CAPÍTULO 7

Web para dispositivos móveis

"A iniciativa da Internet móvel é importante, informações devem ser igualmente disponíveis em qualquer dispositivo"
— Tim Berners-Lee

7.1 - SITE MOBILE OU MESMO SITE?

O volume de usuários que acessam a Internet por meio de dispositivos móveis cresceu exponencialmente nos últimos anos. Usuários de iPhones, iPads e outros smartphones e tablets têm demandas diferentes dos usuários Desktop. Redes lentas e acessibilidade em dispositivos limitados e multitoque são as principais diferenças.

Como atender a esses usuários?

Para que suportemos usuários móveis, antes de tudo, precisamos tomar uma decisão: fazer um Site exclusivo - e diferente - focado em dispositivos móveis ou adaptar nosso Site para funcionar em qualquer dispositivo?

Vários grandes sites da Internet - como Google, Amazon, UOL, Globo.com etc - adotam a estratégia de ter um Site diferente voltado para dispositivos móveis. É comum usar um subdomínio diferente como "m." ou "mobile.", como http://m.uol.com.br.

Essa abordagem é a que traz maior facilidade na hora de pensar nas capacidades de cada plataforma, Desktop e mobile, permitindo que entreguemos uma experiência customizada e otimizada para cada situação. Porém, há diversos problemas envolvidos:

- Como atender adequadamente diversos dispositivos tão diferentes quanto um smartphone com tela pequena e um tablet com tela mediana? E se ainda considerarmos os novos televisores como Google TV? Teríamos que montar um

Site específico para cada tipo de plataforma?

- Muitas vezes esses Sites mobile são versões limitadas dos Sites de verdade e não apenas ajustes de usabilidade para o dispositivo diferente. Isso não frustra o usuário que, cada vez mais, usa dispositivos móveis para completar as mesmas tarefas que antes fazia no Desktop?

- Dar manutenção em um Site já é bastante trabalhoso, imagine dar manutenção em dois - um mobile e um normal.

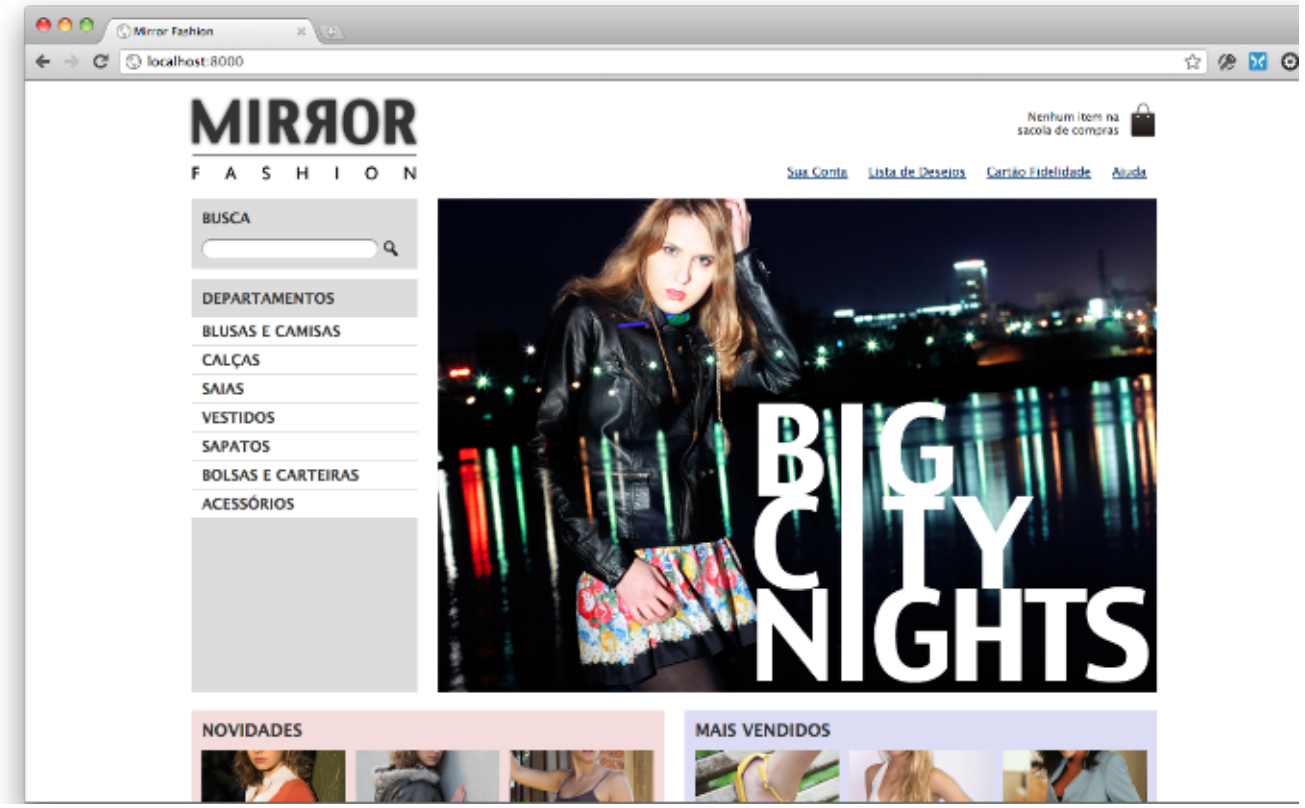
- Você terá conteúdos duplicados entre Sites "diferentes", podendo prejudicar seu SEO se não for feito com cuidado.

