CAPÍTULO 4

Mais HTML e CSS

"O medo é o pai da moralidade"
— Friedrich Wilhelm Nietzsche

4.1 - Analisando o miolo da página

Elaboramos o cabeçalho, mas ainda resta a área central e o rodapé. Focaremos agora nessa área central.

A área central possui duas áreas distintas: o bloco principal inicial, com o menu de navegação e o banner de destaque, e o bloco com os painéis com destaques de produtos.

Na área de navegação, teremos um formulário de busca, permitindo que o usuário busque por produtos.

4.2 - Formulários

Em HTML, temos um elemento chamado `<form>` criado para esta finalidade: capturar os dados do usuário e submetê-lo a algum serviço da internet.

Os dados são passados para o `<form>` por meio de tag `<input>`, que pode uma ter dupla finalidade: receber os dados digitados ou submeter o formulário.

É por meio da propriedade `type` que definimos essa finalidade. Em nosso caso, utilizaremos o tipo `search` para capturar os dados digitados e o tipo `image` para submeter o formulário. Existe também o tipo `submit`, que possui a mesma finalidade do `image`, mas é renderizado como um botão.

```
<form>
  <input type="search">
```

```
  <input type="image" src="img/busca.png" class="lupa">
</form>
```


4.3 - Posicionamento com float e clear

Em nosso layout, precisamos colocar o menu abaixo da busca e alinhado à esquerda com a imagem principal ao lado de ambos. Como conseguir este resultado? Uma solução seria utilizar `position` no menu, mas é algo que quebraria facilmente nosso layout caso a busca aumentasse de tamanho.

Em um dos nossos primeiros exercícios com a página `sobre.html`, colocamos a imagem da família Pelho à direita com a propriedade `float`, fazendo com que o elemento parágrafo a contornasse. Vamos tentar aplicar `float` à busca e ao menu para que ambos fiquem à esquerda, fazendo com que a imagem central os contorne:

```
.busca,
.menu-departamentos {
  float: left;
}
```

O resultado não foi o esperado. Para resolvermos este problema, precisaremos recorrer à propriedade **clear**.

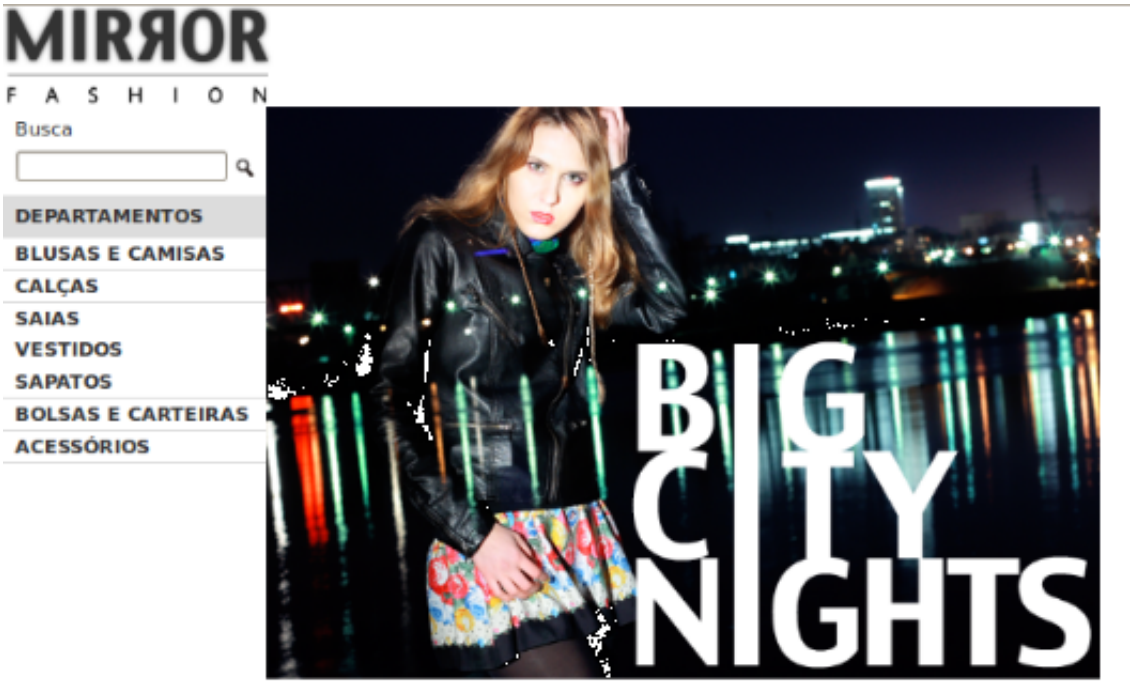
A propriedade clear

Existe uma propriedade que determina qual vai ser o comportamento de outros elementos que vêm ao redor daquele que a recebe e estão flutuando, e essa propriedade é a `clear`. A propriedade `clear` quer dizer "limpe o fluxo do documento ao meu lado" e pode receber os valores `left`, `right` ou `both`.

O valor `left` impede que elementos flutuantes fiquem à esquerda do elemento que recebe essa propriedade, o valor `right` impede que elementos flutuem à direita desse, e o valor `both` impede que elementos flutuem em ambos os lados do elemento. É importante sabermos que a propriedade `clear` de um elemento só interfere no layout da página caso outros elementos à sua volta estiverem flutuando.

Ao aplicarmos `clear:left` em nosso menu, ele não ficará ao lado da nossa busca com propriedade `float` e será renderizado na linha seguinte:

```
.busca,
.menu-departamentos {
  float: left;
}

.menu-departamentos {
  clear: left;
}
```



4.4 - Decoração de texto com CSS

O CSS permite ainda transformações e decorações de texto.

Transformação de texto

A propriedade `text-transform` permite realizar transformações em textos e seus possíveis valores são:

- **uppercase** - Todas as letras em maiúsculo;
- **lowercase** - Todas as letras em minúsculo;
- **capitalize** - Só as primeiras letras das palavras em maiúsculo.

Se quisermos colocar o texto em caixa alta:

```
.menu-departamentos {
  text-transform: uppercase;
}
```

Decoração de texto

Existe uma série de decorações que o navegador adiciona aos textos, dependendo das tags que utilizamos. A decoração mais comum é o sublinhado nos textos de links (tags <a> com valor para o atributo "href"). Existem outros tipos de decoração, como por exemplo, o texto contido na tag <del> (que serve para indicar um texto que foi removido de uma determinada versão do documento) é

exibido com uma linha bem no meio do texto.

É muito comum que em alguns casos seja desejável ocultar a linha inferior nos links, embora seja recomendado que links dentro de textos sejam decorados para destacarem-se do restante, facilitando a usabilidade e navegabilidade. No caso dos menus, onde temos uma área específica e isolada, normalmente estilizada e decorada o suficiente, normalmente podemos ocultar esse sublinhado, como no exemplo:

```
.item-menu {
  text-decoration: none;
}
```

Além do `none` (nenhuma decoração) ainda poderíamos ter configurado `underline` (com uma linha embaixo, o padrão dos links), `overline` (com uma linha em cima do texto), `line-through` (uma linha no meio do texto) e `blink` (o texto fica piscando na tela, o que não é muito recomendado).

4.5 – Cascata e herança no CSS

Algumas propriedades de elementos pais, quando alteradas, são aplicadas automaticamente para seus elementos filhos em cascata. Por exemplo:

```
<div id="pai">
  <h1>Sou um título</h1>
  <h2>Sou um subtítulo</h2>
</div>
```

```
#pai {
  color: blue;
}
```

No exemplo acima, todos os elementos filhos **herdaram** o valor da propriedade `color` do elemento pai a qual eles pertencem.

As propriedades que **não** são aplicadas em cascata em elementos filhos geralmente são aquelas que afetam diretamente a caixa (box) do elemento, como `width`, `height`, `margin` e `padding`.

```
h1 {
  padding-left: 40px;
}
#pai {
  color: blue;
  padding-left: 0;
}
```

Perceba que o `padding` do elemento `<h1>` não foi sobrescrito pelo valor do elemento pai `<div>`, ou seja, o valor `40px` foi mantido.

4.6 – Para saber mais: o valor inherit

Imagine que temos a seguinte divisão com uma imagem:

```
<div>
  <img src="box-model.png" alt="box model">
</div>

div {
  border: 2px solid;
  border-color: red;
  width: 30px;
  height: 30px;
}
```

Queremos que a imagem preencha todo o espaço da `<div>`, mas as propriedades `width` e `height` não são aplicadas em cascata, sendo assim, somos obrigados a definir o tamanho da imagem manualmente:

```
img {
  width: 30px;
  height: 30px;
}
```

Esta não é uma solução elegante, porque, se alterarmos o tamanho da `<div>`, teremos que lembrar de alterar também o tamanho da imagem. Uma forma de resolver este problema é utilizado o valor **inherit** para as propriedades `width` e `height` da imagem:

```
img {
  width: inherit;
  height: inherit;
```

```
}
```

O valor `inherit` indica para o elemento filho que ele deve utilizar o mesmo valor presente no elemento pai, sendo assim, toda vez que o tamanho do elemento pai for alterado, automaticamente o elemento filho herdará o novo valor, facilitando assim, a manutenção do código.

Lembre-se de que o `inherit` também afeta propriedades que não são aplicadas em cascata.

4.7 - Exercícios: menu e destaque

1. Vamos criar um elemento destaque e, dentro dele, uma `section` para busca e outra para o menu:

```
<div class="container destaque">

  <section class="busca">
    <h2>Busca</h2>

    <form>
      <input type="search">
      <input type="image" src="img/busca.png">
    </form>
  </section><!-- fim .busca -->

  <section class="menu-departamentos">
    <h2>Departamentos</h2>

    <nav>
      <ul>
        <li><a href="#">Blusas e Camisas</a></li>
        <li><a href="#">Calças</a></li>
        <li><a href="#">Saias</a></li>
        <li><a href="#">Vestidos</a></li>
        <li><a href="#">Sapatos</a></li>
        <li><a href="#">Bolsas e Carteiras</a></li>
        <li><a href="#">Acessórios</a></li>
      </ul>
    </nav>
  </section><!-- fim .menu-departamentos -->

  <img src="img/destaque-home.png" alt="Promoção: Big City Night">
</div><!-- fim .container .destaque -->
```

Repare como já usamos diversas classes no HTML para depois selecionarmos os elementos via CSS.

2. Aplique o estilo visual do design com CSS. Queremos um fundo cinza nas caixas

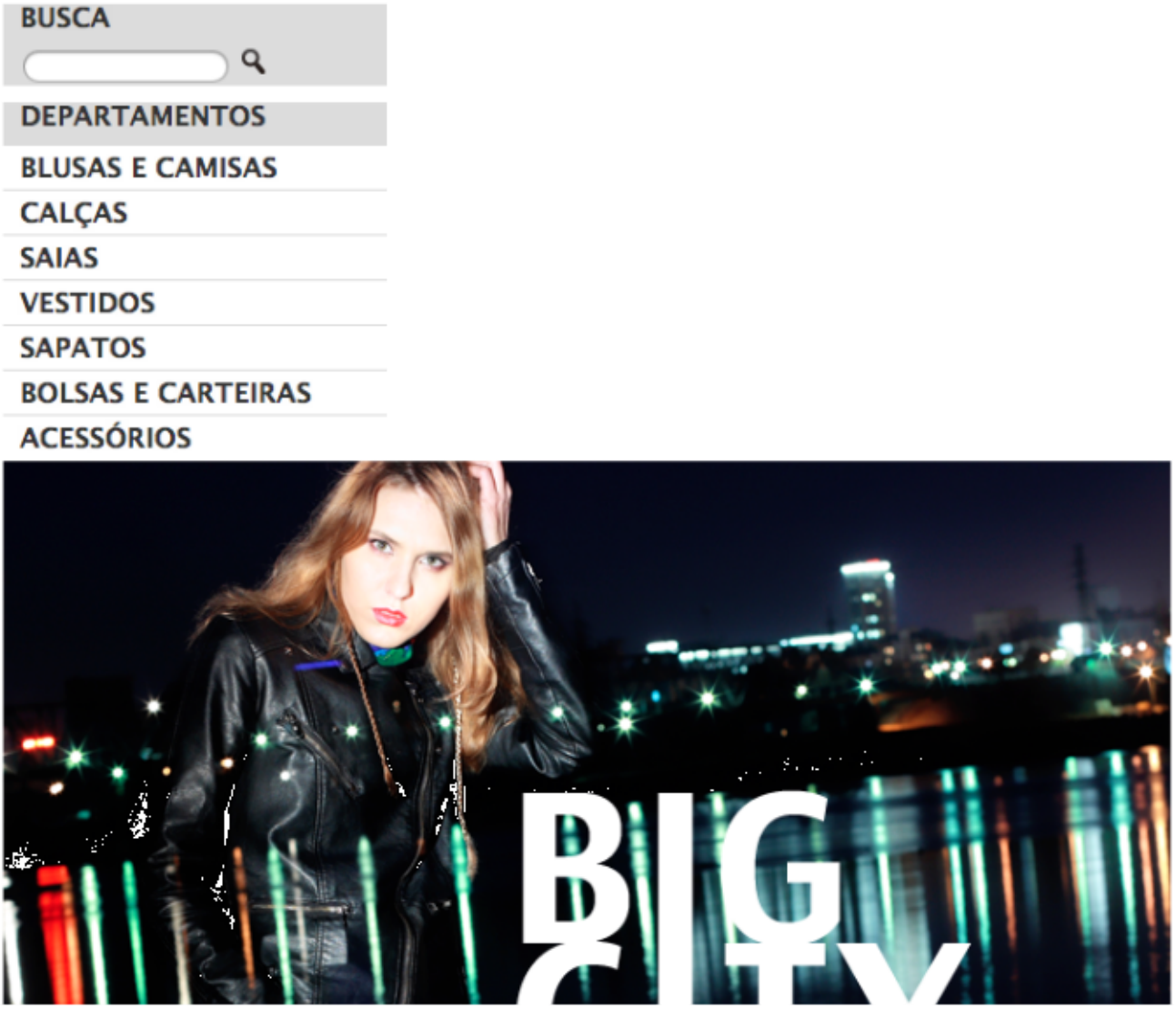
de busca e no menu de departamentos. Além disso, o texto deve estar em negrito e apresentado em maiúsculas. Aplicaremos também algumas regras de tamanhos e margens.

```
.busca,
.menu-departamentos {
  background-color: #dcdcdc;
  font-weight: bold;
  text-transform: uppercase;

  margin-right: 10px;
  width: 230px;
}

.busca h2,
.busca form,
.menu-departamentos h2 {
  margin: 10px;
}

.menu-departamentos li {
  background-color: white;
  margin-bottom: 1px;
  padding: 5px 10px;
}

.menu-departamentos a {
  color: #333333;
  text-decoration: none;
}
```

3. Na busca, use a propriedade `vertical-align` para alinhar o campo de texto à imagem da lupa pelo centro. Aproveite e coloque o tamanho do campo de texto para melhor encaixar no design e use seletores de atributo do CSS para isso (veremos mais desses seletores depois no curso).

```
.busca input {
  vertical-align: middle;
}

.busca input[type=search] {
  width: 170px;
}
```

Teste a página no navegador e veja como o design está quase pronto, apenas o posicionamento dos elementos precisa ser acertado.



4. Para que o menu de departamentos e a busca estejam à esquerda na página, iremos **flutuar** esses elementos com `float:left`.

Mas só isso fará com que o menu fique a direita da busca (faça o teste). Precisamos indicar ao `menu-departamentos` que ele deve flutuar à esquerda mas não ao lado de outro elemento. Conseguimos isso com a propriedade `clear`.

```
.busca,
.menu-departamentos {
  float: left;
}

.menu-departamentos {
    clear: left;
}
```

Observe novamente a página no navegador e veja como o posicionamento está correto.

Repare que o CSS usado foi bastante curto e simples. Mas o conceito do `float` e do `clear` é difícil e complexo. Esteja certo de ter compreendido o porquê do uso dessas propriedades no exercício antes de prosseguir o curso!

5. Mais acertos de design. Acerte as margens e posicionamentos no menu lateral e no topo:

```
.destaque {
  margin-top: 10px;
}
.menu-departamentos {
  margin-top: 10px;
  padding-bottom: 10px;
}
```

Teste o resultado.

4.8 - Display inline-block

Precisamos criar um painel com uma lista de novidades, onde cada produto será representado por uma `<li>`. Já sabemos que por padrão uma `<li>` possui a propriedade `display:block`, mas queremos os produtos lado a lado. Podemos trocar este comportamento mudando a propriedade `display` dos elementos para `inline`.

Também será necessário alterar as propriedades `width`, `margin` e `padding` das `<li>`, mas agora os elementos `<li>` são `inline` e este modo de exibição ignora alterações que afetam as propriedades da *box*. Como resolver este problema?

Os navegadores mais modernos introduzem um modelo de exibição que é a mistura dos dois, o `inline-block`. Os elementos que recebem o valor `inline-block` para a propriedade `display` obedecem às especificações de dimensão das propriedades `height` (altura) e `width` (largura) e ainda permitem que outros elementos sejam exibidos ao seu lado como elementos `inline`.

```
.painel li {
  display: inline-block;
  vertical-align: top;
  width: 140px;
  margin: 2px;
  padding-bottom: 10px;
}
```

Como o inline-block faz os elementos se alinharem como numa linha de texto, podemos controlar o alinhamento vertical dessa linha da mesma forma que fizemos antes com linhas de texto e imagens simples. Isto é, usando a propriedade `vertical-align` que, nesse caso, recebeu valor `top`.

Isso indica que, se tivermos vários produtos de tamanhos diferentes, eles vão se alinhar pelo topo.

4.9 - Exercícios: painéis flutuantes

1. Vamos criar agora nosso painel de novidades. Crie um elemento `<div>` para conter os **dois painéis de produtos**. Ele deve receber a classe **container**, para se alinhar ao meio da tela, e a classe **paineis** que usaremos depois no CSS.

```
<div class="container paineis">
  <!-- os paineis de novidades e mais vendidos entrarão aqui dentro -->
</div>
```

2. **Dentro da div criada acima**, criaremos uma nova `<section>` para cada painel. A primeira, receberá as classes **painel** e **novidades**. Ela conterá o título em um `<h2>` e uma lista ordenada (`<ol>`) de produtos.

Cada produto deve ser representado como um item na lista (`<li>`) com um link para o produto e sua imagem (representado por `figure`, `figcaption` e `img`).

Veja o exemplo com um produto. **Ele deve ser incluído dentro da section que você acabou de criar**:

```
<section class="painel novidades">
    <h2>Novidades</h2>
    <ol>

      <!-- primeiro produto -->
      <li>
        <a href="produto.html">
          <figure>
            <img src="img/produtos/miniatura1.png">
            <figcaption>Fuzz Cardigan por R$ 129,90</figcaption>
          </figure>
        </a>
      </li>


      <!-- coloque mais produtos aqui! -->
    </ol>
  </section>
```

Crie o HTML desse painel e o preencha com vários produtos (6 é um bom número). Lembre-se de que cada produto está na sua própria `li` com link e imagem próprios. Na pasta **img/produtos** do seu projeto, você encontra várias imagens **miniaturaX.png** que podem ser usadas para criar produtos diferentes.

3. Crie um segundo painel, para representar os produtos mais vendidos. Esse painel deve ficar **após** o fechamento do painel anterior, mas ainda **dentro** da div `paineis`.

O novo painel deve receber as classes **painel** e **mais-vendidos**. Sua estrutura é idêntica ao do exercício anterior (dica: copie o código para evitar refazer tudo de novo).

```
<section class="painel mais-vendidos">
  <h2>Mais Vendidos</h2>
  <ol>

    <!-- coloque vários produtos aqui -->

  </ol>
```

```
</section>
```

Nosso HTML já está ficando grande e complexo, como é uma página real cheia de conteúdo. Cuidado para não se confundir na posição das tags. Recapitulando essa parte dos painéis, a estrutura deve estar assim:

- **div**: container paineis
  - **section**: painel novidades
    - **h2**: título Novidades
    - **ol**: lista de produtos
      - vários `li`s com produtos (e links e imagens dentro de cada um)
  - **section**: painel mais-vendidos
    - **h2**: título Mais Vendidos
    - **ol**: lista de produtos
      - vários `li`s com produtos (e links e imagens dentro de cada um)

4. Vamos posicionar nossos painéis para ficarem de acordo com o design.

O painel de `novidades` deve flutuar à esquerda e o `mais-vendidos`, à direita. Cada um deve ocupar `445px` (pouco menos da metade dos 940px), assim um ficará ao lado do outro:

```
.painel {
  margin: 10px 0;
  padding: 10px;
  width: 445px;
}

.novidades {
  float: left;
}

.mais-vendidos {
  float: right;
}
```

Próximo passo: os itens dos produtos dentro da lista de cada painel. Queremos que sejam dispostos lado a lado mas com certo tamanho e espaçamento para alinhamento. Conseguimos isso colocando `display:inline-block` nos elementos da lista e, para alinhar os produtos pelo topo, com `vertical-align:top`.

```
.painel li {
  display: inline-block;
  vertical-align: top;
  width: 140px;

  margin: 2px;
  padding-bottom: 10px;
}
```

O posicionamento em si deve estar certo. Mas falta umas regras para estilo, como tamanho dos títulos e cores de texto e fundo.

```
.painel h2 {
  font-size: 24px;
  font-weight: bold;
  text-transform: uppercase;

  margin-bottom: 10px;
}

.painel a {
  color: #333;
  font-size: 14px;
  text-align: center;
  text-decoration: none;
}

.novidades {
  background-color: #f5dcdc;
}

.mais-vendidos {
  background-color: #dcdcf5;
}
```

Teste a página no navegador e veja o resultado final!

4.10 – Seletores de atributo do CSS3

Além dos seletores de tag, classe e id que observamos anteriormente, existe mais uma série de seletores avançados do CSS.

Um dos seletores CSS mais versáteis é o seletor de atributo, com ele podemos verificar a presença ou valor de um atributo para selecioná-lo. Por exemplo:

```
input[value] {
  color: #cc0000;
}
```

O seletor acima age em todos os elementos da tag `<input>` que têm o atributo "value". Também é possível verificar se o atributo tem um valor específico:

```
input[type="text"] {
  border-radius: 4px;
}
```

Além de verificar um valor integralmente, é possível utilizar alguns operadores para selecionar valores em determinadas condições, como por exemplo o seletor de atributo com prefixo:

```
div[class|="menu"] {
  border-radius: 4px;
}
```

O seletor acima vai agir em todas as tags `<div>` cujo atributo "class" comece com a palavra **menu** seguida de um hífen e qualquer outro valor na sequência, como por exemplo **menu-principal**, **menu-departamentos** e **menu-teste**.

Também é possível buscar por uma palavra específica no valor, não importando o valor completo do atributo. Por exemplo:

```
input[value~="problema"] {
  color: #cc0000;
}
```

O seletor acima agirá sobre todos os elementos da tag `<input>` que contiverem a palavra "problema" em seu conteúdo.

Com o CSS3 é possível utilizar novos operadores com sinais que se assemelham aos das expressões regulares:

```
/* busca por inputs com valor de "name" iniciando em "usuario" */
input[name^="usuario"] {
  color: 99ffcc;
```

```
}

/* busca por inputs com valor de "name" terminando em "teste" */
input[name$="teste"] {
  background-color: #ccff00;
}

/* busca por inputs com valor de "nome" contendo "tela" em qualquer posição
*/
input[name*="tela"] {
  color: #666666;
}
```

Os seletores de atributo têm o mesmo valor de especificidade dos seletores de classe.

4.11 – Rodapé

Para finalizarmos a página, precisamos desenvolver o rodapé. Visualmente, ele é bastante simples. Mas há algumas questões importantes a serem salientadas.

Semântica

No HTML5, a tag apropriada para rodapés é a `<footer>`, que vamos usar no exercício. Além disso, nosso rodapé ainda tem mais 2 conteúdos: o logo em negativo do lado esquerdo e ícones de acesso a redes sociais do lado direito. Que elementos usar?

O logo no lado esquerdo é uma simples imagem:

```
<img src="img/logo-rodape.png" alt="Logo Mirror Fashion">
```

Já a lista de ícones das redes sociais, na verdade, é uma lista de links. Os ícones são meramente ilustrativos. Em um leitor de tela, vamos querer que um link seja lido para o usuário, independentemente do seu ícone gráfico.

Podemos usar então uma simples lista com `<a>`:

```
<ul class="social">
  <li><a href="http://facebook.com/mirrorfashion">Facebook</a></li>
  <li><a href="http://twitter.com/mirrorfashion">Twitter</a></li>
  <li><a href="http://plus.google.com/mirrorfashion">Google+</a></li>
</ul>
```

Esse é um ponto importante para entendermos a diferença entre marcação semântica e apresentação visual. Repare que criamos uma estrutura no HTML com

conteúdo completamente diferente do resultado final visual. Vamos cuidar do visual depois no CSS.

4.12 - Substituição por Imagem

Um truque muito usado em CSS é o chamado **Image Replacement** -- ou *substituição por imagem*. Serve para, usando técnicas de CSS, exibir uma imagem em algum elemento que originalmente foi feito com texto. Perfeito para nosso caso dos ícones das redes sociais.

A ideia básica é:

- Acertar o tamanho do elemento para ficar igual ao da imagem;
- Colocar a imagem como `background` do elemento;
- Esconder o texto.

Para esconder o texto, é comum usar a tática de colocar um `text-indent` negativo bastante alto. Isso, na prática, faz o texto ser renderizado "fora da tela".

```
.facebook {
  /* tamanho do elemento = imagem */
  height: 55px;
  width: 85px;

  /* imagem como fundo */
  background-image: url(../img/facebook.png);

  /* retirando o texto da frente */
  text-indent: -9999px;
}
```

4.13 – Estilização e posicionamento do rodapé

Container interno

Repare que o rodapé, diferentemente de todos os elementos do layout, ocupa 100% da largura da página. Ele não é restrito ao tamanho de 940px do miolo do nosso site. Isso porque o rodapé tem um `background` que se repete até os cantos.

Mas repare que o conteúdo dele é limitado aos 940px e centralizado junto com o resto da página -- onde estávamos usando a `class container`.

O que precisamos fazer então é ter o `<footer>` com 100% além de uma tag interna pra o conteúdo do rodapé em si, e essa tag será o `container`:

```
<footer>
  <div class="container">
    ....
  </div>
</footer>
```

Posicionamento

Ao colocar o rodapé, você perceberá que ele subirá na página ao invés de ficar em baixo. Isso porque os elementos anteriores a ele, os painéis de destaque, estão flutuando na página e, portanto, saíram do fluxo de renderização. Para corrigir isso, basta usar a propriedade `clear: both` no `footer`.

Dentro do rodapé em si, queremos que a lista de ícones seja colocada à direita. Podemos acertar isso com **posicionamento absoluto**, desde que o container do rodapé esteja *posicionado* (basta dar um `position:relative` a ele).

Já os itens dentro da lista (os 3 links), devem ser flutuados lado a lado (e não um em cima do outro). É fácil fazer com `float:left` no `li`.

Estilização

O rodapé em si terá um `background-image` com o fundo preto estampado repetido infinitamente.

Os elementos internos são todos ícones a serem substituídos por imagens via CSS com *image replacement*.

E, para saber qual ícone atribuir a qual link da lista de mídias sociais, podemos

usar seletores de atributo do CSS3:

```
.social a[href*="facebook.com"] {
  background-image: url(../img/facebook.png);
}
```

4.14 – Exercícios: rodapé

1. Vamos finalizar nossa página com o rodapé do layout. Crie uma estrutura semântica no HTML usando a tag `<footer>` e tags `<img>`, `<ul>`, `<li>` e `<a>` para o conteúdo.

Atenção especial para a necessidade de um elemento `container` **dentro** do rodapé para alinhar seu conteúdo com o restante da página.

```
<footer>
  <div class="container">
    <img src="img/logo-rodape.png" alt="Logo Mirror Fashion">

    <ul class="social">
      <li><a href="http://facebook.com/mirrorfashion">Facebook</a></li>
      <li><a href="http://twitter.com/mirrorfashion">Twitter</a></li>
      <li><a href="http://plus.google.com/mirrorfashion">Google+</a></li>
    </ul>

  </div>
</footer>
```

Teste no seu navegador e veja o resultado sem estilo, mas utilizável.

2. Vamos estilizar o conteúdo visual. Coloque o `background` preto no rodapé e faça as substituições de imagens. Use *seletores de atributo* do CSS3 para identificar os ícones de cada rede social.

```
footer {
  background-image: url(../img/fundo-rodape.png);
}

.social li a {
  /* tamanho da imagem */
  height: 32px;
  width: 32px;

  /* image replacement */
  display: block;
  text-indent: -9999px;
}

.social a[href*="facebook.com"] {
  background-image: url(../img/facebook.png);
```

```
}

.social a[href*="twitter.com"] {
  background-image: url(../img/twitter.png);
}

.social a[href*="plus.google.com"] {
  background-image: url(../img/googleplus.png);
}
```

Teste no navegador. O que aconteceu?

O rodapé subiu na página porque os elementos anteriores (os painéis de destaque) estão flutuando. É fácil arrumar, basta adicionar a regra de `clear` no `footer`:

```
footer {
  clear: both;
}
```

Teste novamente. O rodapé voltou para o lugar certo?

3. Último passo: posicionar os elementos internos do rodapé apropriadamente.

Vamos posicionar os ícones sociais absolutamente à direita com `position:absolute`. Para isso, o container do nosso rodapé precisa estar posicionado. Aproveite também e coloque um espaçamento interno:

```
footer {
  padding: 20px 0;
}

footer .container {
  position: relative;
}

.social {
  position: absolute;
  top: 12px;
  right: 0;
}
```

Por fim, precisamos fazer os ícones das redes sociais fluturarem lado a lado horizontalmente:

```
.social li {
  float: left;
  margin-left: 25px;
}
```

Teste no navegador. Como está o resultado final? De acordo com o que o designer queria?

4.15 - Para saber mais: suporte HTML5 no Internet Explorer antigo

A partir do IE9, há um bom suporte a HTML5, até para elementos avançados como os de multimídia. Entretanto, até o IE8 (incluindo as versões 6 e 7), as tags do HTML5 não são reconhecidas.

Se você abrir nossa página no IE8, verá o design todo quebrado pois as tags de header, footer, nav, section etc são ignoradas. Mas é possível corrigir o suporte a esses novos elementos semânticos.

A solução envolve um hack descoberto no IE e chamado de **html5shiv**. Há um projeto opensource com o código disponível para download:

http://code.google.com/p/html5shiv/

Para incluir a correção na nossa página, basta adicionar no header:

```
<!--[if lt IE 9]>
  <script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Repare que isso carrega um JavaScript que acionará o hack. Mas a tag `script` está dentro de um bloco com uma espécie de if dentro de um comentário!

Esse recurso do IE é chamado de **comentários condicionais** e nos permite adicionar código que somente será lido pelo IE -- uma excelente solução para resolver seus bugs e inconsistências sem estragar a página nos demais

navegadores.

Nesse caso, estamos dizendo que o tal script com o hack só deve ser carregado pelas versões anteriores ao IE9; já que, a partir desta versão, há suporte nativo a HTML5 e não precisa de hacks.

IE6 e IE7

Ao testar nesses navegadores muito antigos, você verá que apenas o HTML5shiv não é suficiente. Na verdade, vários recursos e técnicas que usamos no nosso CSS não eram suportados nas versões antigas.

Felizmente, o uso de IE6 e IE7 no Brasil é bastante baixo e cai cada vez mais - hoje já é menos de 1% dos usuários. Na Caelum, já não suportamos mais essas versões em nosso curso e nem em nossos sites.

4.16 - Exercícios opcionais

1. Porte nosso rodapé para a página "Sobre" do capítulo anterior.

2. Nossa página "Sobre" foi construída sem muita preocupação semântica. No HTML5, há novas tags com objetivo específico de lidar com conteúdos textuais divididos em partes, com subtítulos etc.

Podem ser artigos de um jornal, um livro online ou mesmo um texto descrevendo nossa empresa - como nossa página Sobre faz.

Podemos representar o texto todo com `<article>` e suas seções com `<section>`. Use essas novas tags na **sobre.html** para termos uma marcação mais semântica.

3. Adicione suporte ao IE8 na nossa página usando o HTML5shiv.

CAPÍTULO ANTERIOR:

HTML semântico e posicionamento no CSS

PRÓXIMO CAPÍTULO:

[JavaScript e interatividade na Web]

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter