. Aproveite o poder intuitivo da interface de toque, minimizando "dificultadores" (botões, abas, caixas de seleção, sliders e assim por diante) sempre que possível e colocando o conteúdo sempre em foco.

- **Controles:** quando é preciso adicionar controles, tente colocá-los na parte inferior da tela (em outras palavras, sob o conteúdo). Pense em uma máquina de somar, uma balança de banheiro ou até mesmo um computador: os controles estão abaixo do visor. Se não fosse, seria mais complicado de ver o que está acontecendo com o conteúdo, enquanto usamos.

Quarto: modelos de navegação

Há uma abundância de modelos de navegação inovadores para aplicativos móveis. Mas se você estiver tendendo a usar um dos modelos comuns, certifique-se de escolher o que faz mais sentido em sua aplicação.

Mais a esse respeito, leia novamente sobre padrões de navegação mobile na seção *Padrões de navegação mobile*.

Quinto: inputs do usuário

Digitação é complicado até mesmo nos melhores dispositivos, então, você deve fazer o que puder para tornar esse tipo interação

mais fácil para as pessoas. Por exemplo:

- Há cerca de uma dúzia de variações de teclado em smartphones populares (texto, número, e-mail, URL, dentre outros). Considere cada um dos seus campos de entrada e certifique-se de exibir o teclado que será mais útil para a entrada de dados que está sendo feita.
- Corretores ortográficos podem ser tão divertidos quanto frustrantes. Considerar cada um dos seus campos de entrada e decidir quais devem estar ativos (como autocorreção, autocapitalização e *autocomplete*).
- Se seu aplicativo precisa de muita digitação, você deve garantir que o teclado no sentido horizontal (*landscape*) esteja preparado para dedos "gordinhos".

Sexto: gestos

Um dos aspectos mais emblemáticos de interfaces modernas de toque é que elas suportam gestos para interações. Gestos são realmente muito legais e úteis, mas há várias coisas que você precisa levar em consideração.

- **Invisibilidade:** gestos são invisíveis, por isso sua descoberta é um problema. Você tem que decidir como revelar sua existência para o usuário. Uma das abordagens mais inteligentes é como na dos iPads promocionais expostos em lojas de varejo da Apple. Quando uma página é carregada pela primeira vez, as áreas de rolagem fazer um rápido "scroll reverso" até sua posição padrão. Isto imediatamente convida a um

gesto súbito do usuário sem ter de indicar, explicitamente, quais são as áreas de rolagem.

- **Duas mãos:** muitos gestos *multi touch* requerem o uso das duas mãos. Isso é particularmente evidente em aplicativos de mapas do iOS que usam o gesto de "abrir pinça" para aumentar o zoom. Mas se, por exemplo, você está viajando em uma cidade estrangeira com um café em uma mão e o telefone na outra, essa é uma limitação irritante. O Android tem uma abordagem melhor sobre essa questão, incluindo botões de *zoom in* e *zoom out* próximos ao mapa.
- **Bom de ter:** na maioria dos casos, gestos podem ser considerados como "bom de se ter", mas não são críticos. É como alguns tipos de atalhos de teclado: usuários avançados vão amá-los, mas a maioria das pessoas sequer sabem que estão lá.
- **Sem substituição:** ainda não existe um padrão comum para gestos; por isso, é muito cedo para a maioria dos aplicativos descartarem controles visíveis que podem ser usados com um único dedo.

Sétimo: orientação

A orientação portrait é, de longe, a mais popular. Por isso, otimize primeiro para esse caso. Se seu aplicativo precisa de muita digitação, você deve dar suporte a orientação landscape para as pessoas poderem acessar o teclado maior.

Quando a orientação muda inesperadamente, isso é desorientador! Se você acha que seu aplicativo será usado por longos períodos de tempo (por exemplo, o Kindle Reader),

considere a adição de uma trava de orientação diretamente no aplicativo.

Oitavo: comunicações

- **Forneça feedback:** forneça *feedback* imediato para cada interação. Se você não fizer isso, o usuário não vai saber se o aplicativo travou ou se erraram o alvo/interação que tentaram alcançar. Ele pode ser tátil ou visual. Se o usuário tiver solicitado uma ação que vai levar um longo tempo, apresente um indicador de *loading* para que saibam que a solicitação foi enviada e está sendo processada.
- **Alertas modais:** alertas modais são extremamente agressivo e invasivos para o fluxo do usuário. Assim, você só deve usá-los quando algo estiver seriamente errado. Mesmo assim, tente atenuar a intensidade, mantendo a linguagem reconfortante e amigável. Lembre-se de não usar alertas modais para mensagens do tipo "Para sua informação".
- **Confirmações:** quando você tem de pedir a alguém para confirmar uma ação, é aceitável exibir um diálogo de confirmação modal, como "Tem certeza de que deseja excluir este projeto?". As confirmações são menos invasivas do que alertas, porque elas aparecem em resposta a uma ação do usuário e, portanto, em um contexto até esperado. Lembre-se de fazer com que o botão padrão seja a escolha mais segura da caixa de diálogo para ajudar a evitar inadvertidas ações destrutivas.

Nono: inicialização do aplicativo

Quando uma pessoa volta para seu aplicativo depois de já o ter usado antes, você deve retomar as operações exatamente onde ela parou. Isso dará a ilusão de velocidade e contribue para uma sensação geral de capacidade de resposta.

Se possível, a tela de inicialização exibida quando o aplicativo é iniciado (ou acessado) deve ser uma tela desprovida de conteúdo, apresentando somente o logo ou alguma imagem. Qualquer coisa que pareça interativa (como botões, links, ícones, conteúdo), criará frustração ao convidar para um interação falha.

Resista fortemente à tentação de colocar elementos ou materiais de *branding* na tela de inicialização. Eles fazem as pessoas sentirem-se como se estivessem vendo um anúncio e eles ressentem-se com isso, tendendo a ir embora.

Décimo: primeiras impressões

- **Seu ícone:** ele tem de competir por atenção em um mar de outros ícones. Sendo esse o caso, pense nisso mais como um cartão de visitas do que uma obra de arte. Seja literal: mostre o que seu aplicativo faz. Use uma silhueta forte e opte por um mínimo de textos. Um ícone polido sugere um aplicativo polido, então vale a pena dedicar tempo sério e dinheiro para fazê-lo direito.
- **Primeiro lançamento:** o primeiro lançamento é uma situação para encantar ou decepcionar. Se alguém fica confuso ou frustrado ao tentar familiarizar-se com seu

aplicativo, entao vai abandoná-lo o mais rápido que conseguir. Se ele fornece funcionalidades complexas, você pode querer incluir um "Dicas e Truques" ou algumas telas com orientações gerais. Note que essas dicas não servem para substituir um bom projeto. Se você encontrar-se criando um monte de texto de ajuda, é um forte indicativo de que a UI de sua aplicação precisa de um retrabalho.

Capítulo 7

CONTINUANDO SEUS ESTUDOS

Eis que você chegou ao último capítulo, são e salvo! Para ajudar no aprimoramento de seus conhecimentos, foi feita uma compilação de recursos sobre web design responsivo que, certamente, vai render muitas e muitas horas de estudos!

7.1 LIVRO A WEB MOBILE

Sérgio Lopes publicou um livro, também pela editora **Casa do Código**, que aprofunda em vários outros assuntos de mobile e responsivo. O livro **A web mobile** (http://sergiolopes.org/livro-web-mobile/) trata de assuntos como eventos JavaScript para touchscreen; como lidar com telas retina; RESS; conditional loading; usabilidade em mobile e muito mais.

É um ótimo complemento a este livro se você quer entrar de vez no desenvolvimento web responsivo para dispositivos móveis. Altamente recomendado!

7.2 ARTIGOS/TUTORIAIS

Ler artigos e tutoriais é a base para manter-se atualizado sobre o que acontece no mundo da responsividade. Confira algumas boas referências e continue os estudos!

Considerations for mobile design: behavior

O artigo *Considerations for mobile design: behavior* (http://ow.ly/e0r9U) traz informações de caráter importantíssimo sobre o comportamento no ambiente mobile, diferenças entre formas de interação, UIs mobile e mais. Leia!

Convert a menu to a dropdown for small screens

Nos padrões de navegação mobile (seção *Padrões de navegação mobile*), você viu que uma das técnicas é transformar o menu em uma lista de links em um de seleção para telas menores. No *Convert a menu to a dropdown for small screens* (http://ow.ly/dUNa6), você vai saber, passo a passo, como realizar esse efeito.

Design process in the responsive age

O artigo *Design process in the responsive age* (http://ow.ly/e0kZV) aborda pontos muito interessantes sobre o processo de design para sites com web design responsivo. Vale a pena dar uma olhada, pois o planejamento de um projeto é uma de suas principais e imprescindíveis etapas!

Hiding and revealing portions of images

Em *Hiding and revealing portions of images* (http://ow.ly/e0prm), você vai aprender técnicas mais avançadas de

apresentação de imagens em web designs responsivos e como ocultar determinadas partes de imagens quando for mais adequado.

Optimizing your email for mobile devices with the @media query

No tutorial *Optimizing your email for mobile devices with the @media query* (http://ow.ly/dUNmd), você aprenderá como tornar seus e-mails para mobile usando media queries. Afinal, é importante garantir mensagens e newsletters responsivos para uma experiência mobile completa.

Responsive data tables

No tutorial *Responsive data tables* (http://ow.ly/dUN5f), você conhecerá algumas ideias excelentes sobre como tratar tabelas e dados tabulares em projetos com web design responsivo.

Responsive Workflow

Em *Responsive workflow* (http://ow.ly/e0xEK), você vai encontrar a proposta de uma metodologia sobre como planejar, desenvolver e lançar projetos responsivos. É uma leitura interessante que pode ser a base fundamental de seus projetos a partir de agora!

The mobile playbook

The mobile playbook: o guia do executivo ocupado para vencer no mercado para celular (http://ow.ly/e0AK3), da Google, é um livro interativo com informações preciosas sobre o mundo de

dispositivos móveis, marketing mobile, considerações, dicas e estatísticas sobre como vencer no mercado móvel.

7.3 BOOKMARKLETS

Um *bookmarklet* é um trecho em JavaScript que é armazenado como uma URL que pode ser salvo nos favoritos de qualquer navegador. É interessante que você tenha alguns também para o desenvolvimento com web design responsivo.

Viewport Resizer

O Viewport Resizer (http://ow.ly/yypRs) atualmente é um dos melhores bookmarklets relacionados a testes de responsividade. Ao ser ativado, uma barra superior é mostrada no site e permite que o tamanho da viewport seja redimensionado em dimensões predefinidas. Para mais informações, assista a um screencast feito por este que vos escreve: http://ow.ly/yyq2z.

Dimensions Toolkit

Com o mesmo intuito que o Viewport Resizer, mas com mais opções e fluidez de uso, o Dimensions Toolkit (http://ow.ly/yyqdd) é realmente uma ótima ferramenta para testes responsivos! Pode ser usada como um plugin (pago) do Google Chrome ou por meio de um endereço web próprio (gratuito).

Media Query Bookmarklet

Ao ativar o Media Query Bookmarklet (http://ow.ly/dUO7c), ele analisa os arquivos CSS incluídos da página atual e cria uma

série de declarações de media query que são mostradas na tela em tempo real, enquanto você redimensiona a janela do navegador.

Responsive Design bookmarklet

O Responsive Design Bookmarklet (http://ow.ly/e0kvR) permite que você crie um bookmarklet em tempo real para testes com web design responsivo. Defina os tamanhos que gostaria de realizar seus testes e salve-o personalizado para uso futuro.

Yubi

O espaço útil de tela de pessoas em dispositivos móveis é mais restrito; então, uma das boas práticas é oferecer elementos de interação maiores. Ao ativar o Yubi (http://ow.ly/yyqtP), uma marcação segue o ponteiro do mouse, mostrando qual é o tamanho médio de um dedo para que se possa mensurar melhor as interações mobile em um site — em 2 versões: dedo comum e dedo gordinho! ;-)

7.4 ESBOÇO E PLANEJAMENTO

Já que, a partir de agora, você só trabalha com web design responsivo e Mobile First, seus esboços também devem sofrer uma alteração para acompanhar esta nova fase. Simplesmente escolha um dos mostrados e comece a desenhar!

A simple device diagram for responsive design planning

A simple device diagram for responsive design planning

(http://ow.ly/e0g0w), como sugere o próprio nome, é um diagrama simples para ser usado no planejamento de web design responsivo, mostrando uma régua com as principais resoluções dos devices mais usados no mundo.

Responsive design sketchbook

Que tal um caderno quadriculado especialmente feito para planejamento de web designs responsivos? Esta é, justamente, a proposta do *Responsive design sketchbook* (http://ow.ly/dUMH9). Por um preço acessível, é possível adquirir esse caderno espiral com 50 páginas para o seu planejamento.

Responsive web design sketch sheets

Responsive web design sketch sheets (http://ow.ly/dPHlf) é um conjunto de modelos demarcados, indicando vários tamanhos de tela, para o planejamento e esboço das telas em sites responsivos serem bem rápidos.

UX sketching and wireframing templates for mobile projects

UX sketching and wireframing templates for mobile projects (http://ow.ly/dSBK0) é um conjunto de 28 folhas para esboços e *wireframes* em PDF para diversas plataformas, dentre as quais estão: Android, BlackBerry, iOS (iPad and iPhone), Meego, Symbian, webOS e Windows Phone 7.

7.5 FERRAMENTAS

Agora que você já sabe bastante coisa sobre web design responsivo, é importante acrescentar no seu kit pessoal algumas ferramentas para ajudar com o desenvolvimento responsivo.

Fluid grids

Já foram mostrados vários frameworks que possuem grids fluidos para desenvolvimento de sites responsivos. Mas, e se você quiser ter o seu próprio grid? Com *Fluid Grids* (http://ow.ly/dUOhD), essa tarefa será mais fácil do que você imagina: somente especifique quantas colunas, o tamanho de cada uma e o espaço entre elas e pronto, faça o download de seu grid personalizado!

PXtoEM

PXtoEM (http://ow.ly/e0j9U) é uma tabela interativa de conversão de pixels para outras unidades, como EMs, porcentagem e pontos. Possui alguns valores predefinidos e permite a inserção de valores-base para a conversão.

resizeMyBrowser

No *resizeMyBrowser* (http://ow.ly/dUNNr), você encontra botões com as resoluções mais comuns (dos dispositivos mais populares) que, ao serem clicados, abrem uma nova janela com aquele tamanho escolhido.

The responsive calculator

Obviamente, você lembra-se da fórmula mágica do web design responsivo (seção *A fórmula mágica do web design responsivo*).

Mas, caso tenha alguns problemas/dificuldades com cálculos, *The responsive calculator* (http://ow.ly/e0eDr) é uma ferramenta imprescindível para você, já que se trata de uma espécie de calculadora específica para responsive web design, ajudando a converter pixels para porcentagens.

7.6 INSPIRAÇÃO

Agora que você já sabe bastante sobre web design responsivo, é de interesse observar alguns projetos, exemplos de sites e recursos já desenvolvidos e/ou baseados na responsividade.

Media queries

Uma das primeiras e mais conhecidas galerias de sites responsivos, no *Media queries* (http://ow.ly/e0yev) é possível acompanhar projetos responsivos de excelente qualidade.

Mobile UI patterns

Se você está começando agora, ou mesmo se já está dentro do mundo do desenvolvimento mobile, gostará bastante de conhecer o *Mobile UI Patterns* (http://ow.ly/e0yyt). Ele apresenta soluções de UI para mobile separadas por categorias.

7.7 JAVASCRIPT PURO

Em alguns casos, soluções em JavaScript puro também são uma ótima pedida para a solução de problemas com web design responsivo.

Adapt.js

Adapt.js (http://ow.ly/dUOSy) é um JavaScript muito leve (menos de 1KB quando minificado) que determina qual arquivo CSS deve ser carregado antes de o navegador processar uma página. Se o navegador é redimensionado, *Adapt.js* simplesmente verifica a nova largura e provê apenas o CSS necessário, quando necessário.

foresight.js

foresight.js (http://ow.ly/e0rEY) provê a informação que diz se o dispositivo que está fazendo o acesso é capaz de visualizar imagens em alta resolução (como o iPad 3ª geração, por exemplo) antes da imagem ser solicitada no servidor.

Respond

Respond (http://ow.ly/dPGhQ) é um *polyfill* que faz com que IE6-8 reconheçam `min-width` e `max-width` para o uso de media queries.

Retina images

Retina images (http://ow.ly/e0duN) automaticamente serve imagens em alta resolução para dispositivos que possuem *retina display*. Seu uso é bastante simples e, para os sites que querem se valer deste extra, realmente vale a pena!

7.8 PLUGINS JQUERY

É possível, sim, fazer bons sites usando somente HTML e CSS

(pelo menos no nível atual dessas tecnologias). Mas fato é que, para proporcionar experiências realmente marcantes e envolventes, um toque de JavaScript (JS) faz-se necessário. Como, não por acaso, jQuery é a biblioteca JS mais usada no mundo, então, seguem alguns plugins relacionados à web design responsivo.

Camera

Camera (http://ow.ly/dPCg6) permite fazer slideshows responsivos incríveis. Conta com loading, miniaturas, opções e callbacks. Seguindo o estilo jQuery de ser, é simplíssimo de usar.

FitText

FitText (http://ow.ly/dPB4V) torna a propriedade CSS `font-size` flexível. Utilize este plugin em seu projeto para conseguir títulos escaláveis da largura total do elemento-pai.

Flexslider

Flexslider (http://ow.ly/dU3Y0) é outro excelente plugin para slideshows responsivos. Funciona com marcação simples; dá suporte a vários efeitos de animação, sliders múltiplos, API, callbacks e tudo o mais que um plugin completo possui.

Isotope

Isotope (http://ow.ly/e0bNM) é um excelente plugin que, além de reorganizar elementos na tela (com um efeito bem interessante) em tempo real, conforme o tamanho da janela do navegador, permite também usar opções de filtragem para os elementos que

estão sendo apresentados.

scrolldeck

O *scrolldeck* (http://ow.ly/dPEjn) permite fazer sites e apresentações com "decks rolantes"; ou seja, você clica em um link e a página rola até o ponto especificado.

Supersized!

Supersized! (http://ow.ly/dPEHZ) permite slideshow de imagens em tela cheia, seja qual for a resolução usada por quem acessa o site.

TinyNav.js

Nos padrões de navegação mobile (seção *Padrões de navegação mobile*), você viu que uma das técnicas é transformar o menu em uma lista de links em um de seleção para telas menores. O *tinyNav.js* (http://ow.ly/e0c1I) serve justamente para isso!

Wookmark

Wookmark (http://ow.ly/e0cfE) detecta o tamanho da janela do navegador que está realizando o acesso e, a partir disso, rearranja os elementos da página em colunas. Na prática, é uma alternativa mais "enxuta" ao plugin Isotope.

7.9 TESTES DE RESPONSIVIDADE

Existem diversas ferramentas pela web afora que auxiliam a fazer os testes de seus web designs responsivos, tornando a tarefa

de verificar como as coisas estão ficando em diferentes resoluções de maneira mais prática. Confira algumas.

Demonstrating responsive design

Demonstrating Responsive Design (http://ow.ly/dULxj) permite visualizar como um site fica visto em um celular — em landscape e portrait —, tablet — também com as 2 orientações — e desktop, alterando as visualizações de forma interativa.

Responsive play

Responsive Play (http://ow.ly/dSD1b) é uma ferramenta brasileira muito interessante: ela ajuda bastante a tarefa de checar o desenvolvimento com web design responsivo ao permitir visualizar uma animação de uma viewport, estendendo-se conforme uma faixa de largura e tempo definidos por quem a usa.

Responsive web design testing tool

Com *Responsive web design testing tool* (http://ow.ly/dUNFe), é possível visualizar qualquer site nas larguras de 240, 320, 480, 768 ou 1024 pixels. Bastante útil e economiza um bom tempo por mostrar todas as "versões" simultaneamente.

Screenfly

A partir de uma URL informada, *Screenfly* (http://ow.ly/e0cUU) permite escolher a visualização de um site em diversos tipos de dispositivos (desktop, tablet, mobile e TV), inclusive com variações das principais resoluções/modelos de cada um.

Screenqueri.es

Screenqueri.es (http://ow.ly/e0dh6) é uma ferramenta de testes de sites em várias resoluções para os adoradores de *pixel perfect*. Além de permitir visualizar o site em diversas resoluções predefinidas pelos modelos de aparelhos mais conhecidos, permite também arrastar largura e altura para ver como o site apresenta-se dentro de um grid de marcação. Muito bom para testá-lo em vários breakpoints diferentes.

7.10 TEMPLATES E FRAMEWORKS

Depois de já ter aprendido bem os princípios e técnicas do web design responsivo, você pode querer agilizar os projetos usando modelos ou frameworks responsivos. Veja os melhores e mais usados no mundo, estude cada um deles e encontre o que melhor se adequa ao seu estilo.

1140 CSS grid

1140 CSS grid (http://ow.ly/dPJw1) encaixa-se perfeitamente em um monitor de 1280px e, em monitores menores, torna-se fluído e adapta-se à largura do navegador. A partir de certo ponto, ele usa media queries para servir uma versão móvel que, essencialmente, empilha todas as colunas para que o fluxo de informação ainda faça sentido.

320 and up

320 and up (http://ow.ly/dPLxU) é um dos modelos para design responsivo mais conhecidos. Além das características mais

óbvias (como layout fluido, imagens flexíveis etc.), ele tem a vantagem de agradar aos que gostam de LESS e SASS para mexer com o CSS, e já vem com Modernizr e Selectivizr.

Foundation

Foundation (http://ow.ly/dQqdG) autointitula-se *o mais avançado framework responsivo de front-end do mundo*. Seu poderoso sistema de grid é feito com SASS, assim como seus elementos de UI e demais características. Já conta com alguns *add-ons*, como templates HTML, ícones de fontes, tabelas responsivas e mais.

Frameless

Frameless (http://ow.ly/dPIZS) é um grid feito para designs responsivo com colunas de largura fixa, ou seja, em vez de tudo ser fluido, dependendo da resolução, mais ou menos colunas são exibidas na tela. Seu slogan é: *"Adaptar coluna por coluna, não pixel por pixel"*.

Initializr

Initializr (http://ow.ly/e0AjK) é um gerador de templates para HTML5 para ajudar a começar a iniciar novos projetos. O kit conta com HTML5 Boilerplate em combinação com Twitter Bootstrap, jQuery e Modernizr. Ele gera um modelo limpo personalizável com apenas o que é preciso para começar o projeto. Altamente recomendado!

inuitCSS

O *inuitCSS* (http://ow.ly/dPKio) é um framework para design responsivo que conta com uma série de plugins para breadcrumb, dropdown, navegação, dentre outros.

Responsive grid system

Responsive grid system (http://ow.ly/e0z2M) é uma maneira rápida, fácil e flexível de criar web sites responsivos. Seu sistema de grids é realmente fácil de entender e usar por meio de uma marcação simples e rápida.

Skeleton

Skeleton (http://ow.ly/dPID6) é uma coleção de arquivos CSS e JS que forma uma série de componentes de UI para ajudar a desenvolver rapidamente sites que são bem vistos em qualquer resolução.

Twitter bootstrap

Twitter bootstrap (http://ow.ly/dPHWq) é um framework completo para front-end. Possui diversos componentes de UI, plugins JavaScript e um grid responsivo que agilizará bastante o desenvolvimento de seus projetos futuros.

Para mais informações sobre esses e outros frameworks responsivos, confira o artigo *A look at responsive CSS frameworks* (http://ow.ly/yyqNn).

7.11 PALAVRAS FINAIS

Não deve ser nenhuma surpresa que as técnicas e

conhecimentos mostrados no livro são tão dinâmicos e mutáveis quanto a própria web! Então, nada de achar que agora você é o Sr. (ou Sra.) Responsividade.

Você está apenas começando a jornada e, como diria *Morpheus*, **há uma diferença entre conhecer o caminho e trilhar o caminho**.

Bons estudos!