- Terá que lidar com redirects entre URLs móveis e normais, dependendo do dispositivo. O usuário pode receber de um amigo um link para uma página vista no Site normal; mas se ele abrir esse email no celular, terá que ver a versão mobile desse link, sendo redirecionado automaticamente. E mesma coisa no sentido contrário, ao mandar um link de volta para o Desktop.

Uma abordagem que vem ganhando bastante destaque é a de ter um único Site, acessível em todos os dispositivos móveis. Adeptos da ideia da Web única (**One Web**) consideram que o melhor para o usuário é ter o mesmo Site do Desktop normal também acessível no mundo móvel. É o melhor para o desenvolvedor também, que não precisará manter vários Sites diferentes. E é o que garante a compatibilidade com a maior gama de aparelhos diferentes.

Certamente, a ideia não é fazer o usuário móvel acessar a página exatamente da mesma maneira que o usuário Desktop. Usando truques de CSS3 bem suportados no mercado, podemos usar a mesma base de layout e marcação porém ajustando o design para cada tipo de dispositivo.

Nossa página no Desktop agora é mostrada assim:



Queremos que, quando vista em um celular, tenha um layout mais otimizado:

Como desenvolver um Site exclusivo para Mobile?

A abordagem que trataremos no curso é a de fazer adaptações na mesma

página para ser compatível com CSS3. Como faremos caso queiramos fazer um Site exclusivo para mobile?

Do ponto de vista de código, é a abordagem mais simples: basta fazer sua página com design mais enxuto e levando em conta que a tela será pequena (em geral, usa-se width de 100% para que se adapte à pequenas variações de tamanhos de telas entre smartphones diferentes).

Uma dificuldade estará no servidor para detectar se o usuário está vindo de um dispositivo móvel ou não, e redirecioná-lo para o lugar certo. Isso costuma envolver código no servidor que detecte o navegador do usuário usando o User-Agent do navegador.

É uma boa prática também incluir um link para a versão normal do Site caso o usuário não queira a versão móvel.

7.2 - CSS MEDIA TYPES

Desde a época do CSS2, há uma preocupação com o suporte de regras de layout diferentes para cada situação possível. Isso é feito usando os chamados *media types*, que podem ser declarados ao se invocar um arquivo CSS:

```
<link rel="stylesheet" href="site.css" media="screen" />
<link rel="stylesheet" href="print.css" media="print" />
<link rel="stylesheet" href="handheld.css" media="handheld" />
```

Outra forma de declarar os media types é separar as regras dentro do próprio arquivo CSS:

```
@media screen {
  body {
    background: blue;
    color: white;
  }
}

@media print {
  body {
    background: white;
    color: black;
  }
}
```

O media type *screen* determina a visualização normal, na tela do Desktop. É

muito comum também termos um CSS com media type *print* com regras de impressão (por exemplo, retirar navegação, formatar cores para serem mais adequadas para leitura em papel etc).

E havia também o media type *handheld*, voltado para dispositivos móveis. Com ele, conseguíamos adaptar o Site para os pequenos dispositivos como celulares WAP e palmtops.

O problema é que esse tipo *handheld* nasceu em uma época em que os celulares eram bem mais simples do que hoje, e, portanto, costumavam ser usados para fazer páginas bem simples. Quando os novos smartphones touchscreen começaram a surgir - em especial, o iPhone -, eles tinham capacidade para abrir páginas completas e tão complexas quanto as do Desktop. Por isso, o iPhone e outros celulares modernos ignoram as regras de *handheld* e se consideram, na verdade, *screen*.

Além disso, mesmo que *handheld* funcionasse nos smartphones, como trataríamos os diferentes dispositivos de hoje em dia como tablets, televisões etc?

A solução veio com o CSS3 e seus *media queries*.

7.3 - CSS3 MEDIA QUERIES

Todos os smartphones e navegadores modernos suportam uma nova forma de adaptar o CSS baseado nas propriedades dos dispositivos, as **media queries** do CSS3.

Em vez de simplesmente falar que determinado CSS é para *handheld* em geral, nós podemos agora indicar que determinadas regras do CSS devem ser vinculadas a propriedades do dispositivo como tamanho da tela, orientação (landscape ou portrait) e até resolução em dpi.

```
<link rel="stylesheet" href="base.css" media="screen">
<link rel="stylesheet" href="mobile.css" media="(max-width: 480px)">
```

Outra forma de declarar os media types é separar as regras dentro do mesmo arquivo CSS:

```
@media screen {
  body {
    font-size: 16px;
  }
}

@media (max-width: 480px) {
  body {
    font-size: 12px;
  }
}
```

Repare como o atributo *media* agora pode receber expressões complexas. No caso, estamos indicando que que queremos que as telas com largura máxima de 480px tenham uma fonte de 12px.

Você pode testar isso apenas redimensionando seu próprio navegador Desktop para um tamanho menor que 480px.

7.4 - Viewport

Mas, se você tentar rodar nosso exemplo anterior em um iPhone ou Android de verdade, verá que ainda estamos vendo a versão Desktop da página. A regra do *max-width* parece ser ignorada!

Figura 7.3: Site atual rodando num celular Android

Na verdade, a questão é que os smartphones modernos têm telas grandes e resoluções altas, justamente para nos permitir ver sites complexos feitos para Desktop. A tela de um iPhone 4S por exemplo é 960px por 640px. Celulares

Android já chegam a 1280px, o mesmo de um Desktop.

Ainda assim, a experiência desses celulares é bem diferente dos Desktops. 1280px em uma tela de 4 polegadas é bem diferente de 1280px em um notebook de 13 polegadas. A resolução muda. Celulares costumam ter uma resolução em dpi bem maior que Desktops.

Como arrumar nossa página?

Os smartphones sabem que considerar a tela como 1280px não ajudará o usuário a visualizar a página otimizada para telas menores. Há então o conceito de *device-width* que, resumidamente, representa um número em pixels que o fabricante do aparelho considera como mais próximo da sensação que o usuário tem ao visualizar a tela.

Nos iPhones, por exemplo, o *device-width* é considerado como 320px, mesmo com a tela tendo uma resolução bem mais alta.

Por padrão, iPhones, Androids e afins costumam considerar o tamanho da tela visível, chamada de *viewport* como grande o suficiente para comportar os Sites Desktop normais. Por isso a nossa página foi mostrada sem zoom como se estivéssemos no Desktop.

A Apple criou então uma solução que depois foi copiada pelos outros smartphones, que é configurar o valor que julgarmos mais adequado para o viewport:

```
<meta name="viewport" content="width=320">
```

Isso faz com que a tela seja considerada com largura de 320px, fazendo com nosso layout mobile finalmente funcione e nossas media queries também.

Melhor ainda, podemos colocar o viewport com o valor `device-width` definido pelo fabricante, dando mais flexibilidade com dispositivos diferentes com tamanhos diferentes:

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

7.5 - Exercícios: adaptações para mobile

1. Vamos adaptar nossa home page (**index.html**) para mobile.

Comece declarando a meta tag com o viewport dentro do `<head>` da **index.html**:

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

Vamos escrever nosso CSS de adaptação em um novo arquivo, chamado **mobile.css**. Crie esse arquivo e o importe no head do index.html. Use media queries para que ele só seja aplicado em resoluções de no máximo 320px (celulares comuns)

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

2. Nossa página hoje tem o tamanho fixo em 940px e é centralizada (com o uso do seletor **.container**). Para deixarmos a página mais flexível nos celulares, precisamos remover esse tamanho absoluto e colocar algum que faça mais sentido em telas menores, onde queremos ocupar quase que a tela toda (deixando apenas uma pequena margem). Para isso, edite **mobile.css**:

```
.container {
```

```
  width: 96%;
}
```

Já é possível redimensionar a tela para 320px e ver que o site começa a se adaptar. Mas ainda há bastante trabalho pela frente.

3. Próximo passo: vamos ajustar os elementos do topo da página. Vamos centralizar o logotipo na página, esconder as informações (secundárias) sobre a Sacola e ajustar o menu para ficar abaixo do logo e não mais posicionado à direita.

```
header h1 {
  text-align: center;
}
header h1 img {
  max-width: 50%;
}
.sacola {
  display: none;
}
.menu-opcoes {
  position: static;
  text-align: center;
}
```

Lembre-se que, anteriormente, nosso menu estava com `position:absolute` para ficar a direita do logo. Agora, queremos deixá-lo fluir abaixo do logo; bastou restaurar o `position:static`.

Teste novamente com o navegador redimensionado. Está melhorando?

4. Ajustamos a posição do menu do topo e, automaticamente, os links se posicionaram formando duas linhas. Mas repare como estão próximos uns dos outros. Será que o nosso usuário consegue clicar neles usando seu celular? Vamos aumentar o espaço entre eles:

```
.menu-opcoes ul li {
  display: inline-block;
  margin: 5px;
}
```

5. Ajuste a seção de **busca**, o **menu da esquerda** e a **imagem de destaque**. Como eles são muito grandes, em mobile, é melhor renderizarmos um em cima do outro sem quebrar em colunas.

Vamos ocupar 100% da tela com o menu e a busca. A imagem de destaque deverá ser redimensionada para ocupar 100% da tela e não estourar o tamanho.

```
.busca,
.menu-departamentos,
```

```
.destaque img {
  margin-right: 0;
  width: 100%;
}
```

Teste esse passo no navegador redimensionado.

6. Nossa página está ficando boa em mobile. Falta apenas ajustarmos os painéis de destaques de produtos.

Por hora, eles estão com tamanhos absolutos ocupando metade da tela e mostrando 6 elementos, com 3 por linha. Vamos manter o painel com 3 elementos por linha, mas vamos fazer os dois painéis encaixarem um em cima do outro. Para isso, basta tirarmos a restrição de largura do painel para ele ocupar a tela toda.

```
.painel {
  width: auto;
}
```

Com relação aos produtos nos painéis. Vamos precisar redimensioná-los para encaixar 3 em cada linha. Uma boa maneira é colocar cada elemento com 30% do painel, totalizando 90%, e deixando espaço para as margens.

Já a imagem interna de cada produto deverá ocupar 100% do seu quadrado (o `<li>` que ajustamos), senão as imagens vão estourar o layout em certos tamanhos.

```
.painel li {
  width: 30%;
}
.painel img {
  width: 100%;
}
```

Teste a página final no navegador redimensionado. Temos nossa página mobile!

7. O que acontece em resoluções maiores de 320px? Nosso design volta ao padrão de 940px e ficamos com scroll horizontal. A maioria dos smartphones tem 320px de largura, mas nem todos, e nosso layout não se ajusta bem a esses outros. Até mesmo aqueles com 320px de largura, ao girar o aparelho em modo paisagem, a

resolução é maior (480px num iPhone e mais de 500px em muitos Androids).

O melhor era que nosso layout adaptável fosse usado não só em 320px mas em diversas resoluções intermediárias antes dos 940px que estabelecemos para o site Desktop.

Podemos ajustar nossa media query para aplicar o CSS de adaptação a qualquer tamanho de tela menor que os 940px do Desktop (ou seja, no máximo, 939px):

```
<link rel="stylesheet" href="css/mobile.css" media="(max-width: 939px)">
```

Teste, novamente, redimensionando o navegador para várias resoluções diferentes.

Repare que, como fizemos um **layout fluido**, baseado em porcentagens, os elementos se ajustam a diferentes resoluções sem esforço. É uma boa prática usar porcentagens e, sempre que possível, evitar o uso de valores absolutos em pixels.

8. (opcional) Se você fez os exercícios opcionais anteriores do controle de pause/play no banner, precisamos reposicionar o controle de pause/play:

```
.play,
.pause {
  top: auto;
  right: auto;

  left: 10px;
  bottom: 10px;
}
```


7.6 - Responsive Web Design

Repare o que fizemos nesse capítulo. Nossa página, com o mesmo HTML e

pequenos ajustes de CSS, suporta diversas resoluções diferentes, desde a pequena de um celular até um Desktop.

Essa prática é o que o mercado chama de **Web Design Responsivo**. O termo surgiu num famoso artigo de Ethan Marcotte e diz exatamente o que acabamos de praticar. São 3 os elementos de um design responsivo:

- layout fluído usando medidas flexíveis, como porcentagens;
- media queries para ajustes de design;
- uso de imagens flexíveis.

Nós aplicamos os 3 princípios na nossa adaptação da Home pra mobile. A ideia do responsivo é que a página se **adapte a diferentes condições**, em especial a diferentes resoluções. E, embora o uso de porcentagens exista há décadas na Web, foi a popularização das media queries que permitiram layouts verdadeiramente adaptativos.

7.7 - Mobile-first

Nosso exercício seguiu o processo que chamamos de *desktop-first*. Isso significa que tínhamos nossa página codificada para o layout Desktop e, num segundo momento, adicionamos as regras para adaptação a mobile.

Na prática, isso não é muito interessante. Repare como tivemos que **desfazer** algumas coisas do que tínhamos feito no nosso layout para desktop: tiramos alguns posicionamentos e desfizemos diversos ajustes na largura de elementos.

É muito mais comum e recomendado o uso da prática inversa: o **Mobile-first**. Isto é, começar o desenvolvimento pelo mobile e, depois, adicionar suporte a layouts desktop. No código, não há nenhum segredo: vamos apenas usar mais media queries **min-width** ao invés de **max-width**, mais comum em códigos desktop-first.

A grande mudança do mobile-first é que ela permite uma abordagem muito mais simples e evolutiva. Inicia-se o desenvolvimento pela área mais simples e limitada, com mais restrições, o mobile. O uso da tela pequena vai nos forçar a criar páginas mais simples, focadas e objetivas. Depois, a adaptação pra Desktop com media queries, é apenas uma questão de readaptar o layout.

A abordagem desktop-first começa pelo ambiente mais livre e vai tentando

cortar coisas quando chega no mobile. Esse tipo de adaptação é, na prática, muito mais trabalhosa.

Nós recomendamos o uso do mobile-first. E usaremos essa prática no curso a partir das próximas páginas, assim você poderá comparar os dois estilos.

7.8 - Exercícios opcionais: versão tablet

1. Nosso layout anterior tem dois comportamentos: um layout fixo em 940px otimizado para Desktops e outro construído para telas pequenas, mas que é aplicado para qualquer resolução abaixo de 939px.

Repare que, em resoluções altas (mas menores que 940px), nosso design mobile não fica tão bonito (embora continue funcional!). Podemos usar mais media queries para ajustar outros detalhes do layout conforme o tamanho da tela varia entre 320px e 939px.

Por exemplo, podemos usar 2 colunas no nosso menu quando chegarmos em 480px (um iPhone em paisagem):

```
@media (min-width: 480px) {
  header h1 {
    margin: 5px 0;
  }
  .menu-departamentos {
    padding-bottom: 10px;
    margin-bottom: 10px;
  }
  .menu-departamentos nav > ul {
    -webkit-column-count: 2;
       -moz-column-count: 2;
            column-count: 2;
  }
}
```

Em telas um pouco maiores, como tablets (um iPad tem 768px por exemplo), podemos querer fazer outros ajustes com uma media query:

```
@media (min-width: 720px) {

  header h1 {
    text-align: left;
  }

  .menu-opcoes {
    position: absolute;
  }
  .sacola {
```

```
    display: block;
  }
  .painel li {
    width: 15%;
  }
  .busca, .menu-departamentos {
    margin-right: 1%;
    width: 30%;
  }
  .menu-departamentos nav > ul {
    -webkit-column-count: 1;
       -moz-column-count: 1;
            column-count: 1;
  }
  .destaque img {
    width: 69%;
  }
}
```

Teste agora no navegador. Redimensione em diversos tamanhos desde o pequeno 320px até o Desktop grande. Veja a responsividade do nosso design, se ajustando a diversos tamanhos de tela.

2. Acesse sua página mobile no seu smartphone de verdade!

3. (trabalhoso) Adapte o layout das outras páginas (*Sobre*, *Contato* e *Produto*) para mobile, também. Faça uma solução mobile completa!

CAPÍTULO ANTERIOR:

CSS Avançado

PRÓXIMO CAPÍTULO:

[Introdução a PHP]

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter