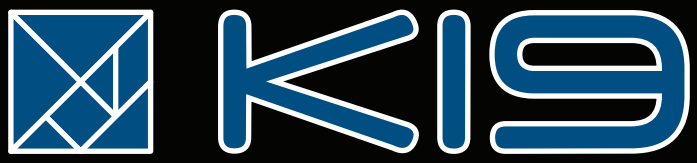

TREINAMENTOS

Desenvolvimento Web com HTML, CSS e Javascript

Desenvolvimento Web com HTML, CSS e Javascript

9 de abril de 2013

Sobre a K19

A K19 é uma empresa especializada na capacitação de desenvolvedores de software. Sua equipe é composta por profissionais formados em Ciência da Computação pela Universidade de São Paulo (USP) e que possuem vasta experiência em treinamento de profissionais para área de TI.

O principal objetivo da K19 é oferecer treinamentos de máxima qualidade e relacionados às principais tecnologias utilizadas pelas empresas. Através desses treinamentos, seus alunos se tornam capacitados para atuar no mercado de trabalho.

Visando a máxima qualidade, a K19 mantém as suas apostilas em constante renovação e melhoria, oferece instalações físicas apropriadas para o ensino e seus instrutores estão sempre atualizados didática e tecnicamente.

Seguro Treinamento

Na K19 o aluno faz o curso quantas vezes quiser!

Comprometida com o aprendizado e com a satisfação dos seus alunos, a K19 é a única que possui o Seguro Treinamento. Ao contratar um curso, o aluno poderá refazê-lo quantas vezes desejar mediante a disponibilidade de vagas e pagamento da franquia do Seguro Treinamento.

As vagas não preenchidas até um dia antes do início de uma turma da K19 serão destinadas ao alunos que desejam utilizar o Seguro Treinamento. O valor da franquia para utilizar o Seguro Treinamento é 10% do valor total do curso.

Termo de Uso

Termo de Uso

Todo o conteúdo desta apostila é propriedade da K19 Treinamentos. A apostila pode ser utilizada livremente para estudo pessoal . Além disso, este material didático pode ser utilizado como material de apoio em cursos de ensino superior desde que a instituição correspondente seja reconhecida pelo MEC (Ministério da Educação) e que a K19 seja citada explicitamente como proprietária do material.

É proibida qualquer utilização desse material que não se enquadre nas condições acima sem o prévio consentimento formal, por escrito, da K19 Treinamentos. O uso indevido está sujeito às medidas legais cabíveis.

Conheça os nossos cursos

K01- Lógica de Programação

K02 - Desenvolvimento Web com HTML, CSS e JavaScript

K03 - SQL e Modelo Relacional

K11 - Orientação a Objetos em Java

K12 - Desenvolvimento Web com JSF2 e JPA2

K21 - Persistência com JPA2 e Hibernate

K22 - Desenvolvimento Web Avançado com JFS2, EJB3.1 e CDI

K23 - Integração de Sistemas com Webservices, JMS e EJB

K41 - Desenvolvimento Mobile com Android

K51 - Design Patterns em Java

K52 - Desenvolvimento Web com Struts

K31 - C# e Orientação a Objetos

K32 - Desenvolvimento Web com ASP.NET MVC

www.k19.com.br/cursos

CAPÍTULO 1

INTRODUÇÃO

Durante muito tempo, a ideia de desenvolvimento web ficou associada apenas a construção de páginas que possuíam somente o intuito de oferecer aos usuários o acesso a um determinado conteúdo. Porém, com a popularização da internet, novas necessidades foram surgindo em diversas áreas.

Na área de entretenimento, cada vez mais jogos online foram aparecendo. Diversas redes sociais ganharam forças graças à grande interatividade permitida entre os usuários. Gravadoras de música passaram a vender seus títulos através de canais especializados. No meio corporativo, as empresas passaram a adotar sistemas web para controlar as suas tarefas administrativas. Enfim, necessidades antes inexistentes surgiram em uma velocidade muito grande. Muitos sites deixaram de ser simples páginas para se tornarem verdadeiras aplicações.

Há cerca de 15 anos, era muito comum que um único desenvolvedor fosse responsável por todo o desenvolvimento de uma aplicação web. Essa pessoa era chamada de webmaster. Com o passar do tempo, o papel do webmaster como era conhecido foi desaparecendo. A complexidade e volume de trabalho para o desenvolvimento de uma aplicação web se tornaram muito grande para apenas uma pessoa ou até mesmo para um grupo pequeno de desenvolvedores (webmasters).

Hoje, a função de webmaster ainda existe mas com um papel um pouco diferente. Geralmente, esse profissional apesar de possuir bons conhecimentos nas diversas tecnologias utilizadas apenas gerencia o desenvolvimento que realizado por outros profissionais.

Como as tarefas antes de responsabilidade do webmaster foram delegadas a outros profissionais, naturalmente, apareceram algumas especializações. Essas especializações podem ser classificadas em dois grupos: desenvolvedores front-end e back-end. Em geral os desenvolvedores front-end são responsáveis pela interface com a qual os usuários interagem enquanto os desenvolvedores back-end são responsáveis pelo funcionamento interno das aplicações.

Client Side e Server Side

Os usuários acessam a interface de uma aplicação web através de navegadores (browsers). Os desenvolvedores front-end devem conhecer bem o funcionamento dos navegadores e das tecnologias e linguagens relacionadas a eles. Essas tecnologias e linguagens são categorizadas como **client side**. Atualmente, as principais linguagens e tecnologias client side são HTML, CSS, Javascript, Adobe Flash, Microsoft Silverlight e VBScript.

Por outro lado, os desenvolvedores back-end trabalham com linguagens como Java, C#, VB.NET, PHP, Ruby, Python, SQL entre outras. Essas linguagens atuam do lado do servidor por isso são classificadas como **server side**. Isso não significa que os desenvolvedores front-end não precisam conhecer as linguagens utilizadas pelo back-end e vice-versa. Na prática, ocorre uma especialização dos profissionais em determinadas tecnologias que podem tender mais para o front-end ou para o

back-end.

HTML, CSS e Javascript

Como acabamos de ver, as principais linguagens e tecnologias client side são HTML, CSS, Javascript, Adobe Flash, Microsoft Silverlight e VBScript. De todas elas as três primeiras são as mais importantes e atualmente estão em maior evidência. Cada uma das três linguagens possui um papel bem específico que podemos resumir da seguinte maneira: o código HTML será responsável por prover o conteúdo de uma página, o código CSS cuidará da formatação visual do conteúdo apresentado e o código Javascript permitirá que as páginas possuam algum tipo de comportamento ("inteligência") e que alguma interação possa ser estabelecida com os usuários.

Nos próximos capítulos, discutiremos em detalhes cada uma dessas três linguagens.

HTML

CAPÍTULO 2

Quando acessamos uma página web, estamos interessados na informação contida nessa página. Essa informação pode estar na forma de texto, imagem ou vídeo. Em geral, o conteúdo de uma página web é definido com a linguagem HTML (HyperText Markup Language). HTML é uma linguagem de marcação originalmente proposta por Tim Berners-Lee no final da década de 1980. O objetivo do Tim Barners-Lee era criar um mecanismo simples que pudesse ser utilizado por qualquer pessoa que quisesse disseminar documentos científicos.

Desde sua proposta até os dias de hoje, a linguagem HTML sofreu diversas alterações. A cada versão, novos recursos são adicionados e problemas corrigidos. A versão mais atual da especificação da linguagem HTML é a 5. Essa versão ainda não foi finalizada, ela está na fase de "trabalho em progresso" (working draft). Porém, já existem navegadores implementando alguns dos novos recursos do HTML5.

As especificações da linguagem HTML são publicadas pelo World Wide Web Consortium mais conhecido por sua sigla W3C. Além do HTML, o W3C também é responsável por linguagens como o XML, o SVG e pela interface DOM (Document Object Model), por exemplo.

Estrutura Básica

Um documento HTML é composto por elementos que possuem uma tag, atributos, valores e possivelmente filhos que podem ser um texto simples ou outros elementos. Cada elemento deve obrigatoriamente possuir uma tag e ela deve ser definida entre parênteses angulares (< e >). Veja o exemplo:

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Exemplo da estrutura básica de um documento HTML</title>
5   </head>
6   <body>
7     <p>Olá mundo!</p>
8   </body>
9 </html>
```

Código HTML 2.1: Exemplo da estrutura básica de um documento HTML

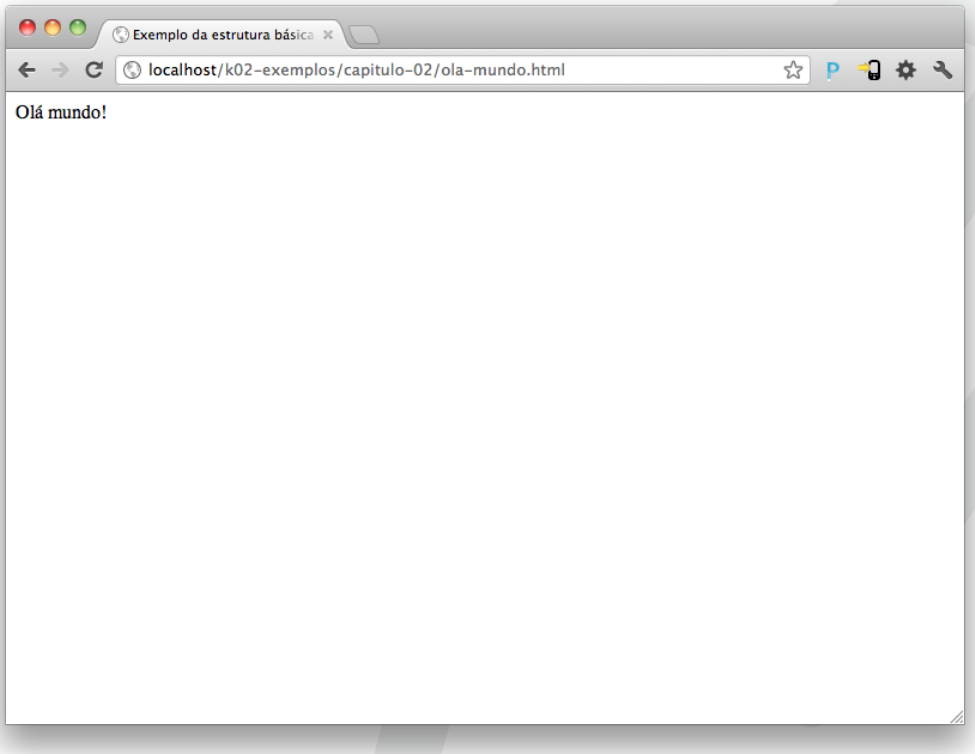

Figura 2.1: Exemplo da estrutura básica de um documento HTML

No exemplo acima, temos um elemento HTML representado pela tag “p” e um texto simples “Olá Mundo!” filho desse elemento.

Exercícios de Fixação

1 Na pasta **Desktop** do seu usuário, crie uma nova pasta com o seu primeiro nome. Dentro dessa pasta, crie um diretório chamado **html**. Para facilitar, utilize apenas letras minúsculas em todas as pastas e arquivos que criarmos durante o curso.

2 Agora, utilizando um editor de texto, crie um novo arquivo chamado *ola-mundo.html* e salve-o dentro da pasta *html*. Em seguida, insira o seguinte código dentro do arquivo *ola-mundo.html*:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo da estrutura básica de um documento HTML</title>
  </head>
  <body>
    <p>Olá mundo!</p>
  </body>
</html>
```

Código HTML 2.2: ola-mundo.html

Abra o arquivo *ola-mundo.html* em um navegador e veja o resultado.

Exercícios Complementares

1. Crie uma página HTML que exiba o nome deste curso duas vezes.

Semântica HTML

De acordo com a especificação da linguagem HTML, cada elemento possui um propósito bem definido. Para o funcionamento correto das páginas de uma aplicação web, é fundamental respeitar o propósito de cada elemento e utilizá-lo nas condições corretas. Muitos autores utilizam o termo semântica HTML ao se referirem ao uso correto dos elementos da linguagem HTML. Por exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso correto da semântica HTML</title>
  </head>
  <body>
    <p>Este é um texto para mostrar o significado da tag p.</p>
  </body>
</html>
```

Código HTML 2.4: Exemplo de uso correto da semântica HTML

No exemplo acima, utilizamos o elemento p para definir um parágrafo. De acordo com a especificação da linguagem HTML, esse elemento deve ser utilizado justamente para definir parágrafos. Portanto, ele foi aplicado corretamente. Agora, vejamos outro exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Meus amigos - Site do Jonas - Criado pelo Jonas</title>
  </head>
  <body>
    <address>
      Rafael Cosentino
      rafael.cosentino@k19.com.br
      Sócio fundador da K19 Treinamentos
      Av. Brigadeiro Faria Lima, 1571 - Jardim Paulistano - São Paulo, SP
      CEP 01452-001
    </address>

    <address>
      Marcelo Martins
      marcelo.martins@k19.com.br
      Sócio fundador da K19 Treinamentos
      Av. Brigadeiro Faria Lima, 1571 - Jardim Paulistano - São Paulo, SP
      CEP 01452-001
    </address>
  </body>
</html>
```

Código HTML 2.5: Exemplo de uso incorreto da semântica HTML

Dessa vez, utilizamos o elemento address. De acordo com a especificação, o elemento address

deve ser utilizado para fornecer informações de contato dos autores do documento ou da maior parte do documento. Normalmente, esse elemento aparece no início ou no final das páginas.

Se observarmos o exemplo mais atentamente, trata-se de uma página do site do Jonas (repare no título da página). O autor da página é o Jonas e não o Rafael ou o Marcelo. Portanto, o elemento `address` foi aplicado incorretamente. Além disso, devemos evitar o uso desse elemento para informar endereços postais a menos que essas informações sejam relevantes para o documento.

Parágrafos

Os parágrafos dentro de um documento HTML, em geral, são definidos através do elemento `p`. Uma das principais características desse elemento é que ele ocupa horizontalmente todo o espaço definido pelo elemento pai. Esse é o comportamento dos **elementos de bloco** que discutiremos com mais detalhes posteriormente.

Por enquanto, o importante é sabermos que, devido ao comportamento de bloco do elemento `p`, o navegador ajustará o texto do parágrafo à largura do elemento pai realizando todas as quebras de linha necessárias. Caso seja necessário forçar uma quebra de linha, podemos utilizar a elemento `br`. Confira o exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de quebra de linha em um parágrafo</title>
  </head>
  <body>
    <p>Um texto bem longo. Longo mesmo! Este parágrafo serve para demonstrar
    o comportamento da quebra de linha automática, ou seja, sem utilizar
    nenhum recurso para que a quebra ocorra.</p>

    <p>Já este parágrafo demonstra a quebra de linha forçada.<br/>Percebeu?</p>
  </body>
</html>
```

Código HTML 2.6: Exemplo de quebra de linha em um parágrafo

Figura 2.2: Exemplo de quebra de linha em um parágrafo

Exercícios de Fixação

3 Crie um novo documento HTML chamado **p-quebra-de-linha.html** na pasta **html**. Em seguida, abra esse arquivo em um navegador (se necessário, redimensione a janela do navegador para verificar o comportamento da quebra de linha).

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de quebra de linha em um parágrafo</title>
  </head>
  <body>
    <p>Um texto bem longo. Longo mesmo! Este parágrafo serve para demonstrar
    o comportamento da quebra de linha automática, ou seja, sem utilizar
    nenhum recurso para que a quebra ocorra.</p>

    <p>Já este parágrafo demonstra a quebra de linha forçada.<br/>Percebeu?</p>
  </body>
</html>
```

Código HTML 2.7: p-quebra-de-linha.html

Exercícios Complementares

2 Crie um documento HTML e force uma quebra de linha em qualquer parte de um parágrafo. Dica: utilize o site www.lipsum.com para gerar um texto fictício.

Cabeçalhos

Assim como em um livro, uma página HTML pode conter uma hierarquia de títulos para estabelecer uma divisão de seu conteúdo. Para conseguirmos realizar essa tarefa devemos utilizar as tags de cabeçalho h1, h2, h3, h4, h5 e h6.

Os sufixos numéricos de 1 a 6 indicam o nível do título dentro da hierarquia de títulos do documento. Veja o exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de cabeçalhos</title>
  </head>
  <body>
    <h1>Título 1</h1>
    <h2>Título 2</h2>
    <h3>Título 3</h3>
    <h4>Título 4</h4>
    <h5>Título 5</h5>
    <h6>Título 6</h6>
  </body>
</html>
```

Código HTML 2.9: Exemplo de cabeçalhos

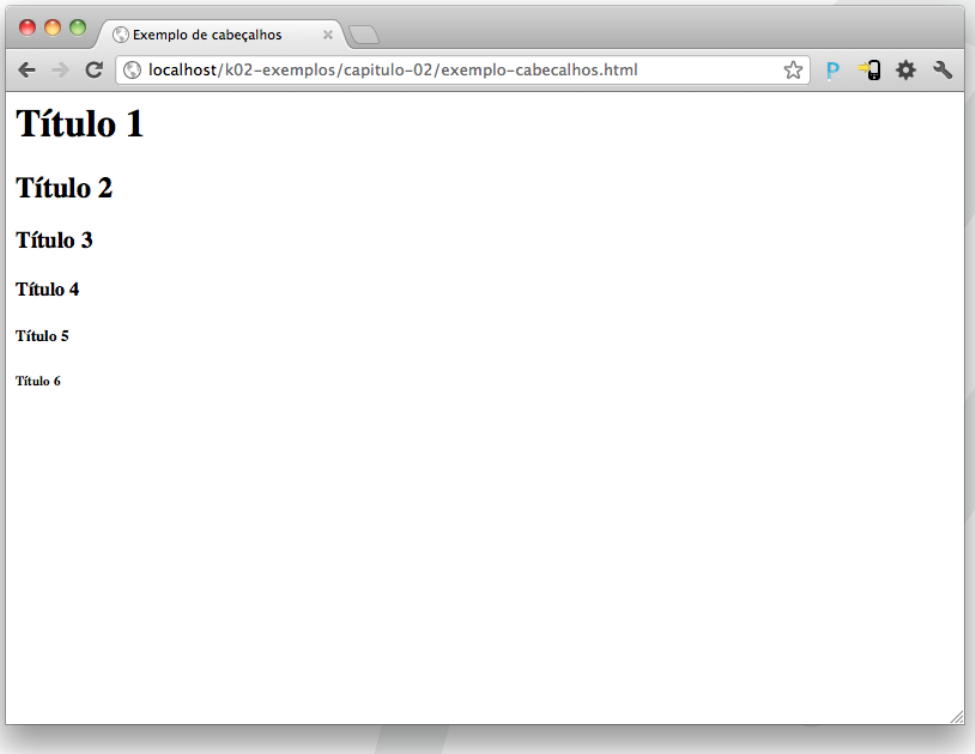

Figura 2.3: Exemplo de cabeçalhos

Perceba que cada nível possui um tamanho diferente de fonte. Esse tamanho é determinado pelo navegador e pode ser alterado através de regras CSS que veremos posteriormente.

Devemos utilizar os cabeçalhos com cautela, pois eles são utilizados como parâmetros de ranqueamento da página por diversos buscadores como Google, Yahoo e Bing, por exemplo. O uso correto das tags de cabeçalho faz parte das técnicas de SEO (Search Engine Optimization) que, como o próprio nome já indica, são técnicas que ajudam a melhorar o ranqueamento de páginas dentro dos buscadores.

De acordo com as técnicas de SEO devemos tomar os seguintes cuidados ao utilizarmos as tags de cabeçalhos:

- Utilizar apenas uma tag <h1> por página.
- Utilizar no máximo duas tags <h2> por página.

Exercícios de Fixação

4 Crie um novo documento HTML chamado **cabecalhos-1.html** com o conteúdo abaixo na pasta *html*. Em seguida, abra esse arquivo em um navegador.

```
1 <html>
```

```
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>K02 - Desenvolvimento Web com HTML, CSS e Javascript</title>
  </head>
  <body>
    <h1>K02 - Desenvolvimento Web com HTML, CSS e Javascript</h1>

    <p>Atualmente, praticamente todos os sistemas corporativos possuem
    interfaces web. Para quem deseja atuar no mercado de desenvolvimento
    de software, é obrigatório o conhecimento das linguagens: HTML, CSS
    e JavaScript.</p>

    <p>Essas linguagens são utilizadas para o desenvolvimento da camada de
    apresentação das aplicações web.</p>

    <h2>Pré-requisitos</h2>

    <p>Familiaridade com algum sistema operacional (Windows/Linux/Mac OS X)</p>
    <p>Lógica de programação</p>

    <h2>Agenda</h2>

    <h3>Aos domingos</h3>

    <p>xx/xx/xxxx das 08:00 às 14:00</p>
    <p>xx/xx/xxxx das 14:00 às 20:00</p>

    <h3>Aos sábados</h3>

    <p>xx/xx/xxxx das 08:00 às 14:00</p>
    <p>xx/xx/xxxx das 14:00 às 20:00</p>
  </body>
</html>
```

Código HTML 2.10: cabecalhos-1.html

5 Imagine que você possua um site de culinária no qual você disponibiliza algumas receitas. Crie uma página com uma receita fictícia utilizando cabeçalhos para organizar conteúdo. Utilize quantos níveis de título achar necessário.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Como preparar um delicioso macarrão instantâneo em 6 min.</title>
  </head>
  <body>
    <h1>Como preparar um delicioso macarrão instantâneo em 6 min.</h1>

    <p>Com esta receita você se tornará um profissional na arte de
    preparar um macarrão instantâneo.</p>

    <h2>Ingredientes</h2>

    <p>Macarrão instantâneo de sua marca favorita</p>
    <p>600ml de água</p>

    <h2>Modo de preparo</h2>

    <h3>No microondas</h3>

    <p>Insira o macarrão instantâneo em um recipiente com 600ml de água e
    programe o microondas por 6 minutos. Aperto o botão iniciar ou
    equivalente.</p>

    <h4>Ponto importante</h4>

```

```
27      <p>Utilize um recipiente que permita o macarrão ficar totalmente submerso
28      na água.</p>
29      <p>Quando ouvir o bip não saia correndo. O microondas não irá explodir,
30      pois o bip significa que o macarrão está pronto.</p>
31
32      <h3>No fogão</h3>
33
34      <p>Ferva a água em uma panela.</p>
35      <p>Insira o macarrão e cozinhe-o por 3 minutos</p>
36
37      <h4>Ponto importante</h4>
38
39      <p>Utilize uma panela que permita o macarrão ficar totalmente submerso
40      na água.</p>
41      <p>Não se distraia com a televisão ou qualquer outra coisa. Você poderá
42      queimar a sua refeição</p>
43    </body>
44  </html>
```

Código HTML 2.11: receita.html

Exercícios Complementares

3 Utilizando as tags de cabeçalho, crie uma página HTML que exiba hierarquicamente o nome e uma curiosidade de alguns continentes, países, estados (províncias) e cidades.

4 Utilizando as tags de cabeçalho, crie uma página de um produto e suas especificações, observações e ou comentários. Utilize quantos níveis de título achar necessário.

Links

Normalmente, um site é formado por um conjunto de páginas que estão interligadas de alguma forma. Para permitir que um usuário navegue de uma página para outra, devemos utilizar os links. Um link pode fazer a ligação de uma página para outra do mesmo site (link interno) ou para uma página de outro site (link externo).

Para criarmos um link, devemos utilizar a tag a. Porém, essa tag sem atributos não criará nenhum link interno ou externo. Para que um link seja criado, devemos, no mínimo, utilizar o atributo `href` com o caminho relativo ou absoluto de uma outra página. Veja o exemplo:

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exemplo de uso da tag a</title>
5     </head>
6     <body>
7       <p><a href="pagina2.html">Exemplo de link relativo</a></p>
8       <p><a href="outros/pagina3.html">Outro exemplo de link relativo</a></p>
9       <p><a href="http://www.k19.com.br">Exemplo de link absoluto</a></p>
10    </body>
11  </html>
```

Código HTML 2.14: Exemplo de uso da tag a

Além do atributo `href`, podemos utilizar o atributo `target` para informar onde o documento deve ser aberto. Os possíveis valores para o atributo `target` são:

- _blank - em uma nova janela ou aba
- _self - na mesma janela ou frame do documento que contém o link
- _parent - em um frame que seja o "pai" do frame no qual o link se encontra
- _top - na mesma janela do documento que contém o link

Ao testar os valores acima, logo percebemos que `_self` e `_top` possuem o mesmo comportamento se a página que contém o link não estiver em um frame. Caso o link esteja em um frame e com o valor `_top` no atributo `target`, o link será aberto na janela na qual o frame se encontra. Se o valor for `_self`, o link será aberto no próprio frame.

Dentro de uma única página podemos ter diversos frames e, às vezes, queremos que um link de um determinado frame seja aberto em outro. Para realizarmos tal tarefa devemos inserir como o valor do atributo `target` o nome do frame no qual o link será aberto.

O comportamento padrão de um link é abrir o documento na mesma página ou frame caso o atributo `target` não seja utilizado.

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exemplo de uso da tag a com o atributo target</title>
5    </head>
6    <body>
7      <p><a href="pagina1.html" target="_blank">Abre em outra janela/aba</a></p>
8      <p><a href="pagina2.html" target="_self">Abre na mesma janela</a></p>
9      <p><a href="pagina3.html">Abre na mesma janela</a></p>
10   </body>
11 </html>
```

Código HTML 2.15: Exemplo de uso da tag `a` *com o atributo* `target`

Importante

Ao longo da evolução do HTML, a tag `frame` foi caindo em desuso até que no HTML5 foi totalmente retirada da especificação. Contudo a grande maioria dos navegadores ainda interpretam a tag corretamente mesmo dentro de um documento HTML5. Porém, devemos nos lembrar que ainda estamos num momento de transição para o HTML5. Logo, para evitar problemas no futuro, evite o uso da tag `frame` ao máximo.

Exercícios de Fixação

6 Crie dois documentos HTML em arquivos chamados **pagina1.html** e **pagina2.html** dentro da pasta **html** e em seu corpo crie quatro links: um que aponte para uma página externa e outros três que apontem para uma página interna de maneiras diferentes. Lembre-se de criar também a página para a qual o seu link interno irá apontar.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag a com o atributo target</title>
  </head>
  <body>
    <p><a href="http://www.k19.com.br" target="_blank">Link externo</a></p>
    <p><a href="pagina2.html" target="_self">Link interno</a></p>
    <p><a href="pagina2.html" target="_top">Link interno</a></p>
    <p><a href="pagina2.html">Link interno</a></p>
  </body>
</html>
```

Código HTML 2.16: pagina1.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag a com o atributo target</title>
  </head>
  <body>
    <h1>Página 2</h1>
  </body>
</html>
```

Código HTML 2.17: pagina2.html

Exercícios Complementares

5 Crie vários links em uma página e para cada link escolha um `target` diferente. Crie também quantas páginas de destino forem necessárias (caso seja necessário).

6 Pesquise na internet como criar um `iframe` dentro de um documento HTML. Em seguida, crie uma página com alguns links e um `iframe`. Crie também alguns links na página contida no `iframe`. Para cada link em ambas as páginas, utilize valores diferentes para o atributo `target` e observe os resultados.

Âncoras

Além de criar links para outras páginas, podemos criar links para uma determinada seção dentro da própria página na qual o link se encontra ou dentro de outra página. Esse recurso chama-se **ancoragem**, pois as seções para as quais queremos criar um link devem possuir uma **âncora**.

Para criarmos uma âncora devemos utilizar novamente a tag `a`, porém sem o atributo `href`. Dessa vez utilizaremos o atributo `name` para identificar a seção através de um nome.

O link também muda levemente, pois agora ao invés de passar somente o nome do arquivo da página como valor do atributo `href` devemos passar o nome da seção prefixada com o caractere #.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
4       <title>Exemplo de uso da tag a como âncora</title>
5     </head>
6     <body>
7       <p><a href="#mais_info">Veja mais informações</a></p>
8       <p><a href="pagina2.html#outra_ancora">Âncora em outra página</a></p>
9       ...
10      ...
11      ...
12
13      <a name="mais_info">Mais informações</a>
14
15      <p>
16      ...
17      ...
18      ...
19      </p>
20    </body>
21  </html>
```

Código HTML 2.22: Exemplo de uso da tag a *como âncora*

Lembre-se

Até a versão 4 do HTML e no XHTML, a especificação dizia para utilizarmos o atributo `name` da tag `a` para criarmos as âncoras. Porém, no HTML5, a recomendação do W3C é que o atributo `id` seja utilizado para tal propósito. Desenvolvedores mais preocupados em estar sempre atualizados podem ficar tranquilos e utilizar somente o atributo `id` ao invés do `name`, pois os navegadores mais modernos conseguem interpretar o uso de ambos os atributos em qualquer versão do HTML.

Exercícios de Fixação

7 Crie um documento HTML em um arquivo chamado **ancora-pagina1.html** na pasta **html** que contenha um link que aponta para uma âncora dentro da própria página. Dica: insira um conteúdo suficientemente grande para que a barra de rolagem vertical do navegador apareça e coloque a âncora no final da página.

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício sobre âncoras</title>
5     </head>
6     <body>
7       <p><a href="#sobre">Sobre o texto dessa página</a></p>
8
9       <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
10      massa, sodales sit amet eleifend a, elementum eu nibh. Donec egestas dolor
11      quis turpis dictum tincidunt. Donec blandit tempus velit, sit amet
12      adipiscing velit consequat placerat. Curabitur id mauris.</p>
13      <p> ... </p>
14      <p> ... </p>
15      <p> ... </p>
16
17      <p>At nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra.
18      Duis vel tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
19      accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>
20
```

```
21      <a name="sobre">Sobre o texto dessa página</a>
22      <p>O texto dessa página foi gerado através do site:
23      <a href="http://www.lipsum.com/">http://www.lipsum.com/</a></p>
24    </body>
25  </html>
```

Código HTML 2.23: ancora-pagina1.html

8 Crie um novo arquivo chamado **ancora-pagina2.html** na pasta **html** com um âncora chamada **outra_ancora**. Dica: insira um conteúdo suficientemente grande para que a barra de rolagem vertical do navegador apareça e coloque a âncora no final da página.

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício sobre âncoras</title>
5     </head>
6     <body>
7       <h1>Ancora página 2</h1>
8
9       <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
10      massa, sodales sit amet eleifend a, elementum eu nibh. Donec egestas dolor
11      quis turpis dictum tincidunt. Donec blandit tempus velit, sit amet
12      adipiscing velit consequat placerat. Curabitur id mauris </p>
13      <p> ... </p>
14      <p> ... </p>
15      <p> ... </p>
16
17      <p> at nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra.
18      Duis vel tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
19      accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>
20
21      <a name="outra_ancora">Mais uma âncora</a>
22
23      <p>Se você chegou aqui deu tudo certo! :)</p>
24    </body>
25  </html>
```

Código HTML 2.24: ancora-pagina2.html

9 Crie um novo link no arquivo **ancora-pagina1.html** que aponte para âncora **outra_ancora** do arquivo **ancora-pagina2.html**.

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício sobre âncoras</title>
5     </head>
6     <body>
7       <p><a href="#sobre">Sobre o texto dessa página</a></p>
8       <p><a href="ancora-pagina2.html#outra_ancora">Âncora em outra página</a></p>
9
10      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
11      massa, sodales sit amet eleifend a, elementum eu nibh. Donec egestas dolor
12      quis turpis dictum tincidunt. Donec blandit tempus velit, sit amet
13      adipiscing velit consequat placerat. Curabitur id mauris.</p>
14      <p> ... </p>
15      <p> ... </p>
16      <p> ... </p>
17
18      <p>At nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra.
19      Duis vel tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
20      accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>
```

```
21
22      <a name="sobre">Sobre o texto dessa página</a>
23      <p>O texto dessa página foi gerado através do site:
24      <a href="http://www.lipsum.com/">http://www.lipsum.com/</a></p>
25    </body>
26  </html>
```

Código HTML 2.25: ancora-pagina1.html

Exercícios Complementares

7 Crie dois documentos HTML. No primeiro crie um link que aponte para o site da K19 e também coloque um texto qualquer. Em qualquer ponto deste texto, crie uma âncora. No segundo documento crie um link que aponte para a âncora criada no primeiro documento, coloque também um link qualquer ou um texto.

8 Dentro de um documento HTML crie três links. Dois devem apontar para âncoras de páginas externas e o último link deve aparecer no final da página e apontar para uma âncora no topo da própria página.

Imagens

Certamente, os sites na internet seriam muito entediantes se não fosse possível adicionar imagens ao conteúdo das páginas. Como não queremos que as nossas páginas fiquem muito monótonas, neste capítulo, utilizaremos a tag `img` para melhorar um pouco a aparência das páginas com algumas imagens.

A tag `img` possui o atributo `src` que utilizaremos para informar qual imagem queremos carregar dentro de um documento HTML. O valor do atributo pode ser o caminho absoluto ou relativo de uma imagem.

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exemplo de uso da tag img</title>
5     </head>
6     <body>
7       <h1>K19 Treinamentos</h1>
8       <img src="http://www.k19.com.br/css/img/main-header-logo.png" />
9
10      <h2>Cursos</h2>
11      <p>
12        <img src="http://www.k19.com.br/css/img/k01-logo-large.png" />
13        K01 - Lógica de Programação
14      </p>
15      <p>
16        <img src="http://www.k19.com.br/css/img/k02-logo-large.png" />
17        K02 - Desenvolvimento Web com HTML, CSS e JavaScript
18      </p>
19      <p>
20        <img src="http://www.k19.com.br/css/img/k03-logo-large.png" />
21        K03 - SQL e Modelo Relacional
22      </p>
```

```
23      <p>
24        <img src="http://www.k19.com.br/css/img/k11-logo-large.png" />
25        K11 - Orientação a Objetos em Java
26      </p>
27      <p>
28        <img src="http://www.k19.com.br/css/img/k12-logo-large.png" />
29        K12 - Desenvolvimento Web com JSF2 e JPA2
30      </p>
31      ...
32      ...
33      ...
34    </body>
35  </html>
```

Código HTML 2.29: Exemplo de uso da tag `img`

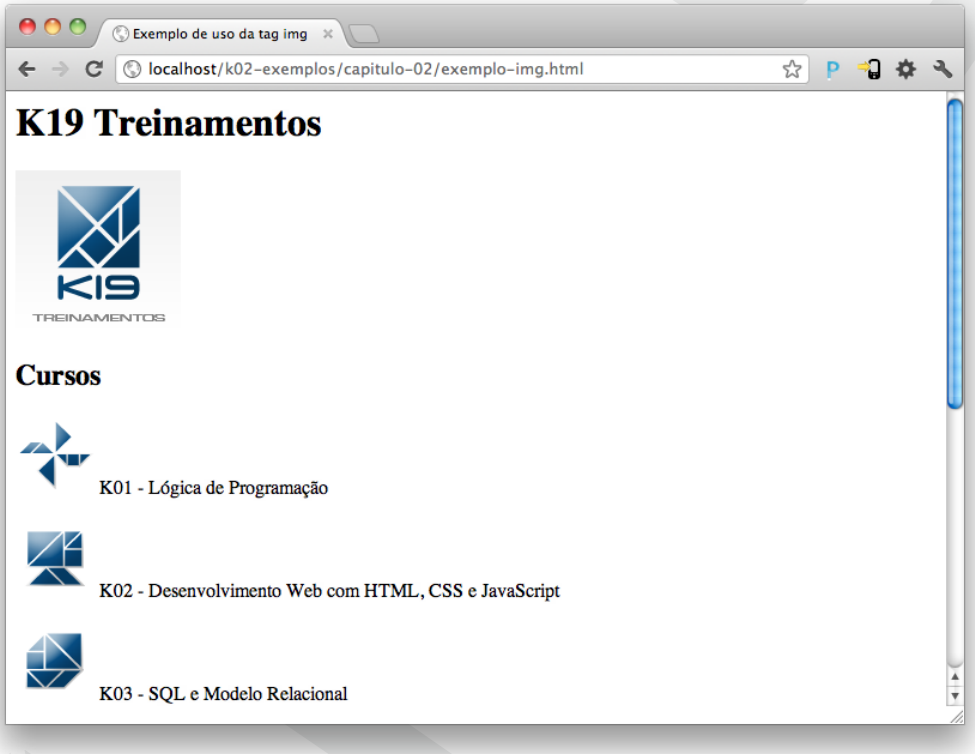

Figura 2.4: Exemplo de uso da tag `img`

Exercícios de Fixação

10 Crie um documento HTML em um arquivo chamado **imagem.html** na pasta **html** que contenha alguns elementos `IMG`.

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exemplo de uso da tag img</title>
5     </head>
```

```
6     <body>
7       <h1>K19 Treinamentos</h1>
8       <img src="http://www.k19.com.br/css/img/main-header-logo.png" />
9
10      <h2>Cursos</h2>
11      <p>
12        <img src="http://www.k19.com.br/css/img/k01-logo-large.png" />
13        K01 - Lógica de Programação
14      </p>
15      <p>
16        <img src="http://www.k19.com.br/css/img/k02-logo-large.png" />
17        K02 - Desenvolvimento Web com HTML, CSS e JavaScript
18      </p>
19      <p>
20        <img src="http://www.k19.com.br/css/img/k03-logo-large.png" />
21        K03 - SQL e Modelo Relacional
22      </p>
23    </body>
24  </html>
```

Código HTML 2.30: imagem.html

Exercícios Complementares

9 Em um documento HTML insira no mínimo duas imagens utilizando o elemento `IMG`.

Tabelas

Suponha que você esteja desenvolvendo o site de uma empresa e precisa apresentar alguns relatórios em páginas HTML. Considere que os dados desses relatórios são obtidos de planilhas eletrônicas. Daí surge a necessidade de definir dados de forma tabular em HTML.

Para resolver esse problema, podemos utilizar o elemento `table` do HTML. Esse elemento permite apresentar um conjunto de dados na forma de tabelas. Veja o exemplo:

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exemplo de uso da tag table</title>
5     </head>
6     <body>
7       <h1>Carros</h1>
8
9       <table>
10        <tr>
11          <th>Marca</th>
12          <th>Modelo</th>
13          <th>Ano</th>
14        </tr>
15        <tr>
16          <td>Toyota</td>
17          <td>Corolla</td>
18          <td>2010</td>
19          </tr>
20          <tr>
21          <td>Honda</td>
22          <td>Civic</td>
23          <td>2011</td>
```

```
24        </tr>
25        <tr>
26          <td>Mitsubishi</td>
27          <td>Lancer</td>
28          <td>2012</td>
29        </tr>
30        <tr>
31          <td colspan="3">Última atualização: 06/2012</td>
32        </tr>
33      </table>
34    </body>
35  </html>
```

Código HTML 2.32: Exemplo de uso da tag `table`

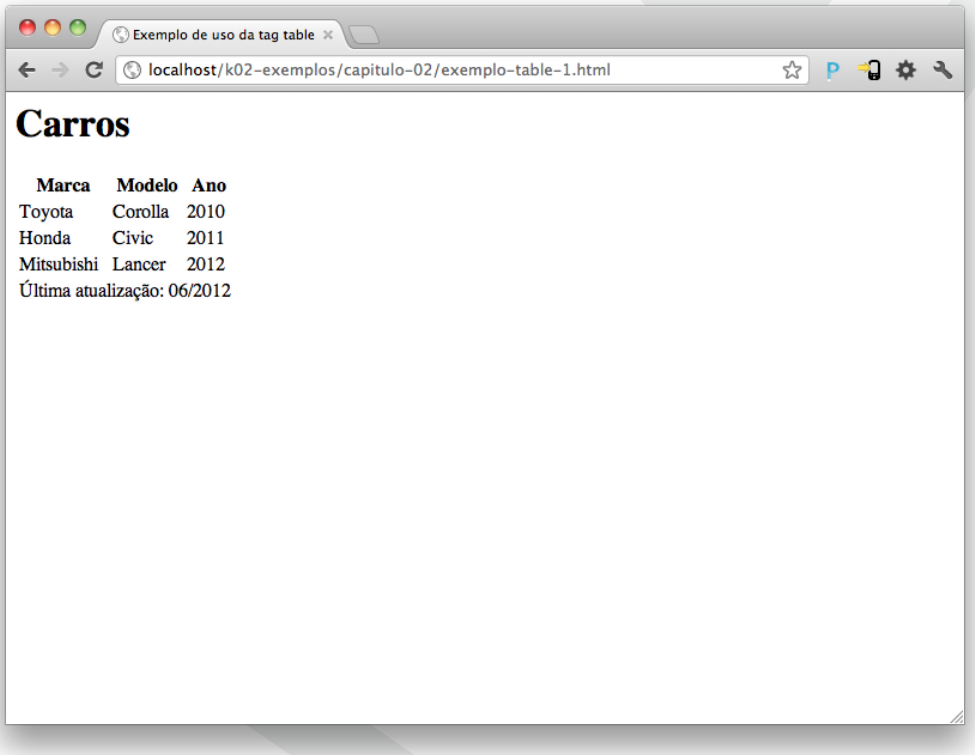

Figura 2.5: Exemplo de uso da tag `table`

Perceba que a tag `table` não é utilizada sozinha. Ela necessita de pelo menos um ou mais elementos `tr` que, por sua vez, necessitam de pelo menos um ou mais elementos `th` ou `td`.

O que significam essas tags?

- tr - define uma linha da tabela
- th - define uma célula de cabeçalho
- td - define uma célula

Existe uma outra forma de criar a mesma tabela:

```
1  <html>
```

```
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Exemplo de uso da tag table</title>
5   </head>
6   <body>
7     <h1>Carros</h1>
8
9     <table>
10      <thead>
11        <tr>
12          <th>Marca</th>
13          <th>Modelo</th>
14          <th>Ano</th>
15        </tr>
16      </thead>
17      <tfoot>
18        <tr>
19          <td colspan="3">Última atualização: 06/2012</td>
20        </tr>
21      </tfoot>
22      <tbody>
23        <tr>
24          <td>Toyota</td>
25          <td>Corolla</td>
26          <td>2010</td>
27        </tr>
28        <tr>
29          <td>Honda</td>
30          <td>Civic</td>
31          <td>2011</td>
32        </tr>
33        <tr>
34          <td>Mitsubishi</td>
35          <td>Lancer</td>
36          <td>2012</td>
37        </tr>
38      </tbody>
39    </table>
40  </body>
41 </html>
```

Código HTML 2.33: Exemplo de uso da tag `table`

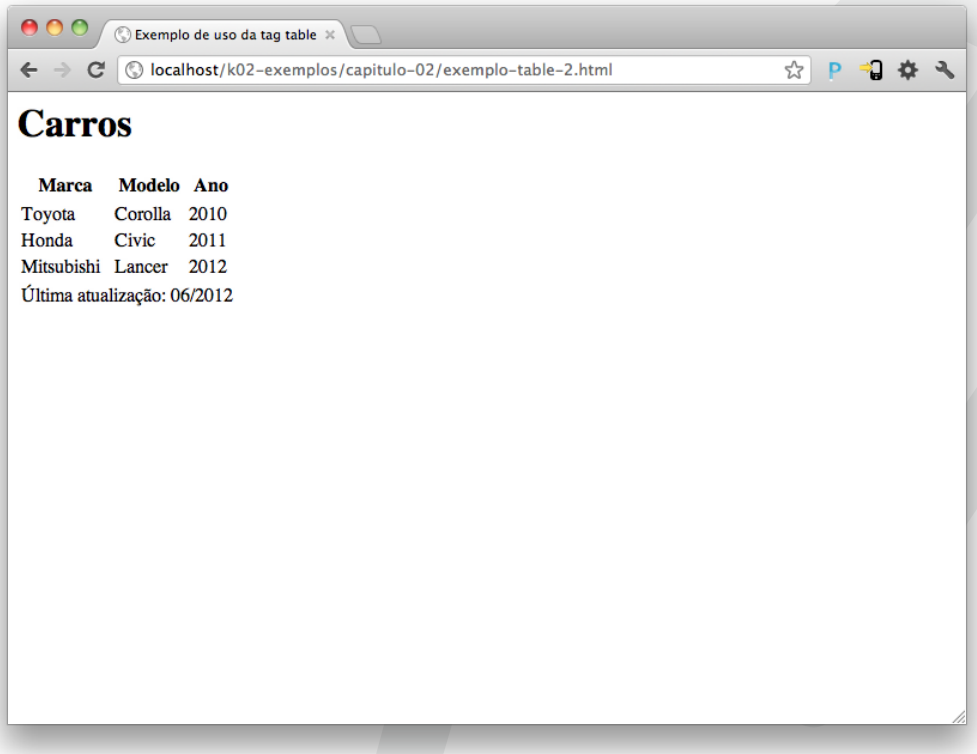

Figura 2.6: Exemplo de uso da tag `table`

Repare que visualmente não mudou absolutamente nada. Além disso, apareceram mais algumas tags: `thead`, `tfoot` e `tbody`.

O que significam essas tags?

- thead - define o cabeçalho da tabela
- tfoot - define o rodapé da tabela
- tbody - define o corpo da tabela

Por que complicar? Qual o motivo da existência dessas tags?

- A tag `thead`, assim como a `tfoot`, servem para agrupar as linhas de cabeçalho e de rodapé, respectivamente.
- O código fica mais claro.
- Facilita a aplicação de estilos CSS (através do seletor de elemento)
- Pode permitir que o conteúdo do corpo da tabela possua rolagem*.
- Ao imprimir a página com uma tabela muito extensa, pode permitir que o cabeçalho e rodapé sejam replicados em todas as páginas*.

* Esses recursos podem existir ou não, pois os desenvolvedores de navegadores podem decidir não implementá-los. Esses recursos são sugestões da especificação.

Outros dois atributos importantes para a construção de tabelas são `colspan` e `rowspan` que podem ser aplicados nos elementos `td` e `th`.

Como podemos observar nos exemplos dados, o atributo `colspan` faz com que a célula ignore o número de colunas definidas em seu valor. Analogamente, o atributo `rowspan` faz o mesmo, porém com linhas.

Exercícios de Fixação

11 Crie uma página HTML em um arquivo chamado **tabela.html** na pasta **html** que contenha uma tabela de acordo com a imagem abaixo:

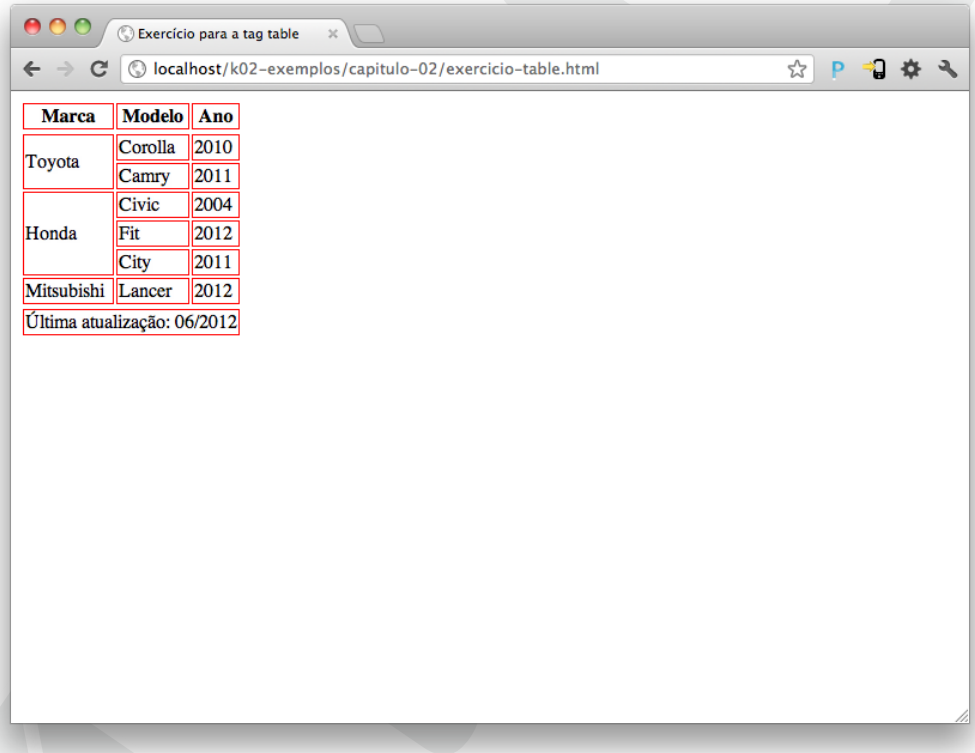

Figura 2.7: Exercício para a tag `table`

As linhas vermelhas foram colocadas na imagem apenas para facilitar a visualização do problema.

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exercício para a tag table</title>
5    </head>
6    <body>
7      <table>
8        <thead>
9          <tr>
```

```
10            <th>Marca</th>
11            <th>Modelo</th>
12            <th>Ano</th>
13          </tr>
14        </thead>
15        <tfoot>
16          <tr>
17            <td colspan="3">Última atualização: 06/2012</td>
18          </tr>
19        </tfoot>
20        <tbody>
21          <tr>
22            <td rowspan="2">Toyota</td>
23            <td>Corolla</td>
24            <td>2010</td>
25          </tr>
26          <tr>
27            <td>Camry</td>
28            <td>2011</td>
29          </tr>
30
31          <tr>
32            <td rowspan="3">Honda</td>
33            <td>Civic</td>
34            <td>2004</td>
35          </tr>
36          <tr>
37            <td>Fit</td>
38            <td>2012</td>
39          </tr>
40          <tr>
41            <td>City</td>
42            <td>2011</td>
43          </tr>
44
45          <tr>
46            <td>Mitsubishi</td>
47            <td>Lancer</td>
48            <td>2012</td>
49          </tr>
50        </tbody>
51      </table>
52    </body>
53  </html>
```

Código HTML 2.34: tabela.html

Exercícios Complementares

10 Crie em um documento HTML uma tabela que contenha o continente/subcontinente, o nome e o idioma de algumas cidades do mundo.

Listas

Em um documento HTML podemos ter três tipos de listas e cada uma delas deve ser utilizada de acordo com a sua semântica, ou seja, você deve escolher um tipo de lista para cada situação.

Os três tipos possíveis de listas são:

- Lista de definição - utilizada para exibir definições de termos. Funciona como nos dicionários, no qual temos uma palavra e em seguida o seu significado.

- Lista ordenada - utilizada para exibir qualquer conteúdo de forma ordenada.

- Lista sem ordem - utilizada para exibir qualquer conteúdo sem ordenação.

Lista de definição

Para criarmos uma lista de definição devemos utilizar a tag `dl`. O elemento `dl` deve possuir pelo menos um elemento filho `dt` seguido de um elemento `dd`. Um item em uma lista de definição é composto por um par de elementos `dt` e `dd`.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag dl</title>
  </head>
  <body>
    <h1>Cursos - K19 Treinamentos</h1>

    <dl>
      <dt>K01 - Lógica de Programação</dt>
      <dd>
        Conhecimentos em Lógica de Programação é o pré-requisito fundamental
        para que uma pessoa consiga aprender qualquer Linguagem de Programação...
      </dd>
      <dt>K02 - Desenvolvimento Web com HTML, CSS e JavaScript</dt>
      <dd>
        Atualmente, praticamente todos os sistemas corporativos possuem
        interfaces web. Para quem deseja atuar no mercado de desenvolvimento...
      </dd>
      <dt>K03 - SQL e Modelo Relacional</dt>
      <dd>
        Como as aplicações corporativas necessitam armazenar dados é fundamental
        que os desenvolvedores possuam conhecimentos sobre persistência de dados.
      </dd>
    </dl>
  </body>
</html>
```

Código HTML 2.36: Exemplo de uso da tag `dl`

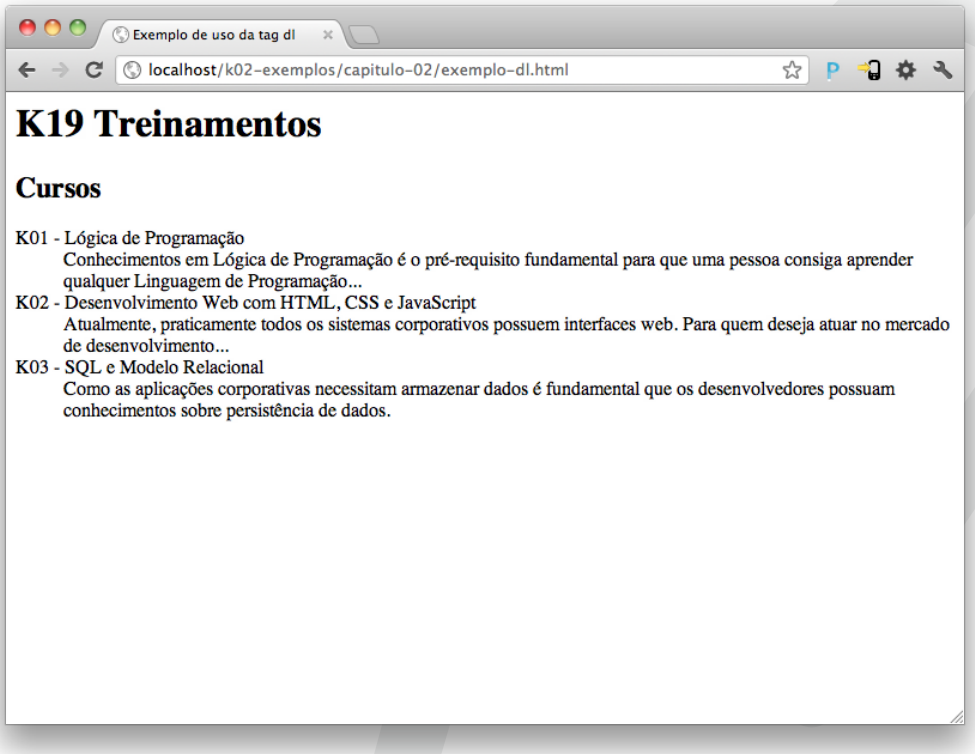

Figura 2.8: Exemplo de uso da tag `dl`

Exercícios de Fixação

12 Crie um documento HTML em um arquivo chamado **restaurante.html** na pasta **html** que contenha o cardápio de um restaurante com os nomes dos seus pratos e uma breve descrição sobre os mesmos.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Menu - K19 Pizzaria</title>
  </head>
  <body>
    <h1>Menu - K19 Pizzaria</h1>

    <dl>
      <dt>À moda da casa</dt>
      <dd>
        Presunto coberto com mussarela, ovos e palmito.
      </dd>
      <dt>À moda do pizzaiolo</dt>
      <dd>
        Mussarela, presunto, ovos e bacon.
      </dd>
      <dt>Aliche</dt>
      <dd>
        Aliche, parmesão e rodelas de tomate.
      </dd>
```

```
22      </dl>
23    </body>
24  </html>
```

Código HTML 2.37: restaurante.html

Exercícios Complementares

11 Crie um documento HTML que contenha uma lista de alguns pontos turísticos do Brasil de que você tenha conhecimento e cite algumas atrações do mesmo.

Lista ordenada

Para criarmos uma lista ordenada, devemos utilizar a tag `ol`. O elemento `ol` deve possuir pelo menos um elemento filho `li`.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag ol</title>
  </head>
  <body>
    <h1>Macarrão instantâneo - K19 Receitas</h1>
    <h2>Modo de preparo</h2>

    <ol>
      <li>Ferver 600ml de água em uma panela.</li>
      <li>Retirar o macarrão do pacote.</li>
      <li>Colocar o macarrão na panela no fogo baixo.</li>
      <li>Cozinhar o macarrão por 3min.</li>
      <li>Temperar a gosto.</li>
    </ol>
  </body>
</html>
```

Código HTML 2.39: Exemplo de uso da tag `ol`

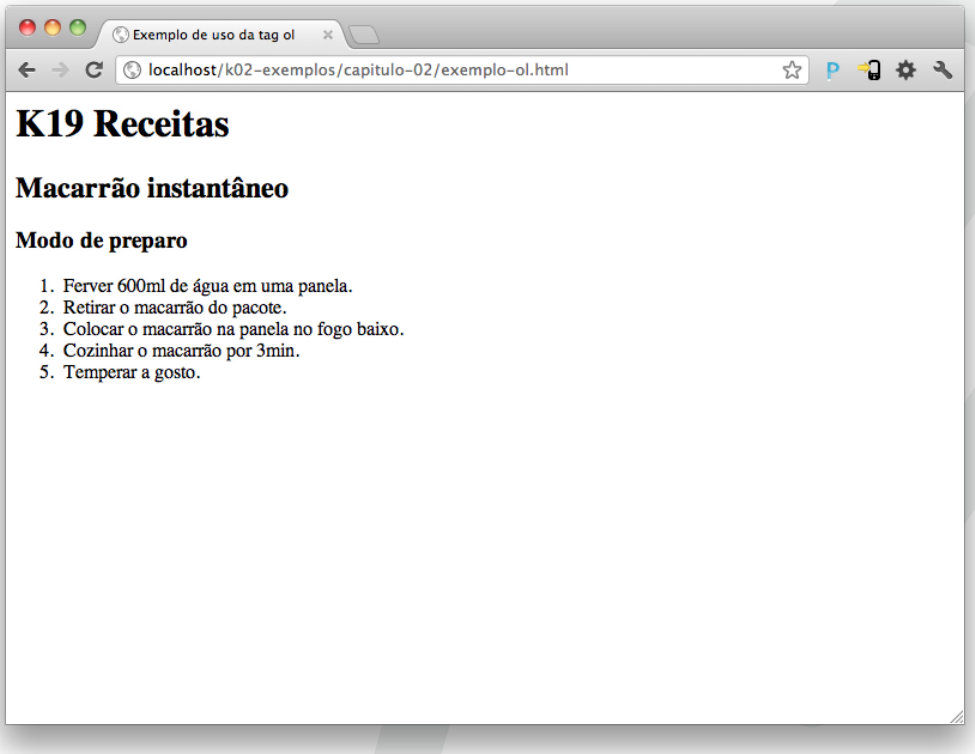

Figura 2.9: Exemplo de uso da tag `ol`

Exercícios de Fixação

13 Crie um documento HTML em um arquivo chamado **manual.html** na pasta **html** que contenha um manual que explica passo-a-passo o uso de um caixa eletrônico para a operação de saque.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Operação de saque - Manual do caixa eletrônico - K19 Bank</title>
  </head>
  <body>
    <h1>Operação de saque - Manual do caixa eletrônico - K19 Bank</h1>

    <ol>
      <li>Insira o cartão</li>
      <li>Digite a senha</li>
      <li>Escolha a opção de saque</li>
      <li>Informe o valor que deseja sacar</li>
      <li>Insira o cartão novamente</li>
      <li>Aguarde até a liberação do dinheiro</li>
    </ol>
  </body>
</html>
```

Código HTML 2.40: manual.html

Exercícios Complementares

12 Crie um documento HTML que contenha um manual que explique passo-a-passo a instalação, manutenção ou manuseio de um aparelho eletrônico.

Lista sem ordem

Para criarmos uma lista sem ordem, devemos utilizar a tag `ul`. O elemento `ul` deve possuir pelo menos um elemento filho `li`.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag dl</title>
  </head>
  <body>
    <h1>K02 - Desenvolvimento Web com HTML, CSS e JavaScript</h1>
    <h2>Pré-requisitos</h2>

    <ul>
      <li>Conhecimento de algum sistema operacional (Windows/Linux/MacOS X)</li>
      <li>Lógica de programação</li>
    </ul>
  </body>
</html>
```

Código HTML 2.42: Exemplo de uso da tag `ul`

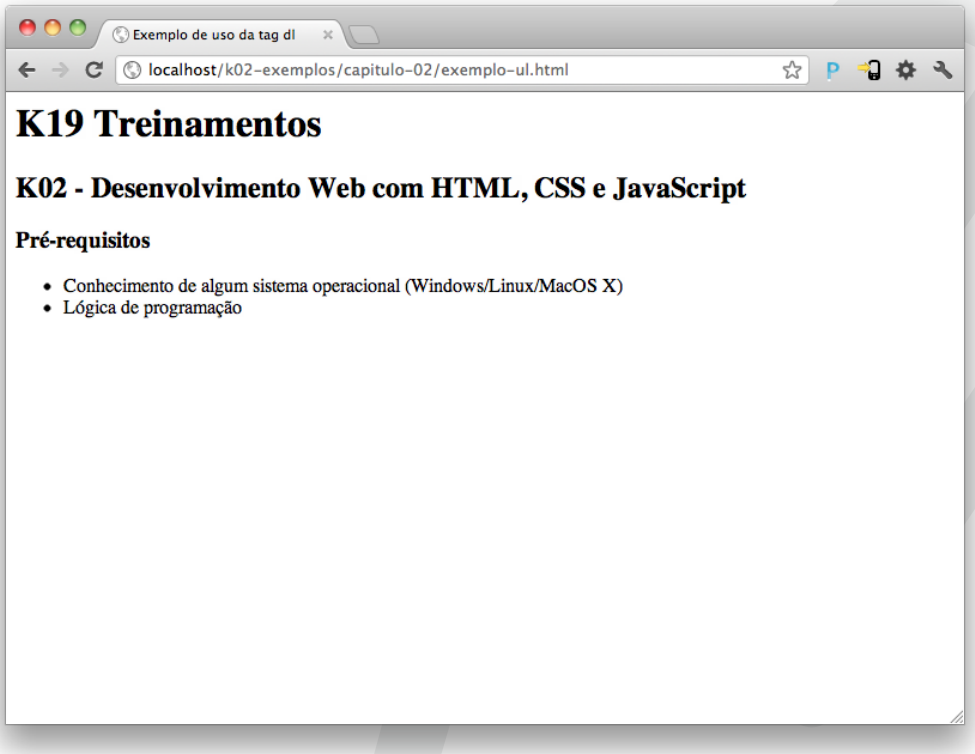

Figura 2.10: Exemplo de uso da tag `ul`

Exercícios de Fixação

14 Crie um documento HTML em um arquivo chamado **lista-curso.html** na pasta **html** que contenha a lista dos cursos da Formação Básica da K19.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>K00 - Formação Básica - K19 Treinamentos</title>
  </head>
  <body>
    <dl>
      <h1>K00 - Formação Básica</h1>

      <ul>
        <li>K01 - Lógica de Programação</li>
        <li>K02 - Desenvolvimento Web com HTML, CSS e JavaScript</li>
        <li>K03 - SQL e Modelo Relacional</li>
      </ul>
    </dl>
  </body>
</html>
```

Código HTML 2.43: lista-cursos.html

Exercícios Complementares

13 Crie um documento HTML que contenha a lista dos cursos da Formação Desenvolvedor Java da K19.

Formulário

Para trazermos um pouco mais de interatividade para as nossas páginas, podemos utilizar os elementos de formulário. Esses elementos recebem algum tipo de entrada do usuário, seja através de um clique ou digitando algum valor.

A tag `input`

A tag `input` permite que o elemento que a contenha assuma diversas formas dependendo do seu atributo `type`. O atributo `type` pode receber os seguintes valores:

- text - cria uma caixa de texto de uma linha.
- password - cria uma caixa de texto de uma linha escondendo os caracteres digitados.
- checkbox - cria uma caixa que assume dois estados: checado e "deschecado". Em conjunto com o atributo `name` é possível que se crie um grupo de *checkboxes* no qual um ou mais *checkboxes* seja "checado".
- radio - cria uma caixa que assume dois estados: checado e "deschecado". Em conjunto com o atributo `name` é possível que se crie um grupo de *radios* no qual apenas um *radio* seja "checado".
- button - cria um botão. Através do atributo `value` definimos o texto do botão.
- submit - cria um botão para o envio do formulário. Através do atributo `value` definimos o texto do botão.
- file - cria um botão que, ao ser clicado, abre uma caixa de diálogo para a escolha de um arquivo no computador do usuário.
- reset - cria um botão que descarta todas as alterações feitas dentro de um formulário. Através do atributo `value` definimos o texto do botão.
- image - cria um botão para o envio do formulário. Dese ser utilizado em conjunto com o atributo `src` para que uma imagem de fundo seja utilizada no botão.
- hidden - cria um elemento que não fica visível para o usuário, porém pode conter um valor que será enviado pelo formulário.

Existem outros valores possíveis para o atributo `type`, porém eles fazem parte da especificação do HTML5 e nem todos os navegadores suportam tais valores.

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exemplo de uso da tag input</title>
5    </head>
6    <body>
```

```
7     <form action="pagina.html" method="get">
8       <p>
9         text:
10        <input type="text" />
11      </p>
12      <p>
13        password:
14        <input type="password" />
15      </p>
16      <p>
17        checkboxes:
18        <input type="checkbox" name="checkgroup" />
19        <input type="checkbox" name="checkgroup" />
20        <input type="checkbox" name="checkgroup" />
21      </p>
22      <p>
23        radios:
24        <input type="radio" name="checkgroup" />
25        <input type="radio" name="checkgroup" />
26        <input type="radio" name="checkgroup" />
27      </p>
28      <p>
29        button:
30        <input type="button" value="Botão" />
31      </p>
32      <p>
33        submit:
34        <input type="submit" value="Enviar" />
35      </p>
36      <p>
37        file:
38        <input type="file" />
39      </p>
40      <p>
41        reset:
42        <input type="reset" value="Descartar alterações" />
43      </p>
44      <p>
45        image:
46        <input
47        type="image"src="http://www.k19.com.br/css/img/main-header-logo.png" />
48      </p>
49      <p>
50        hidden:
51        <input type="hidden" value="valor escondido" />
52      </p>
53    </form>
54  </body>
55 </html>
```

Código HTML 2.45: Exemplo de uso da tag `input`

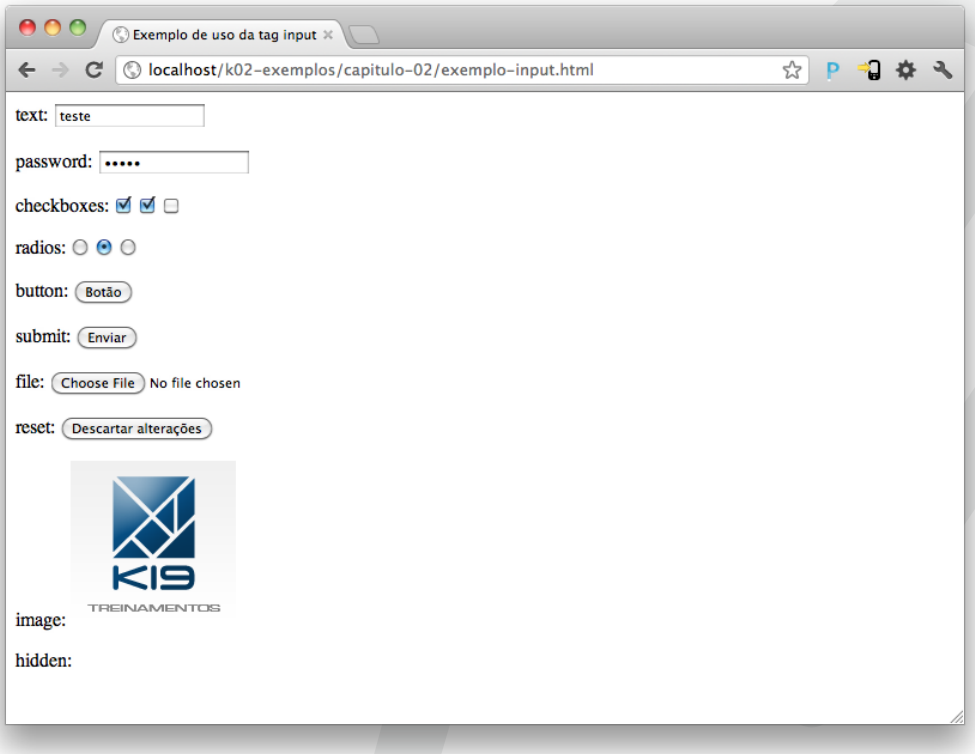

Figura 2.11: Exemplo de uso da tag `input`

A tag `select`

A tag `select` permite ao usuário escolher um ou mais itens de uma lista. O atributo `multiple`, quando presente, informa ao navegador que mais de um item pode ser selecionado.

A lista de itens deve ser informada através de elementos `option`. Tais elementos devem ser filhos diretos ou indiretos de elementos `select`. Além disso, cada item pode conter o atributo chamado `value` para informar o valor associado a uma determinada opção.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag select</title>
  </head>
  <body>
    <form action="pagina.html" method="get">
      <p>
        Selecione uma cidade:
        <select>
          <option value="sao-paulo">São Paulo</option>
          <option value="rio-de-janeiro">Rio de Janeiro</option>
          <option value="porto-alegre">Porto Alegre</option>
          <option value="curitiba">Curitiba</option>
        </select>
      </p>

      <p>
        Selecione uma ou mais categorias de produtos (mantenha a tecla
        "control" (ou "command" no Mac) pressionada para escolher mais de uma
        categoria):
        <select multiple="multiple">
```

```
23            <option value="desktops">Desktops</option>
24            <option value="notebooks">Notebooks</option>
25            <option value="tablets">Tablets</option>
26            <option value="celulares">Celulares</option>
27          </select>
28        </p>
29      </form>
30    </body>
31  </html>
```

Código HTML 2.46: Exemplo de uso da tag `select`

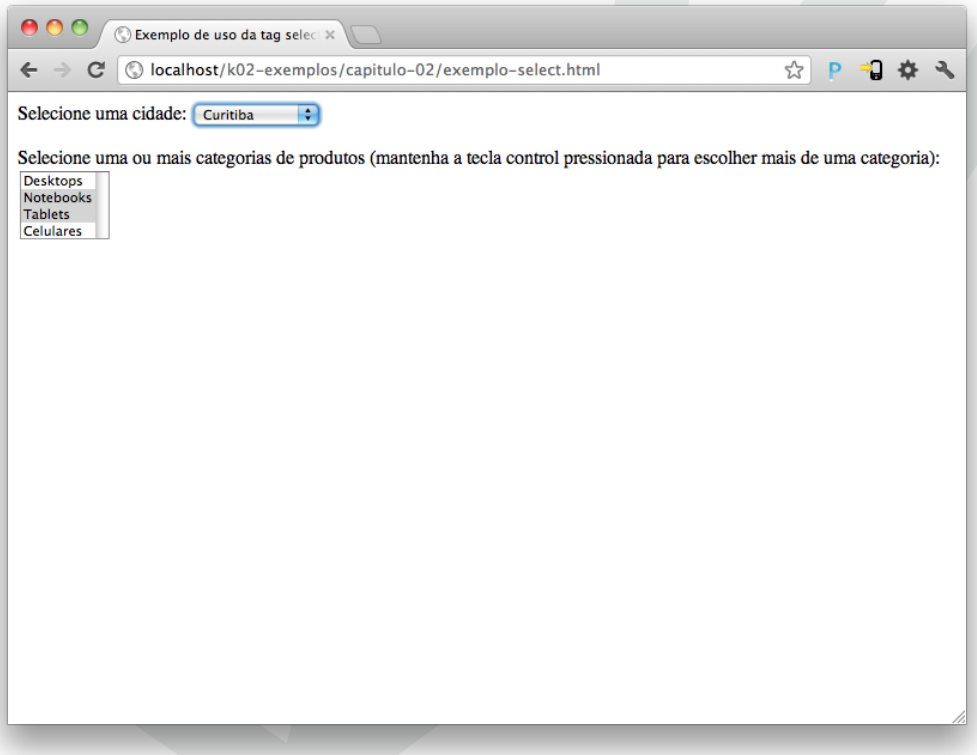

Figura 2.12: Exemplo de uso da tag `select`

Caso exista a necessidade de agrupar as opções de um elemento `select`, podemos utilizar o elemento `optgroup` em conjunto com o atributo `label`. Veja o exemplo:

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exemplo de uso da tag select</title>
5     </head>
6     <body>
7       <form action="pagina.html" method="get">
8         <p>
9           Selecione uma cidade:
10          <select>
11            <optgroup label="Região Sudeste">
12              <option value="sao-paulo">São Paulo</option>
13              <option value="rio-de-janeiro">Rio de Janeiro</option>
14            </optgroup>
15            <optgroup label="Região Sul">
16              <option value="porto-alegre">Porto Alegre</option>
```

```
17              <option value="curitiba">Curitiba</option>
18            </optgroup>
19          </select>
20        </p>
21
22        <p>
23          Selecione uma ou mais categorias de produtos (mantenha a tecla
24          "control" (ou "command" no Mac) pressionada para escolher mais de uma
25          categoria):
26          <select multiple="multiple">
27            <optgroup label="De mesa">
28              <option value="desktops">Desktops</option>
29            </optgroup>
30            <optgroup label="Portáteis">
31              <option value="notebooks">Notebooks</option>
32              <option value="tablets">Tablets</option>
33              <option value="celulares">Celulares</option>
34            </optgroup>
35          </select>
36        </p>
37      </form>
38    </body>
39  </html>
```

Código HTML 2.47: Exemplo de uso da tag `select`

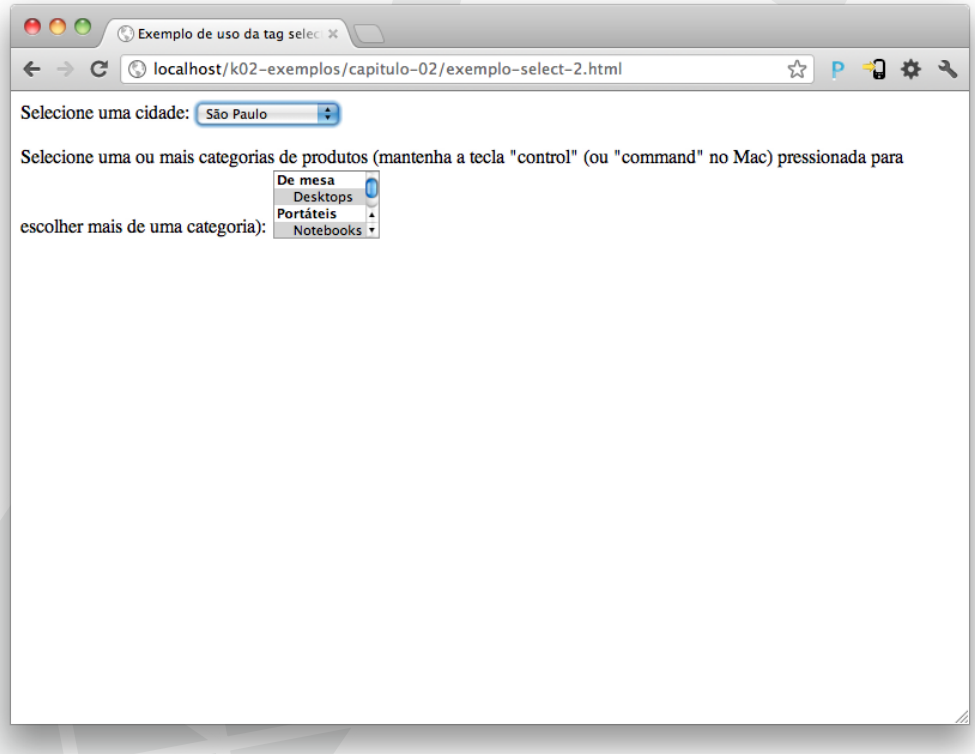

Figura 2.13: Exemplo de uso da tag `select`

A tag `textarea`

A tag `textarea` exibe uma caixa de texto na qual o usuário poderá inserir um texto qualquer. A diferença para a tag `input` com o atributo `type` com o valor `text` é que a tag `textarea` permite a inserção de textos mais longos e com quebras de linha.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de uso da tag textarea</title>
  </head>
  <body>
    <form action="pagina.html" method="get">
      <p>
        textarea:
        <textarea>
        </textarea>
      </p>
    </form>
  </body>
</html>
```

Código HTML 2.48: Exemplo de uso da tag `textarea`

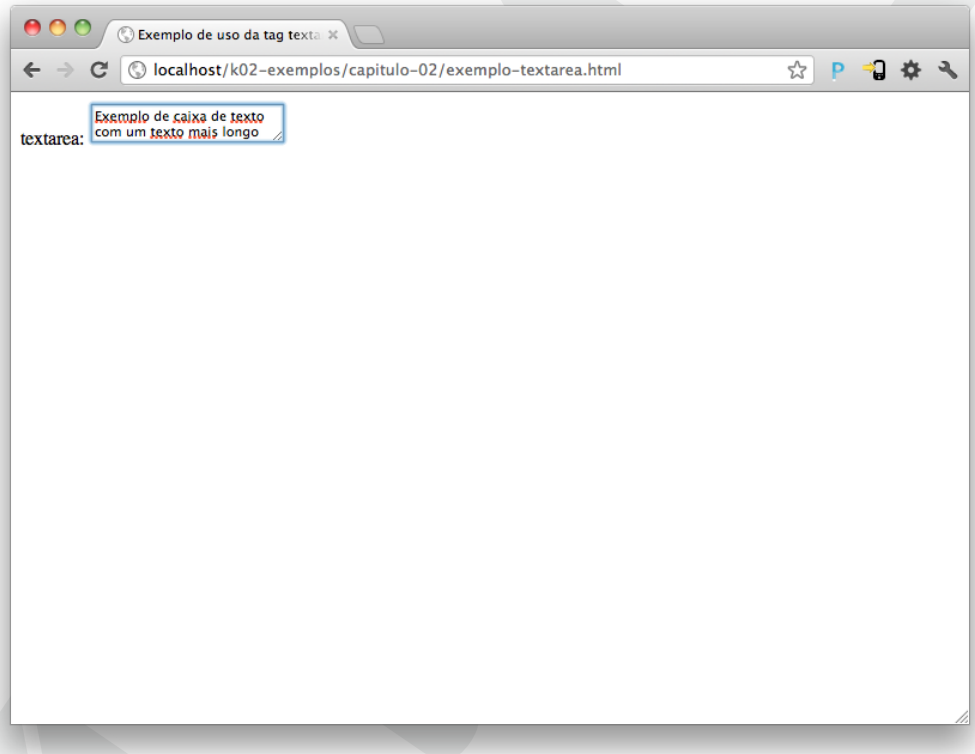

Figura 2.14: Exemplo de uso da tag `textarea`

A tag `label`

Em alguns dos exemplos anteriores foram inseridos textos ao lado dos elementos de formulário apresentados. Podemos pensar nesses textos como rótulos de cada elemento e é exatamente para esse fim que devemos utilizar a tag `label` do HTML.

Além de servir como rótulo, a tag `label` auxilia o usuário a interagir com os elementos do formulário. Utilizando o atributo `for` podemos fazer com que um elemento do formulário receba o foco. Veja o exemplo:

```
<html>
```

```
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Exemplo de uso da tag label</title>
5   </head>
6   <body>
7     <form action="pagina.html" method="get">
8       <p>
9         <label for="nome">Nome:</label>
10        <input type="text" id="nome" />
11      </p>
12    </form>
13  </body>
14 </html>
```

Código HTML 2.49: Exemplo de uso da tag `label`

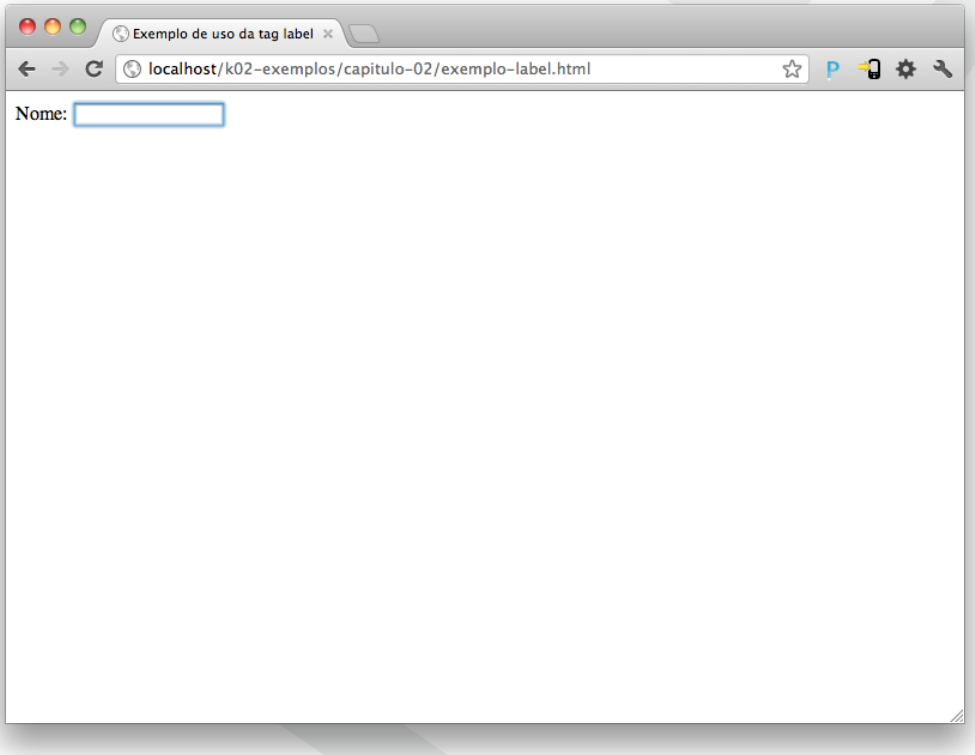

Figura 2.15: Exemplo de uso da tag `label`

Repare que o atributo `for` da tag `label` deve conter um valor igual ao do atributo `id` do elemento que desejamos focar.

A tag `form`

Desde o momento em que começamos a falar sobre os elementos de formulário utilizamos a tag `form`, porém não falamos nada sobre ela. A tag `form` irá definir, de fato, o nosso formulário. Todos os elementos de formulário que vimos anteriormente devem ser filhos diretos ou indiretos de um elemento com a tag `form` para que os dados vinculados a esses elementos sejam enviados.

O papel do formulário é enviar os dados contidos nele para algum lugar, mas para onde? O atributo `action` é quem diz para onde os dados de um formulário deve ser enviado. Além disso, devemos

informar a maneira como queremos que esses dados sejam enviados, ou seja, se queremos que eles sejam enviados através de uma requisição do tipo GET ou POST (métodos de envio definidos no protocolo HTTP).

Exercícios de Fixação

15 Crie um documento HTML em um arquivo chamado **formulario.html** na pasta **html** com diversos elementos de formulário e para cada elemento crie um rótulo. Cada rótulo deve focar o elemento de formulário correspondente.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>A tag label e os elementos de formulário</title>
  </head>
  <body>
    <form action="pagina.html" method="get">
      <p>
        <label for="nome">Nome:</label>
        <input type="text" id="nome" />
      </p>
      <p>
        <label for="senha">Senha:</label>
        <input type="password" id="senha" />
      </p>
      <p>
        Sexo:
        <input type="radio" name="sexo" id="masculino" />
        <label for="masculino">Masculino</label>
        <input type="radio" name="sexo" id="feminino" />
        <label for="feminino">Feminino</label>
      </p>
      <p>
        <label for="mensagem">Mensagem:</label>
        <textarea id="mensagem"></textarea>
      </p>
    </form>
  </body>
</html>
```

Código HTML 2.50: formulario.html

Exercícios Complementares

14 Em um documento HTML, crie um formulário que utilize as tags `input`, `select`, `label` e `textarea`. Tente utilizar todos os tipos do elemento `input` vistos neste capítulo.

CAPÍTULO 3

CSS

Até o momento, utilizamos os elementos HTML sem modificar a forma de exibição dos mesmos. Nos exemplos mostrados no Capítulo 2, os elementos foram exibidos com a formatação padrão definida pelo navegador utilizado.

A formatação padrão pode variar de navegador para navegador. Em geral, os navegadores tentam seguir as sugestões do W3C. Mas, algumas vezes, erros de interpretação da especificação ou erros de implementação podem ocorrer. Além disso, o W3C pode apenas sugerir, ele não possui o poder de obrigar que todos os navegadores tenham o mesmo comportamento. Portanto, é uma boa prática formatarmos cada elemento para que o efeito visual seja o mesmo em todos os navegadores. E esse não é o único motivo, pois na grande maioria das vezes, desejamos aplicar em nossas páginas um design único, com a identidade visual da nossa empresa ou cliente.

Os elementos HTML possuem alguns atributos para formatarmos a sua aparência. Porém, além de serem limitados, o uso desses atributos estão caindo em desuso. Inclusive, existem elementos cuja única função é alterar a aparência de um texto, por exemplo. Contudo, esses elementos também caíram em desuso e provavelmente não estarão nas próximas especificações do HTML.

Para alterarmos o aspecto visual dos elementos do HTML, o W3C recomenda que utilizemos o CSS (Cascading Style Sheets - Folhas de Estilo em Cascata). Atualmente o CSS encontra-se em sua terceira versão. Porém, nem todos os navegadores implementaram todos os novos recursos.

Podemos aplicar o CSS de três formas em um documento HTML:

- Definindo as propriedades CSS diretamente no elemento HTML através do seu **atributo** `style`.
- Definindo as regras CSS dentro de um elemento com a **tag** `style`.
- Definindo as regras CSS em arquivo à parte do documento HTML.

Mas o que são essas regras e propriedades? Como elas são definidas?

Propriedade CSS é uma característica visual de um elemento HTML. Já a regra é um conjunto de propriedades definidas para um elemento ou para um grupo de elementos HTML.

Vamos ver um exemplo de aplicação das propriedades CSS diretamente em um elemento HTML:

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exemplo de CSS diretamente em um elemento</title>
5    </head>
6      <body>
7        <p style="font-size: 40px; color: #ff0000">Olá mundo!</p>
8        <p>Olá mundo novamente!</p>
9    </body>
10 </html>
```

Código HTML 3.1: Exemplo de aplicação das propriedades CSS inline

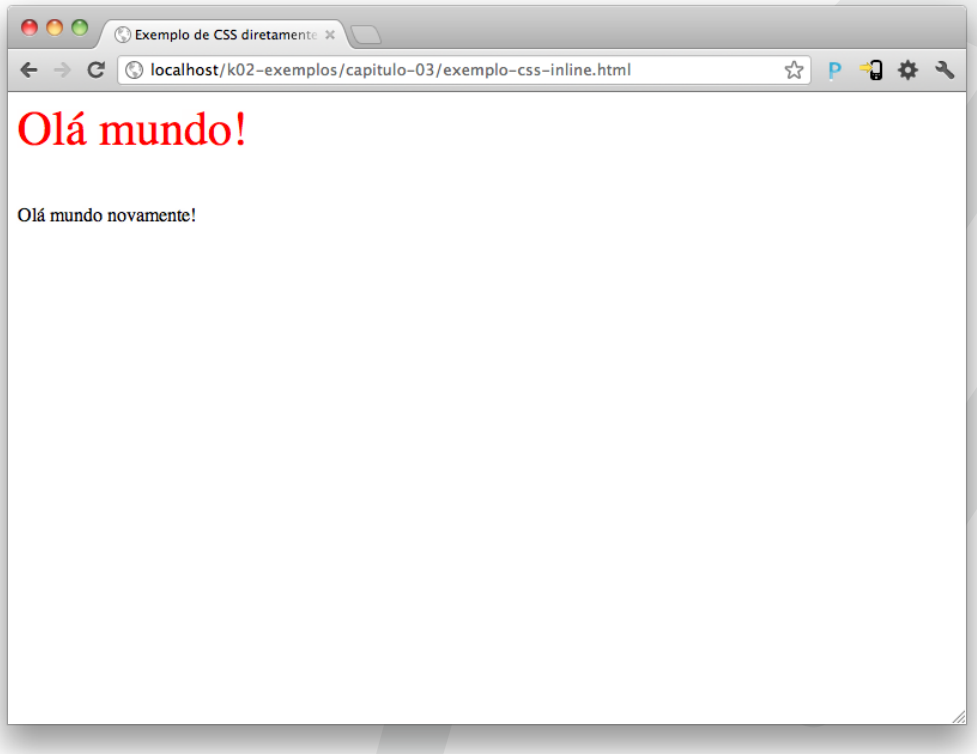

Figura 3.1: Exemplo de aplicação das propriedades CSS inline

Quando aplicamos as propriedades CSS diretamente a um elemento através da propriedade `style`, estamos utilizando a abordagem *CSS inline*. Essa prática não é recomendada, pois dessa forma não é possível reaproveitar o código CSS, além de dificultar a leitura do código HTML.

Vamos ver agora a aplicação das regras CSS utilizando a tag `style`:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo de aplicação das regras CSS através da tag style</title>
    <style type="text/css">
      p {
        font-size: 40px;
        color: #ff0000;
      }
    </style>
  </head>
  <body>
    <p>Olá mundo!</p>
    <p>Olá mundo novamente!</p>
  </body>
</html>
```

Código HTML 3.2: Exemplo de aplicação das regras CSS através da tag `style`

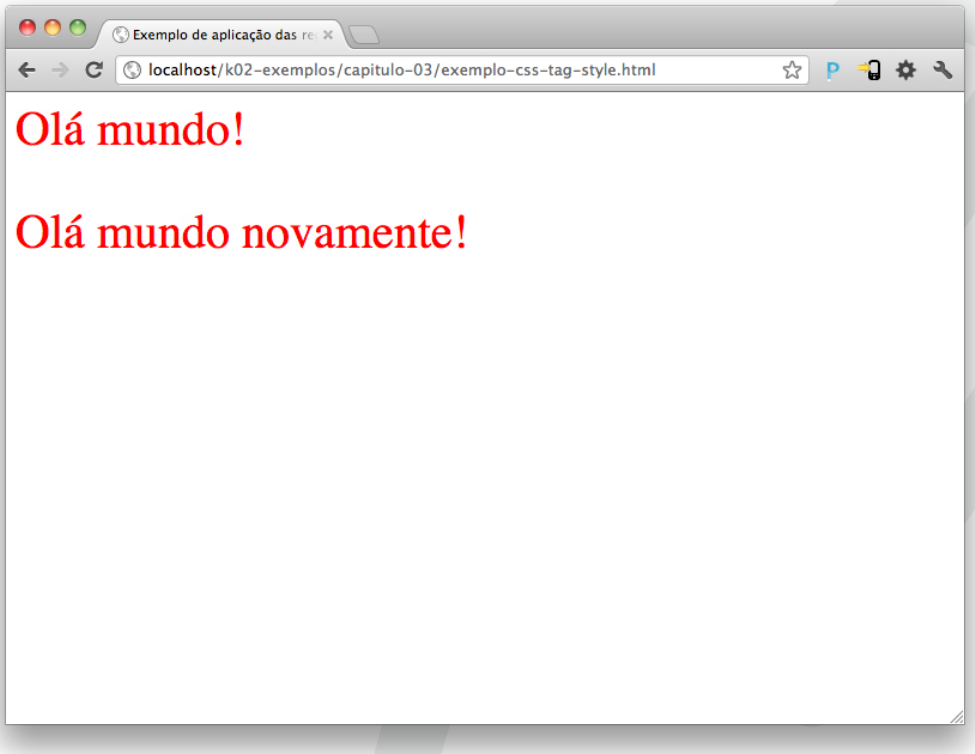

Figura 3.2: Exemplo de aplicação das regras CSS através da tag `style`

Como vimos anteriormente, também podemos definir as regras CSS em um arquivo à parte. Com esse arquivo em mãos, dentro de um documento HTML, para indicarmos qual o arquivo que contém as regras CSS, devemos utilizar a tag `link`. Veja o exemplo:

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exemplo de aplicação das regras CSS através de um arquivo</title>
5      <link rel="stylesheet" type="text/css" href="estilo.css"/>
6    </head>
7    <body>
8      <p>Olá mundo!</p>
9      <p>Olá mundo novamente!</p>
10   </body>
11 </html>
```

Código HTML 3.3: Exemplo de aplicação das regras CSS através de um arquivo

```
1  p {
2    font-size: 40px;
3    color: #ff0000;
4  }
```

Código CSS 3.1: estilo.css

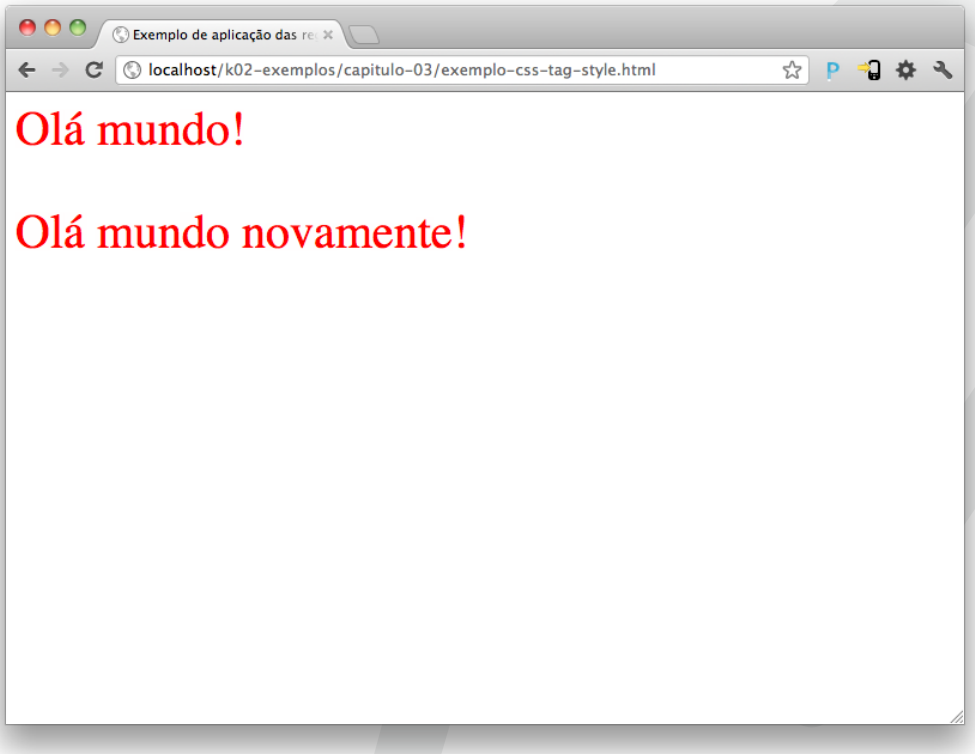

Figura 3.3: Exemplo de aplicação das regras CSS através de um arquivo

Perceba que o efeito foi o mesmo de quando aplicamos as regras CSS através da tag `style`. Isso se deve ao fato de estarmos utilizando a mesma regra. O que mudamos foi apenas onde a definimos.

Exercícios de Fixação

1 Dentro da pasta com o seu nome utilizada no capítulo anterior, crie uma subpasta chamada **css**. Novamente para facilitar, utilize apenas letras minúsculas em todas as pastas e arquivos que criarmos durante esse capítulo.

2 Crie um documento HTML em um arquivo chamado **testando-css.html** dentro da pasta **css**. Todos os parágrafos desse documento devem ser exibidos em negrito, com cor azul e com fonte 20px. Defina uma regra CSS para formatar os parágrafos.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Testando CSS</title>
    <style type="text/css">
      p {
        font-weight: bold;
        font-size: 20px;
        color: #0000FF;
      }
```

```
    </style>
  </head>
  <body>
    <p>Um parágrafo formatado</p>
    <p>Outro parágrafo formatado</p>
  </body>
</html>
```

Código HTML 3.4: testando-css.html

3 Para organizar melhor o conteúdo e a formatação da página criada no exercício anterior, crie um arquivo CSS chamado **regras-de-formatacao.css** na pasta **css**.

```
p {
  font-weight: bold;
  font-size: 20px;
  color: #0000FF;
}
```

Código CSS 3.2: regras-de-formatacao.css

4 Modifique o arquivo **testando-css.html** para aplicar as regras de formatação criadas no exercício anterior.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Testando CSS</title>
    <link rel="stylesheet" type="text/css" href="regras-de-formatacao.css"/>
  </head>
  <body>
    <p>Testando o parágrafo com o CSS</p>
    <p>Testando novamente o parágrafo com o CSS</p>
  </body>
</html>
```

Código HTML 3.5: testando-css.html

Estrutura de uma regra CSS

Uma regra CSS é composta por três partes como podemos observar na imagem abaixo:

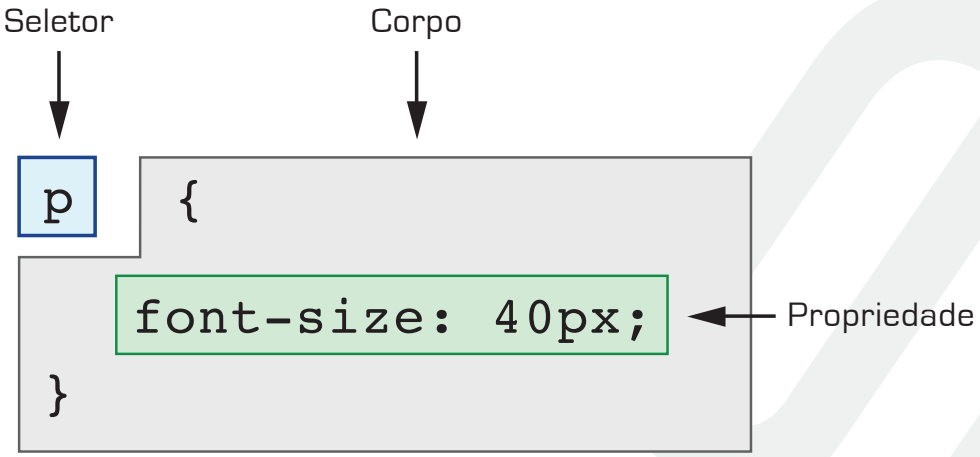

Figura 3.4: Estrutura de uma regra CSS

Podemos ler a regra acima da seguinte forma: será atribuído o valor `40px` à `propriedade` `font-size` em todos os elementos que forem selecionados pelo `seletor` `p`.

No exemplo acima, utilizamos aquilo que chamamos de `seletor de tipo`. Todos os elementos `p` receberão as propriedades definidas no corpo da regra.

Tipos de seletores

Na linguagem CSS existem diversos tipos de seletores, sendo os principais:

- Seletor universal
- Seletor de tipo
- Seletor de descendentes
- Seletor de filhos
- Seletor de irmão adjacente
- Seletor de atributos
- Seletor de id
- Seletor de classe

Seletor universal

O seletor universal seleciona todos os elementos de um documento HTML.

No exemplo a seguir, faremos com que todos os elementos tenham margem igual a `0px`.

```
1 * {
2   margin: 0px;
3 }
```

Código CSS 3.3: Usando o seletor universal

Seletor de tipo

O seletor de tipo seleciona todos os elementos com tag igual ao tipo indicado na declaração da regra CSS.

No exemplo a seguir, selecionaremos todos os elementos que possuem a tag `textarea`.

```
1 textarea {
2   border: 1px solid red;
3 }
```

Código CSS 3.4: Usando o seletor de tipo

Seletor de descendentes

Chamamos de seletor de descendentes a seleção de um ou mais elementos fazendo o uso de outros seletores separados por espaços. Um espaço indica que os elementos selecionados pelo seletor à sua direita são **filhos diretos ou indiretos** dos elementos selecionados pelo seletor à sua esquerda.

No exemplo a seguir iremos selecionar todos os elementos que possuem a tag `input` e que sejam filhos diretos ou indiretos de elementos com a tag p.

```
1 p input {
2   border: 1px solid red;
3 }
```

Código CSS 3.5: Usando o seletor de descendentes

Seletor de filhos

Chamamos de seletor de filhos a seleção de um ou mais elementos fazendo o uso de outros seletores separados pelo caractere >. Um caractere > indica que os elementos selecionados pelo seletor à sua direita são **filhos diretos** dos elementos selecionados pelo seletor à sua esquerda.

No exemplo a seguir iremos selecionar todos os elementos que possuem a tag a e que sejam filhos diretos de elementos com a tag p.

```
1 p > a {
2   color: green;
3 }
```

Código CSS 3.6: Usando o seletor de filhos

Seletor de irmão adjacente

Chamamos de seletor de irmão adjacente a seleção de um ou mais elementos fazendo o uso de outros seletores separados pelo caractere +. Um caractere + indica que os elementos selecionados pelo seletor à sua direita sejam **irmãos e imediatamente precedidos** pelos elementos selecionados pelo seletor à sua esquerda.

No exemplo a seguir iremos selecionar todos os elementos que possuem a tag `input` e que sejam irmãos e imediatamente precedidos por elementos com a tag `label`.

```
1 label + input {
2   color: green;
```

```
3 }
```

Código CSS 3.7: Usando o seletor de irmão adjacente

Seletor de atributos

O seletor de atributos seleciona um ou mais elementos que possuam o atributo ou o atributo juntamente com o seu valor da mesma forma como é indicada pelo seletor na declaração da regra CSS. Os atributos são informados dentro de colchetes `[]`.

No exemplo abaixo iremos selecionar todos os elementos que possuam o atributo `name` igual a `cidade`.

```
1 *[name=cidade] {
2   font-weight: italic;
3 }
```

Código CSS 3.8: Usando o seletor de atributos

Se desejarmos também podemos informar apenas o atributo. No exemplo a seguir iremos selecionar todos os elementos com a tag `img` que possuam o atributo `title`.

```
1 img[title] {
2   width: 100px;
3 }
```

Código CSS 3.9: Usando o seletor de atributos

Seletor de id

O seletor de id seleciona um único elemento cujo atributo `id` possui o valor indicado pelo seletor na declaração da regra CSS.

No exemplo abaixo iremos selecionar o elemento cujo atributo `id` possui o valor `cidade`. Repare que o seletor de id começa com o caractere #.

```
1 #cidade {
2   font-weight: bold;
3 }
```

Código CSS 3.10: Usando o seletor de id

Seletor de classe

O seletor de classe seleciona todos os elementos cujo atributo `class` possui o valor indicado pelo seletor na declaração da regra CSS.

No exemplo abaixo iremos selecionar todos os elementos cujo atributo `class` possui o valor `títulos`. Repare que o seletor de classe começa com o caractere `.` (ponto).

```
1 .titulos {
2   font-size: 40px;
3 }
```

Código CSS 3.11: Usando o seletor de classe

Exercícios de Fixação

5 Para testar o funcionamento de alguns seletores, crie um documento html em um arquivo chamado **seletores.html**. Além disso, defina algumas regras de formatação em um arquivo CSS chamado **seletores.css**. Salve esses arquivos na pasta **css**.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="seletores.css" />
  </head>
  <body>
    <ul id="main-menu">
      <li><a class="selected" href="http://www.k19.com.br/cursos">Cursos</a></li>
      <li><a href="http://www.k19.com.br/downloads/apostilas">Apostilas</a></li>
      <li><a href="http://www.k19.com.br/artigos/">Artigos</a></li>
      <li><a href="http://www.k19.com.br/sobre-a-k19">Sobre</a></li>
    </ul>

    <h1>K19 - Treinamentos</h1>

    <h3>Curso Java, C# e ASP.NET MVC? Conheça os Cursos da K19</h3>

    <p>
      Para saber mais informações acesse
      <a href="http://www.k19.com.br">www.k19.com.br</a>
    </p>
  </body>
</html>
```

Código HTML 3.6: seletores.html

```
* {
  font-family: Monaco, 'DejaVu Sans Mono', monospace;
}

h1 {
  color: blue;
}

h3 {
  font-size: 20px;
}

p > a {
  font-weight: bold;
  text-decoration: none;
}

#main-menu {
  font-size: 18px;
  list-style: none;
  padding: 0px;
}

#main-menu li {
  display: inline;
    margin: 0 50px 0 0;
}

#main-menu li a {
```

```
30    border: 2px solid #0000ff;
31    text-decoration: none;
32  }
33
34  .selected {
35    background-color: blue;
36    color: white;
37  }
```

Código CSS 3.12: formatando-pagina.css

Visualize a página HTML através de um navegador

Cores em CSS

Tradicionalmente, as cores podem ser definidas de três formas diferentes: pelo nome da cor, pelo valor hexadecimal associado à cor ou através da função `rgb()`. Nem todas as cores possuem um nome. Por isso, normalmente, utilizamos a forma hexadecimal ou a função `rgb()`.

Uma cor é definida em hexadecimal da seguinte forma: `#RRGGBB`. Onde `RR`, `GG` e `BB` podem variar de `00` a `FF` e representam os componentes vermelho, verde e azul de uma cor.

Para definir uma cor utilizando a função `rgb()`, é necessário passar como argumento as "quantidades" de vermelho, verde e azul necessárias para formar a cor desejada. Essas quantidade podem ser expressas na escala de 0 a 255 ou em porcentagem. Veja os exemplos abaixo:

```
1  color: rgb(20, 255, 40);
2  background-color: rgb(10%, 80%, 40%);
```

Mais Sobre

Em CSS3, há mais três funções para definir uma cor: `rgba()`, `hsl()` e `hsla()`.

A função `hsl()` define as cores através da matiz, saturação e luminosidade (hue, saturation e lightness). Essa função possui três parâmetros `hsl(H, S, L)`. Onde H pode variar de `0` a `360`, S e L de `0%` a `100%`.

As funções `rgba(R, G, B, A)` e `hsla(H, S, L, A)` possuem um último parâmetro que significa a opacidade da cor (alpha). Esse valor varia de `0` a `1` com passo de `0.1`.

Unidades de medida

Em CSS possuímos diversas unidades de medida como podemos verificar na listagem abaixo:

- in - polegada.
- cm - centímetro.
- mm - milímetro.
- em - tamanho relativo ao tamanho de fonte atual no documento. 1em é igual ao tamanho da fonte atual, 2em o dobro do tamanho da fonte atual e assim por diante.
- ex - essa unidade é igual à altura da letra "x"minúscula da fonte atual do documento.

- pt - ponto (1pt é o mesmo que 1/72 polegadas).
- px - pixels (um ponto na tela do computador).

Da lista acima, as unidades mais utilizadas são `px` e `em`.

Principais propriedades CSS

Propriedades de background

- background-attachment - define se a imagem de background deve se mover com a rolagem de um elemento ou não.
- background-color - define a cor do background de um elemento.
- background-image - define a imagem de background de um elemento.
- background-position - define a posição do background de um elemento.
- background-repeat - define se o background de um elemento de se repetir caso este seja menor que a parte visível do elemento.
- background - define todas as propriedades de background em uma única linha.

```
1  body {
2    background-attachment: fixed;
3    background-color: #dddddd;
4    background-image: url('http://www.k19.com.br/css/img/main-header-logo.png');
5    background-position: left top;
6    background-repeat: repeat;
7  }
8
9  div {
10   background: #dddddd url('http://www.k19.com.br/css/img/main-header-logo.png')
11   no-repeat center center fixed;
12 }
```

Código CSS 3.14: Propriedades de background

Propriedades de texto

- color - define a cor do texto.
- direction - define a direção do texto.
- letter-spacing - define o espaçamento entre as letras do texto.
- line-height - define a altura das linhas de um texto.
- text-align - define o alinhamento horizontal do texto.
- text-decoration - define uma "decoração"ou efeito para um texto.
- text-indent - define a identação da primeira linha de um bloco de texto.
- text-transform - define a capitalização do texto.
- vertical-align - define o alinhamento vertical do texto.
- white-space - define como os espaços do texto serão tratados.

- word-spacing - define o espaçamento entre as palavras do texto.

```
p {
  color: green;
  direction: rtl;
  letter-spacing: 10px;
  line-height: 30px;
  text-align: right;
  text-decoration: blink;
  text-indent: 50px;
  text-transform: uppercase;
  vertical-align: middle;
  white-space: nowrap;
  word-spacing: 30px;
}
```

Código CSS 3.15: Propriedades de texto

Propriedades de fonte

- font-family - define a família de fontes a ser utilizada.
- font-size - define o tamanho da fonte.
- font-style - define o estilo de fonte.
- font-variant - define se a fonte será utilizada no formato small-caps ou não.
- font-weight - define a espessura dos traços de uma fonte.
- font - define todas as propriedades de fonte em uma única linha.

```
p {
  font-family: sans-serif, serif, monospace;
  font-size: 14px;
  font-style: italic;
  font-variant: small-caps;
  font-weight: bold;
}

a {
  font: italic small-caps bold 14px/20px sans-serif, serif, monospace;
}
```

Código CSS 3.16: Propriedades de fonte

Propriedades de lista

- list-style-image - define qual será o indicador de item da lista.
- list-style-position - define se o indicador de item da lista será exibido do lado de dentro ou de fora do elemento do item.
- list-style-type - define qual o tipo de indicador de item será usado na lista.
- list-style: define todas as propriedades de lista em uma única linha.

```
ul {
  list-style-image: url('http://www.k19.com.br/css/img/box-dot-orange.png');
  list-style-position: inside;
  list-style-type: disc;
}
```

```
6
7  ol {
8    list-style: square outside
9    url('http://www.k19.com.br/css/img/box-dot-orange.png');
10 }
```

Código CSS 3.17: Propriedades de lista

Propriedades de tabela

- border-collapse - faz com que as bordas das células não fiquem duplicadas quando estas possuírem bordas.

```
1  table {
2    border-collapse: collapse;
3  }
4  table,th, td {
5    border: 1px solid blue;
6  }
```

Código CSS 3.18: Propriedades de tabela

Propriedades de dimensão

- width - define a largura de um elemento.
- min-width - define a largura mínima de um elemento.
- max-width - define a largura máxima de um elemento.
- height - define a altura de um elemento.
- min-height - define a altura mínima de um elemento.
- max-height - define a altura máxima de um elemento.

```
1  div {
2    width: 300px;
3    height: 300px;
4  }
5
6  h1 {
7    min-width: 10px;
8    max-width: 300px;
9    min-height: 10px;
10   max-height: 300px;
11 }
```

Código CSS 3.19: Propriedades de dimensão

Exercícios de Fixação

6 A partir de agora, o nosso objetivo é formatar a página principal do blog da K19. Ao término dos exercícios, essa página deve ficar igual à ilustração abaixo.

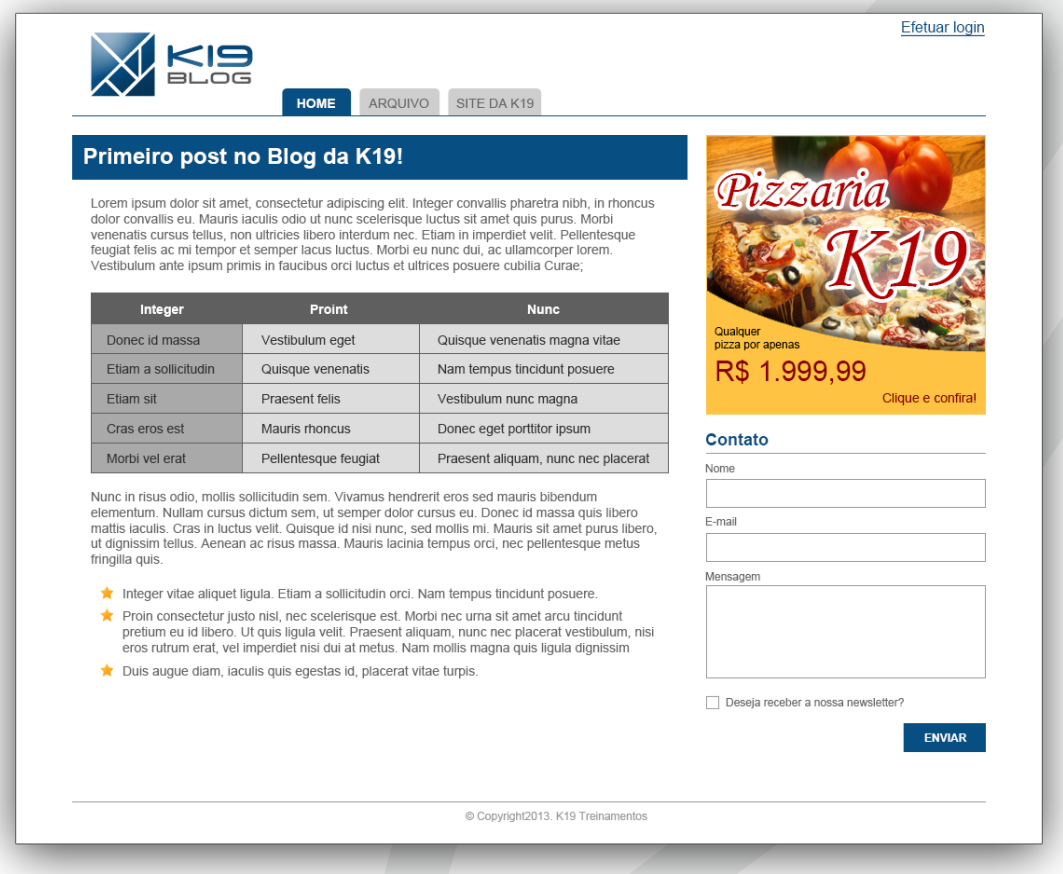

Figura 3.5: Blog Layout

O primeiro passo é criar o documento HTML com o conteúdo desejado. Faça um arquivo chamado **index.html** na pasta **css** com o seguinte conteúdo.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="blog.css" />
  </head>
  <body>
    <div id="blog">
      <div id="header">
        <a id="logo-link" href="#">
          <img src="logo.png"
             alt="Tangram representando o logo do blog da K19"
             title="K19 Blog" />
        </a>

        <ul id="main-menu">
          <li><a class="selected" href="#">Home</a></li>
          <li><a href="#">Arquivo</a></li>
          <li><a href="#">Site da K19</a></li>
        </ul>

        <div id="login-area">
          <a href="#">Efetuar login</a>
        </div>
```

```
26      </div>
27
28      <div id="main-content">
29        <h1>Primeiro post no Blog da K19!</h1>
30
31        <div id="blog-content">
32          <p>
33            Lorem ipsum dolor sit amet, consectetur adipiscing elit.
34            Integer convallis pharetra nibh, in rhoncus dolor convallis eu.
35            Mauris iaculis odio ut nunc scelerisque luctus sit amet quis purus.
36            Morbi venenatis cursus tellus, non ultricies libero interdum nec.
37            Etiam in imperdiet velit. Pellentesque feugiat felis ac mi tempor
38            et semper lacus luctus. Morbi eu nunc dui, ac ullamcorper lorem.
39            Vestibulum ante ipsum primis in faucibus orci luctus et ultrices
40            posuere cubilia Curae;
41          </p>
42
43          <table>
44            <thead>
45              <tr>
46                <th>Integer</th>
47                <th>Proint</th>
48                <th>Nunc</th>
49              </tr>
50            </thead>
51            <tbody>
52              <tr>
53                <td class="first-col">Donec id massa</td>
54                <td>Vestibulum eget</td>
55                <td>Quisque venenatis magna vitae</td>
56              </tr>
57              <tr>
58                <td class="first-col">Etiam a sollicitudin</td>
59                <td>Quisque venenatis</td>
60                <td>Nam tempus tincidunt posuere</td>
61              </tr>
62              <tr>
63                <td class="first-col">Etiam sit</td>
64                <td>Praesent felis</td>
65                <td>Vestibulum nunc magna</td>
66              </tr>
67              <tr>
68                <td class="first-col">Cras eros est</td>
69                <td>Mauris rhoncus </td>
70                <td>Donec eget porttitor ipsum</td>
71              </tr>
72              <tr>
73                <td class="first-col">Morbi vel erat</td>
74                <td>Pellentesque feugiat</td>
75                <td>Praesent aliquam, nunc nec placerat</td>
76              </tr>
77            </tbody>
78          </table>
79
80          <p>
81            Nunc in risus odio, mollis sollicitudin sem. Vivamus hendrerit
82            eros sed mauris bibendum elementum. Nullam cursus dictum sem,
83            ut semper dolor cursus eu. Donec id massa quis libero mattis
84            iaculis. Cras in luctus velit. Quisque id nisi nunc, sed mollis
85            mi. Mauris sit amet purus libero, ut dignissim tellus. Aenean
86            ac risus massa. Mauris lacinia tempus orci, nec pellentesque metus
87            fringilla quis.
88          </p>
89
90          <ul>
91            <li>
92              Integer vitae aliquet ligula. Etiam a sollicitudin orci. Nam tempus
93              tincidunt posuere.
94            </li>
95            <li>
```

```
96                Proin consectetur justo nisl, nec scelerisque est. Morbi nec
97                urna sit amet arcu tincidunt pretium eu id libero. Ut quis ligula
98                velit. Praesent aliquam, nunc nec placerat vestibulum, nisi eros
99                rutrum erat, vel imperdiet nisi dui at metus. Nam mollis magna quis
100               ligula dignissim.
101             </li>
102             <li>
103               Duis augue diam, iaculis quis egestas id, placerat vitae turpis.
104             </li>
105           </ul>
106         </div>
107       </div>
108
109       <div id="side-content">
110         <a id="ad" href="#">
111           <img src="publicidade.png"
112              alt="Propaganda da Pizzaria K19 com a imagem de uma pizza ao fundo"
113              title="Pizzaria K19" />
114         </a>
115
116         <form id="contact-form" action="#">
117           <h3>Contato</h3>
118
119           <div>
120             <label for="contact-form-name">Nome</label>
121             <input id="contact-form-name" type="text" name="nome" />
122           </div>
123
124           <div>
125             <label for="contact-form-email">E-mail</label>
126             <input id="contact-form-email" type="text" name="nome" />
127           </div>
128
129           <div>
130             <label for="contact-form-message">Mensagem</label>
131             <textarea id="contact-form-message"></textarea>
132           </div>
133
134           <div>
135             <input id="contact-form-newsletter" type="checkbox" name="newsletter" />
136             <label
137               for="contact-form-newsletter">Deseja receber a nossa newsletter?</label>
138           </div>
139
140           <input type="submit" value="Enviar" />
141         </form>
142       </div>
143
144       <div id="footer">
145         &copy; Copyright 2013. K19 Treinamentos.
146       </div>
147     </div>
148   </body>
149 </html>
```

Código HTML 3.7: k19-blog.html

7 Copie os arquivos **logo.png**, **publicidade.png** e **list-bullet.png** da pasta **K19-Arquivos/imagens/k02** da sua área de trabalho para a pasta **css**.

8 Para definir a formatação da página que desejamos criar, faça um arquivo chamado **blog.css** na pasta **css** com o seguinte conteúdo.

```
1 * {
```

```
2     font-family: arial, sans-serif;
3   }
4
5   body {
6     white-space: nowrap;
7   }
8
9   #blog {
10    width: 980px;
11  }
12
13  #logo-link {
14    text-decoration: none;
15  }
16
17  #logo-link img {
18    vertical-align: bottom;
19  }
20
21  #main-menu {
22    list-style: none;
23  }
24
25  #main-menu li a {
26    background: #cecece;
27    color: #333333;
28    text-decoration: none;
29  }
30
31  #main-menu li a.selected {
32    background: #074f83;
33    color: #ffffff;
34  }
35
36  #main-content {
37    vertical-align: top;
38    width: 660px;
39    color: #666666;
40    white-space: normal;
41  }
42
43  #main-content h1 {
44    background: #074f83;
45    color: #ffffff;
46    font-size: 24px;
47  }
48
49  #blog-content > table {
50    width: 100%;
51  }
52
53  #blog-content > table th {
54    color: #ffffff;
55    background: #5f5f5f;
56  }
57
58  #blog-content > table td {
59    background: #dddddd;
60    color: #333333;
61  }
62
63  #blog-content > table td.first-col {
64    background: #a9a9a9;
65  }
66
67  #blog-content ul {
68    list-style-image: url('list-bullet.png');
69  }
70
71  #side-content {
```

```
72    vertical-align: top;
73  }
74
75  #contact-form {
76    text-align: right;
77  }
78
79  #contact-form h3 {
80    font-size: 18px;
81    color: #074f83;
82    text-align: left;
83  }
84
85  #contact-form div {
86    text-align: left;
87  }
88
89  #contact-form label {
90    font-size: 14px;
91    vertical-align: middle;
92    color: #666666;
93  }
94
95  #contact-form input,
96  #contact-form textarea {
97    width: 288px;
98    font-size: 18px;
99  }
100
101 #contact-form textarea {
102   height: 100px;
103 }
104
105 #contact-form input[type='checkbox'] {
106   width: auto;
107 }
108
109 #contact-form input[type='submit'] {
110   background: #074f83;
111   color: #ffffff;
112   text-transform: uppercase;
113   font-size: 14px;
114   width: auto;
115 }
116
117 #footer {
118   font-size: 12px;
119   color: #999999;
120   text-align: center;
121 }
```

Código CSS 3.20: blog.css

- Linhas 1 a 3: Selecionamos, através da propriedade `font-family` e do seletor universal (*), as famílias de fontes que devem ser utilizadas para todos os elementos HTML. No caso, as famílias selecionadas foram arial e sans-serif.
- Linhas 5 a 7: Evitamos, através da propriedade `white-space`, que uma quebra de linha dentro do corpo do documento HTML seja interpretada como espaço em branco.
- Linhas 9 a 11: Definimos, através da propriedade `width`, que o conteúdo do blog terá uma largura de 980px.
- Linhas 13 a 15: Definimos, através da propriedade `text-decoration`, que o texto do logo não deve receber nenhum tipo de decoração.

- Linhas 17 a 19: Definimos, através da propriedade `vertical-align`, que a imagem do logo deve ser alinhada inferiormente aos elementos da mesma linha.
- Linhas 21 a 23: Definimos, através da propriedade `list-style`, que nenhum símbolo deve ser utilizado nos itens do menu principal.
- Linhas 25 a 29: Definimos, através das propriedades `background` e `color`, as cores #cecece e #333333 para o fundo e para o texto dos links do menu principal respectivamente. Além disso, retiramos as decorações desses links através da propriedade `text-decoration`.
- Linhas 31 a 33: Definimos, através das propriedades `background` e `color`, as cores #074f83 e #ffffff para o fundo e para o texto do link selecionado do menu principal respectivamente
- Linhas 36 a 41: Definimos que o conteúdo principal será alinhado verticalmente na parte de cima, terá largura de 660px e a cor do texto será #666666.
- Linhas 43 a 47: Definimos, através das propriedades `background`, `color` e `font-size`, que os elementos H1 do conteúdo principal terão cor de fundo #074f83 e fonte de cor #ffffff com tamanho 24px.
- Linhas 49 a 51: Definimos, através da propriedade `width`, que toda tabela filha de conteúdo do blog terá largura de 100
- Linhas 53 a 56: Definimos, através da propriedade `background` e `color`, as cores #ffffff e #5f5f5f para o texto e fundo dos títulos das colunas das tabelas filhas do conteúdo do blog respectivamente.
- Linhas 58 a 61: Definimos, através da propriedade `background` e `color`, as cores #333333 e #dddddd para o texto e fundo das células das tabelas filhas do conteúdo do blog respectivamente.
- Linhas 63 a 65: Definimos, através da propriedade `background`, a cor #a9a9a9 para o fundo das células da primeira coluna das tabelas filhas do conteúdo do blog.
- Linhas 67 a 69: Definimos, através da propriedade `list-style-image`, a imagem que deve ser utilizada como símbolo dos itens de todas as listas sem ordem do conteúdo do blog.
- Linhas 71 a 73: Definimos, através da propriedade `vertical-align`, que o conteúdo lateral será alinhado verticalmente na parte de cima.
- Linhas 75 a 77: Definimos, através da propriedade `text-align`, que o texto do formulário de contato será alinhado à direita.
- Linhas 79 a 83: Definimos, através das propriedades `font-size` `color` e `text-align`, que todo título nível 3 dentro do formulário de contato terá fonte com tamanho 18px e cor #074f83 e o texto será alinhado à esquerda.
- Linhas 85 a 87: Definimos, através da propriedade `text-align`, que todo elemento DIV dentro do formulário de contato terá o texto alinhado à esquerda
- Linhas 89 a 93: Definimos, através das propriedades `font-size`, `vertical-align` e `color`, que todo rótulo dentro do formulário de contato terá fonte com tamanho 14px e cor #666666 e o alinhamento vertical do texto será no centro.
- Linhas 95 a 99: Definimos, através das propriedades `width` e `font-size`, a largura e o tamanho da fonte dos elementos INPUT e TEXTAREA do formulário de contato.
- Linhas 101 a 103: Definimos, através da propriedade `height`, a altura dos elementos TEXTAREA do formulário de contato.
- Linhas 105 a 107: Definimos, através da propriedade `width`, a largura dos checkboxes do formulário de contato.

- Linhas 109 a 115: Definimos, através das propriedades `background`, `color`, `text-transform` e `font-size`, a cor do fundo, a cor da fonte, o tipo de letra, o tamanho da fonte e largura do botão de envio do formulário de contato.
- Linhas 117 a 121: Definimos, através das propriedades `font-size`, `color` e `text-align`, que o o texto do rodapé terá fonte de tamanho 12px, cor #999999 e alinhamento centralizado.

Box model

O termo *box model* é utilizado para explicar o comportamento visual dos elementos HTML, pois podemos imaginar que cada elemento em uma página está envolvido por uma caixa. Essa caixa possui três partes: uma margem interna (`padding`), uma borda (`border`) e uma margem externa (`margin`).

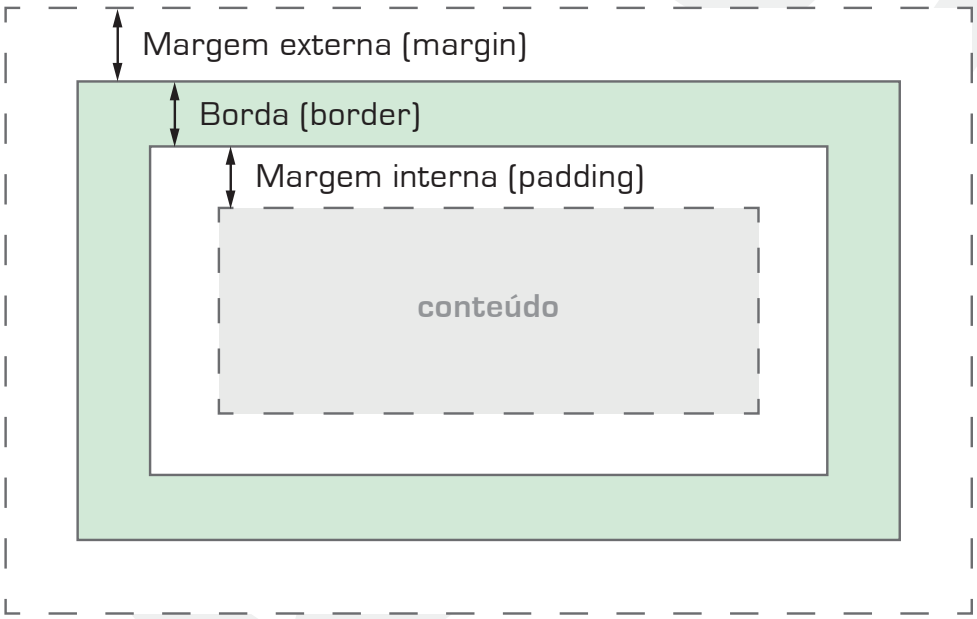

Figura 3.6: Box model

Um erro muito comum quando estamos começando a aprender CSS é que nos esquecemos de considerar as dimensões das margens internas e externas no cálculo das dimensões de um elemento.

Vamos pensar no seguinte caso: suponha que você possua um espaço de `300px` para encaixar um conteúdo dentro da sua página. Você poderia incluir no HTML um elemento com a tag `div` e a seguinte regra CSS:

```
1  div {
2    width: 300px;
3    padding: 10px;
4    margin: 10px;
5    border: 1px solid green;
6  }
```

Código CSS 3.21: Exemplo de uso incorreto das dimensões de um elemento

Num primeiro momento pode parecer que tudo está correto, porém ao abrir a página você perceberá que seu elemento está ultrapassando o limite dos `300px`. Isso ocorre porque devemos incluir

as margens internas, as margens externas e a borda na hora de calcular as dimensões finais de um elemento. No exemplo acima, o correto seria:

```
div {
  width: 258px;
  padding: 10px;
  margin: 10px;
  border: 1px solid green;
}
```

Código CSS 3.22: Exemplo de uso correto das dimensões de um elemento

O comportamento do *box model* é facilmente observado em elementos de bloco. Em elementos inline o *box model* continua valendo, porém sempre devemos nos lembrar que eles não sofrem os efeitos das propriedades `width` e `height`. Além disso, as propriedades `margin-top`, `margin-bottom`, `padding-top` e `padding-bottom` não afetam o posicionamento do conteúdo ao redor do elemento inline com tais propriedades. Observe o exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">
      span {
        border: 1px solid red;
        padding: 20px;
        margin: 40px;
      }
    </style>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
    massa, sodales sit amet eleifend a, elementum eu nibh. Lorem ipsum dolor
    sit amet, consectetur adipiscing elit. Donec justo massa, sodales sit amet
    eleifend a, elementum eu nibh. <span>Donec</span> egestas dolor quis turpis
    dictum tincidunt. Donec blandit tempus velit, sit amet adipiscing velit
    consequat placerat. Curabitur id mauris. Lorem ipsum dolor
    sit amet, consectetur adipiscing elit. Donec justo massa, sodales sit amet
    eleifend a, elementum eu nibh. Lorem ipsum dolor
    sit amet, consectetur adipiscing elit. Donec justo massa, sodales sit amet
    eleifend a, elementum eu nibh.</p>
  </body>
</html>
```

Código HTML 3.8: Box model em elemento inline

O resultado do código acima pode ser observado na imagem abaixo:

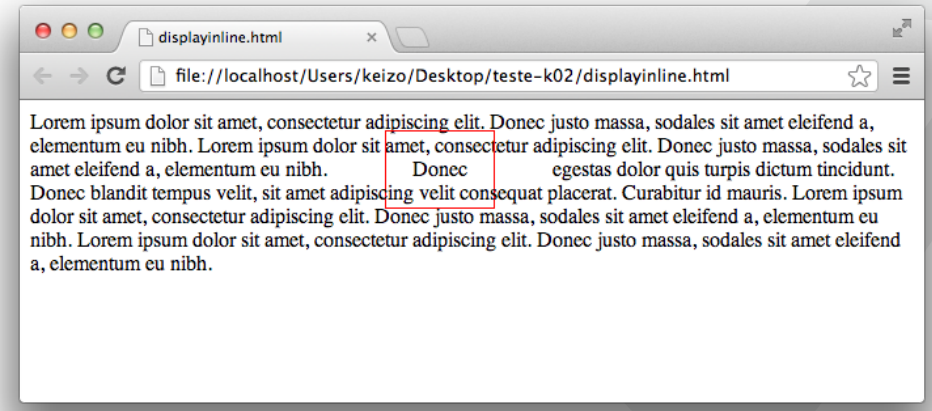

Figura 3.7: Box model em um elemento inline

A propriedade display

Todo elemento possui uma propriedade CSS chamada `display` que afeta a maneira como tal elemento será posicionado na tela. Essa propriedade pode assumir diversos valores, sendo os principais:

- block - o elemento que tiver a propriedade `display: block` fará com que o próximo elemento seja posicionado logo abaixo e terá a largura definida pelo atributo `width` ou herdará a largura do elemento que o contém. Chamamos esse tipo de elemento de *elemento de bloco*. Um elemento de bloco pode conter elementos de bloco, elementos inline e elementos inline-block.
- inline - o elemento que tiver a propriedade `display: inline` fará com que outros elementos inline (ou inline-block) ao seu redor sejam posicionados um do lado do outro, na ordem em que foram criados em um documento HTML. Chamamos esse tipo de elemento de *elemento inline*. Um elemento inline só pode conter outros elementos inline.
- inline-block - o elemento que tiver a propriedade `display: inline-block` fará com que outros elementos inline (ou inline-block) ao seu redor sejam posicionados um do lado do outro, na ordem em que foram criados em um documento HTML. Chamamos esse tipo de elemento de *elemento inline-block*. Um elemento inline-block pode conter elementos de bloco, elementos inline e outros elementos inline-block.

Durante a aplicação do CSS em uma página HTML muitas vezes temos a necessidade de inserir ou remover alguns elementos HTML afim de obtermos o resultado desejado. Isso é perfeitamente normal. Contudo, em algumas situações esse simples remanejamento dos elementos HTML não é o suficiente.

Em alguns momentos temos a necessidade de que um elemento de bloco se comporte como um elemento inline e vice-versa. Para atingirmos esse objetivo devemos utilizar a propriedade `display`, como nos exemplos a seguir:

```
div {
  display: inline;
}
```

Código CSS 3.23: Elemento de bloco se comportando como inline

```
span {
  display: block;
}
```

Código CSS 3.24: Elemento inline se comportando como bloco

Num primeiro momento temos a falsa impressão de que podemos utilizar essa propriedade livremente sem nos preocuparmos com os valores originais da propriedade `display` dos elementos HTML.

De acordo com a especificação do HTML, elementos inline não podem conter elementos de bloco. Já elementos de bloco podem conter tanto elementos inline assim como elementos de bloco.

Portanto, no primeiro exemplo, ao fazermos um `div` se comportar como um elemento inline, esse `div` não poderá conter elementos de bloco. Devemos prestar atenção nesse tipo de detalhe quando estivermos alterando as propriedades CSS de um elemento HTML.

Por outro lado, no segundo exemplo, apesar de fazermos um span se comportar como um elemento de bloco, não poderemos fazer com que ele contenha elementos de bloco ou inline-block. Isso se deve ao fato de que antes de qualquer regra CSS ser aplicada o documento já teria um erro de validação. Alterar a propriedade `display` de um elemento não significa que podemos desobedecer as regras estabelecidas na especificação do HTML.

Exercícios de Fixação

9 Continuando o exercício anterior, altere o arquivo **blog.css** conforme as linhas destacadas abaixo:

```
* {
  margin: 0;
  padding: 0;
  font-family: arial, sans-serif;
}

body {
  white-space: nowrap;
}

#blog {
  width: 980px;
  margin: 0 auto;
}

#header {
  border-bottom: 1px solid #074f83;
  margin: 0 0 20px 0;
}

#logo-link {
  text-decoration: none;
}
```

```
24
25  #logo-link img {
26    margin: 20px 40px 20px 20px;
27    vertical-align: bottom;
28  }
29
30  #main-menu {
31    list-style: none;
32    display: inline-block;
33  }
34
35  #main-menu li {
36    display: inline-block;
37    margin: 0 10px 0 0;
38  }
39
40  #main-menu li a {
41    display: block;
42    padding: 8px 10px 5px 10px;
43    background: #cecece;
44    color: #333333;
45    text-decoration: none;
46
47    border-top-left-radius: 5px; /* CSS3 */
48    border-top-right-radius: 5px; /* CSS3 */
49  }
50
51  #main-menu li a.selected {
52    background: #074f83;
53    color: #ffffff;
54  }
55
56  #main-content {
57    display: inline-block;
58    vertical-align: top;
59    width: 660px;
60    color: #666666;
61    white-space: normal;
62  }
63
64  #main-content h1 {
65    padding: 10px;
66    background: #074f83;
67    color: #ffffff;
68    font-size: 24px;
69  }
70
71  #blog-content {
72    padding: 20px;
73  }
74
75  #blog-content > p {
76    margin: 0 0 20px 0;
77  }
78
79  #blog-content > table {
80    border-collapse: collapse;
81    border: 1px solid #5f5f5f;
82    margin: 0 0 20px 0;
83    width: 100%;
84  }
85
86  #blog-content > table th,
87  #blog-content > table td {
88    border: 1px solid #5f5f5f;
89    padding: 10px 5px;
90  }
91
92  #blog-content > table th {
93    color: #ffffff;
```

```
94    background: #5f5f5f;
95  }
96
97  #blog-content > table td {
98    background: #dddddd;
99    color: #333333;
100 }
101
102 #blog-content > table td.first-col {
103   background: #a9a9a9;
104 }
105
106 #blog-content ul {
107   list-style-image: url('list-bullet.png');
108   margin: 0 0 0 30px;
109 }
110
111 #side-content {
112   display: inline-block;
113   margin: 0 0 0 20px;
114   vertical-align: top;
115 }
116
117 #ad {
118   display: block;
119   margin: 0 0 20px 0;
120 }
121
122 #contact-form {
123   text-align: right;
124 }
125
126 #contact-form h3 {
127   font-size: 18px;
128   color: #074f83;
129   text-align: left;
130   border-bottom: 1px solid #999999;
131   margin: 0 0 10px 0;
132   padding: 0 0 5px 0;
133 }
134
135 #contact-form div {
136   margin: 0 0 10px 0;
137   text-align: left;
138 }
139
140 #contact-form label {
141   font-size: 14px;
142   vertical-align: middle;
143   color: #666666;
144 }
145
146 #contact-form input,
147 #contact-form textarea {
148   display: block;
149   border: 1px solid #999999;
150   padding: 5px;
151   margin: 4px 0 0 0;
152   width: 288px;
153   font-size: 18px;
154 }
155
156 #contact-form textarea {
157   height: 100px;
158 }
159
160 #contact-form input[type='checkbox'] {
161   display: inline;
162   width: auto;
163 }
```

```
164
165 #contact-form input[type='submit'] {
166   margin: 10px 0 0 0;
167   border: none;
168   padding: 10px 20px;
169   display: inline;
170   background: #074f83;
171   color: #ffffff;
172   text-transform: uppercase;
173   font-size: 14px;
174   width: auto;
175 }
176
177 #footer {
178   margin: 40px 0 0 0;
179   padding: 10px 0 0 0;
180   border-top: 1px solid #999999;
181   font-size: 12px;
182   color: #999999;
183   text-align: center;
184 }
```

Código CSS 3.25: blog.css

- Linhas 2 a 3: Definimos para todos os elementos, através das propriedades `margin` e `padding`, as margens externas e internas, respectivamente.
- Linha 13: Definimos, através da propriedade `margin`, a margem externa do conteúdo do blog. As margens da esquerda e direita serão calculadas automaticamente.
- Linhas 16 a 19: Definimos, através das propriedades `border-bottom` e `margin`, a borda inferior do cabeçalho, assim como suas margens externas.
- Linha 26: Definimos, através da propriedade `margin`, as margens externas da imagem do logo.
- Linha 32: Definimos, através da propriedade `display`, a forma de exibição do menu principal.
- Linhas 35 a 38: Definimos, através das propriedades `display` e `margin`, a forma de exibição e as margens externas dos itens menu principal.
- Linhas 41 a 42: Definimos, através das propriedades `display` e `padding`, a forma de exibição e as margens internas dos links do menu principal.
- Linhas 47 a 48: Definimos, através das propriedades `border-top-left` e `border-top-right`, bordas arredondadas na parte superior esquerda e direita, respectivamente dos links do menu principal (propriedades do CSS3).
- Linha 57: Definimos, através da propriedade `display`, a forma de exibição do conteúdo principal.
- Linha 65: Definimos, através da propriedade `padding`, a margem interna dos cabeçalhos nível 1 do conteúdo principal.
- Linha 72: Definimos, através da propriedade `padding`, a margem interna do conteúdo do blog.
- Linha 76: Definimos, através da propriedade `margin`, a margem externa dos parágrafos filhos do conteúdo do blog.
- Linha 80: Definimos, através da propriedade `border-collapse`, a maneira como as bordas adjacentes das tabelas filhas do conteúdo do blog devem se comportar.
- Linhas 81 a 82: Definimos, através das propriedades `border` e `margin`, a borda e margem externa das tabelas filhas do conteúdo do blog.

- Linhas 86 a 90: Definimos, através das propriedades `border` e `padding`, a borda e margem interna dos títulos das colunas e células das tabelas filhas do conteúdo do blog.
- Linha 108: Definimos, através da propriedade `margin`, as margens externas das listas do conteúdo do blog.
- Linhas 112 a 113: Definimos, através das propriedades `display` e `margin`, a forma de exibição e as margens externas do conteúdo lateral.
- Linhas 117 a 120: Definimos, através das propriedades `display` e `margin`, a forma de exibição e as margens externas da publicidade.
- Linhas 130 a 132: Definimos, através das propriedades `border-bottom`, `margin` e `padding`, a borda inferior e as margens externa e interna dos cabeçalhos nível 3 do formulário de contato.
- Linha 136: Definimos, através da propriedade `margin`, a margem externa dos elementos `DIV` do formulário de contato.
- Linhas 148 a 151: Definimos, através das propriedades `display`, `border`, `margin` e `padding`, a forma de exibição, as bordas e as margens externas e internas dos elementos input e textarea do formulário de contato.
- Linha 161: Definimos, através da propriedade `display`, a forma de exibição dos checkboxes do formulário de contato.
- Linhas 166 a 169: Definimos, através das propriedades `display`, `border`, `margin` e `padding`, a forma de exibição, as bordas e as margens externas e internas do botão de envio do formulário de contato.
- Linhas 178 a 180: Definimos, através das propriedades `border-top`, `margin` e `padding`, a borda superior e as margens externa e interna do rodapé.

Posicionando elementos

Para posicionar um elemento dentro de um documento HTML o CSS possui os seguintes atributos:

- position - define o tipo de posicionamento.
- top - define a distância do topo do elemento em relação a outro elemento ou em relação a janela.
- left - define a distância do lado esquerdo do elemento em relação a outro elemento ou em relação a janela.
- bottom - define a distância da base do elemento em relação a outro elemento ou em relação a janela.
- right - define a distância do lado direito do elemento em relação a outro elemento ou em relação a janela.

Ao posicionarmos um elemento utilizando os atributos acima devemos nos lembrar que o sistema de coordenadas dentro de um documento HTML possui a coordenada (0,0) no canto superior esquerdo de um elemento ou da janela. Também devemos nos lembrar que se definirmos uma distância para o atributo `left`, não devemos utilizar o atributo `right`. A mesma ideia vale para os atributos `top` e `bottom`.

Posicionamento estático

Este tipo de posicionamento, em geral, não precisa ser definido, pois é o tipo de posicionamento padrão de todos os elementos. O posicionamento estático é definido através do atributo `position` com o valor `static` e não é afetado pelos atributos `top`, `bottom`, `left` e `right`.

Posicionamento fixo

Um elemento com posicionamento fixo é posicionado em relação à janela do navegador. É definido através do atributo `position` com o valor `fixed` e sua posição é definida pelos atributos `top`, `bottom`, `left` e/ou `right`.

Todos os elementos posicionados fixamente não mudam de posição mesmo quando ocorrer uma rolagem vertical ou horizontal.

Posicionamento relativo

Um elemento com posicionamento relativo é posicionado em relação à sua posição original. É definido através do atributo `position` com o valor `relative` e sua posição é definida pelos atributos `top`, `bottom`, `left` e/ou `right`.

Posicionamento absoluto

Um elemento com posicionamento absoluto é posicionado em relação à um elemento ancestral ou à janela do navegador. É definido através do atributo `position` com o valor `absolute` e sua posição é definida pelos atributos `top`, `bottom`, `left` e/ou `right`.

Quando nenhum dos ancestrais de um elemento posicionado absolutamente define um tipo de posicionamento, o posicionamento absoluto ocorre em relação à janela do navegador. Para que ele ocorra em relação a um ancestral, o elemento ancestral deve definir um posicionamento relativo, por exemplo.

Exercícios de Fixação

10 Continuando o exercício anterior, altere o arquivo **blog.css** conforme as linhas destacadas abaixo:

```
1  * {
2    margin: 0;
3    padding: 0;
4    font-family: arial, sans-serif;
5  }
6
7  body {
8    white-space: nowrap;
9  }
10
11 #blog {
12   width: 980px;
13   margin: 0 auto;
14 }
15
16 #header {
17   position: relative;
18   border-bottom: 1px solid #074f83;
```

```
19    margin: 0 0 20px 0;
20  }
21
22  #logo-link {
23    text-decoration: none;
24  }
25
26  #logo-link img {
27    margin: 20px 40px 20px 20px;
28    vertical-align: bottom;
29  }
30
31  #main-menu {
32    list-style: none;
33    display: inline-block;
34  }
35
36  #main-menu li {
37    display: inline-block;
38    margin: 0 10px 0 0;
39  }
40
41  #main-menu li a {
42    display: block;
43    padding: 8px 10px 5px 10px;
44    background: #cecece;
45    color: #333333;
46    text-decoration: none;
47
48    border-top-left-radius: 5px; /* CSS3 */
49    border-top-right-radius: 5px; /* CSS3 */
50  }
51
52  #main-menu li a.selected {
53    background: #074f83;
54    color: #ffffff;
55  }
56
57  #main-content {
58    display: inline-block;
59    vertical-align: top;
60    width: 660px;
61    color: #666666;
62    white-space: normal;
63  }
64
65  #main-content h1 {
66    padding: 10px;
67    background: #074f83;
68    color: #ffffff;
69    font-size: 24px;
70  }
71
72  #blog-content {
73    padding: 20px;
74  }
75
76  #blog-content > p {
77    margin: 0 0 20px 0;
78  }
79
80  #blog-content > table {
81    border-collapse: collapse;
82    border: 1px solid #5f5f5f;
83    margin: 0 0 20px 0;
84    width: 100%;
85  }
86
87  #blog-content > table th,
88  #blog-content > table td {
```

```
  border: 1px solid #5f5f5f;
  padding: 10px 5px;
}

#blog-content > table th {
  color: #ffffff;
  background: #5f5f5f;
}

#blog-content > table td {
  background: #dddddd;
  color: #333333;
}

#blog-content > table td.first-col {
  background: #a9a9a9;
}

#blog-content ul {
  list-style-image: url('list-bullet.png');
  margin: 0 0 0 30px;
}

#side-content {
  display: inline-block;
  margin: 0 0 0 20px;
  vertical-align: top;
}

#ad {
  display: block;
  margin: 0 0 20px 0;
}

#contact-form {
  text-align: right;
}

#contact-form h3 {
  font-size: 18px;
  color: #074f83;
  text-align: left;
  border-bottom: 1px solid #999999;
  margin: 0 0 10px 0;
  padding: 0 0 5px 0;
}

#contact-form div {
  margin: 0 0 10px 0;
  text-align: left;
}

#contact-form label {
  font-size: 14px;
  vertical-align: middle;
  color: #666666;
}

#contact-form input,
#contact-form textarea {
  display: block;
  border: 1px solid #999999;
  padding: 5px;
  margin: 4px 0 0 0;
  width: 288px;
  font-size: 18px;
}

#contact-form textarea {
  height: 100px;
```

```
159 }
160
161 #contact-form input[type='checkbox'] {
162   display: inline;
163   width: auto;
164 }
165
166 #contact-form input[type='submit'] {
167   margin: 10px 0 0 0;
168   border: none;
169   padding: 10px 20px;
170   display: inline;
171   background: #074f83;
172   color: #ffffff;
173   text-transform: uppercase;
174   font-size: 14px;
175   width: auto;
176 }
177
178 #footer {
179   margin: 40px 0 0 0;
180   padding: 10px 0 0 0;
181   border-top: 1px solid #999999;
182   font-size: 12px;
183   color: #999999;
184   text-align: center;
185 }
186
187 #login-area {
188   position: absolute;
189   top: 0px;
190   right: 0px;
191   padding: 4px 0;
192 }
193
194 #login-area a {
195   color: #074F83;
196 }
```

Código CSS 3.26: blog.css

- Linha 17: Definimos, através da propriedade `position`, o tipo de posicionamento do cabeçalho.

- Linhas 187 a 192: Definimos, através das propriedades `position`, `top`, `right` e `padding`, o tipo de posicionamento, a distância em relação ao topo, a distância em relação ao lado direito e a margem interna da área de login.

- Linhas 194 a 196: Definimos, através da propriedade `color`, a cor do link da área de login.

Desafios

1 Observe a página inicial da K19. Utilizando todos os seus conhecimentos em CSS, faça uma página que siga a mesma estrutura da página inicial da K19. Não se preocupe com as cores e imagens, no lugar das imagens da K19 você pode usar uma imagem qualquer e escolher livremente as cores que desejar.

Figura 3.8: Box model

JAVASCRIPT

CAPÍTULO 4

Para que possamos criar uma página que possua um comportamento e oferecer aos nossos usuários um site mais interativo e dinâmico, com certeza trabalharemos com a linguagem JavaScript.

Um código JavaScript pode ser inserido em um documento HTML de duas formas: colocando o código JavaScript como filho de um elemento com a tag `script` ou utilizando o atributo `src` de um elemento com a tag `script` no qual devemos passar o caminho relativo ou absoluto para um arquivo que contenha o código JavaScript.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Inserindo código JavaScript em um documento HTML</title>
    <script type="text/javascript" src="codigo.js"></script>
    <script type="text/javascript">
      window.onload = function(){
        document.getElementById('ola-mundo').innerHTML = 'Olá Mundo!';
      }
    </script>
  </head>
  <body>
  <p id="ola-mundo"></p>
  </body>
</html>
```

Código HTML 4.1: Inserindo código JavaScript em um documento HTML

Declarando e inicializando variáveis em JavaScript

Em JavaScript podemos declarar e inicializar uma variável da seguinte maneira:

```
var numero = 0;
var numeroComCasasDecimais = 1.405;
var texto = 'Variável com um texto';
var outroTexto = "Outra variável com um texto";
var valorBooleano = true;
```

Código Javascript 4.1: Declarando e inicializando variáveis em JavaScript

Operadores

Para manipular os valores das variáveis de um programa, devemos utilizar os operadores oferecidos pela linguagem de programação adotada. A linguagem JavaScript possui diversos operadores e os principais são categorizados da seguinte forma:

- Aritmético (`+`, `-`, `*`, `/`, `%`)

- Atribuição (=, +=, -=, *=, /=, %=)
- Relacional (==, !=, <, <=, >, >=)
- Lógico (&&, ||)

Aritmético

Os operadores aritméticos funcionam de forma muito semelhante aos operadores na matemática. Os operadores aritméticos são:

- Soma +
- Subtração -
- Multiplicação *
- Divisão /
- Módulo %

```
1 var umMaisUm = 1 + 1;            // umMaisUm = 2
2 var tresVezesDois = 3 * 2;       // tresVezesDois = 6
3 var quatroDivididoPor2 = 4 / 2; // quatroDivididoPor2 = 2
4 var seisModuloCinco = 6 % 5;     // seisModuloCinco = 1
5 var x = 7;
6 x = x + 1 * 2;                   // x = 9
7 x = x - 4;                       // x = 10
8 x = x / (6 - 2 + (3*5)/(16-1)); // x = 2
```

Código Javascript 4.2: Exemplo de uso dos operadores aritméticos.

Importante

O módulo de um número x, na matemática, é o valor numérico de x desconsiderando o seu sinal (valor absoluto). Na matemática expressamos o módulo da seguinte forma: $|-2| = 2$.

Em linguagens de programação, o módulo de um número é o resto da divisão desse número por outro. No exemplo acima, o resto da divisão de 6 por 5 é igual a 1. Além disso, lemos a expressão 6%5 da seguinte forma: seis módulo cinco.

Importante

As operações aritméticas em JavaScript obedecem as mesmas regras da matemática com relação à precedência dos operadores e parênteses. Portanto, as operações são resolvidas a partir dos parênteses mais internos até os mais externos, primeiro resolvemos as multiplicações, divisões e os módulos. Em seguida, resolvemos as adições e subtrações.

Atribuição

Nas seções anteriores, já vimos um dos operadores de atribuição, o operador = (igual). Os operadores de atribuição são:

- Simples =
- Incremental +=

- Decremental -=
- Multiplicativa *=
- Divisória /=
- Modular %=

```
1 var valor = 1;    // valor = 1
2 valor += 2;       // valor = 3
3 valor -= 1;       // valor = 2
4 valor *= 6;       // valor = 12
5 valor /= 3;       // valor = 4
6 valor %= 3;       // valor = 1
```

Código Javascript 4.3: Exemplo de uso dos operadores de atribuição.

As instruções acima poderiam ser escritas de outra forma:

```
1 var valor = 1;           // valor = 1
2 valor = valor + 2;       // valor = 3
3 valor = valor - 1;       // valor = 2
4 valor = valor * 6;       // valor = 12
5 valor = valor / 3;       // valor = 4
6 valor = valor % 3;       // valor = 1
```

Código Javascript 4.4: O mesmo exemplo anterior, usando os operadores aritméticos.

Como podemos observar, os operadores de atribuição, com exceção do simples (=), reduzem a quantidade de código escrito. Podemos dizer que esses operadores funcionam como "atalhos" para as operações que utilizam os operadores aritméticos.

Relacional

Muitas vezes precisamos determinar a relação entre uma variável ou valor e outra outra variável ou valor. Nessas situações, utilizamos os operadores relacionais. As operações realizadas com os operadores relacionais devolvem valores booleanos. Os operadores relacionais são:

- Igualdade ==
- Diferença !=
- Menor <
- Menor ou igual <=
- Maior >
- Maior ou igual >=

```
1 var valor = 2;
2 var t = false;
3 t = (valor == 2);    // t = true
4 t = (valor != 2);    // t = false
5 t = (valor < 2);     // t = false
6 t = (valor <= 2);    // t = true
7 t = (valor > 1);     // t = true
8 t = (valor >= 1);    // t = true
```

Código Javascript 4.5: Exemplo de uso dos operadores relacionais em JavaScript.

Lógico

A linguagem JavaScript permite verificar duas ou mais condições através de operadores lógicos. Os operadores lógicos devolvem valores booleanos. Os operadores lógicos são:

- "E" lógico &&
- "OU" lógico ||

```
1 var valor = 30;
2 var teste = false;
3 teste = valor < 40 && valor > 20;      // teste = true
4 teste = valor < 40 && valor > 30;      // teste = false
5 teste = valor > 30 || valor > 20;      // teste = true
6 teste = valor > 30 || valor < 20;      // teste = false
7 teste = valor < 50 && valor == 30;     // teste = true
```

Código Javascript 4.6: Exemplo de uso dos operadores lógicos em JavaScript.

Controle de fluxo

`if` e `else`

O comportamento de uma aplicação pode ser influenciado por valores definidos pelos usuários. Por exemplo, considere um sistema de cadastro de produtos. Se um usuário tenta adicionar um produto com preço negativo, a aplicação não deve cadastrar esse produto. Caso contrário, se o preço não for negativo, o cadastro pode ser realizado normalmente.

Outro exemplo, quando o pagamento de um boleto é realizado em uma agência bancária, o sistema do banco deve verificar a data de vencimento do boleto para aplicar ou não uma multa por atraso.

Para verificar uma determinada condição e decidir qual bloco de instruções deve ser executado, devemos aplicar o comando **if**.

```
1 if (preco < 0) {
2   alert('O preço do produto não pode ser negativo');
3 } else {
4   alert('Produto cadastrado com sucesso');
5 }
```

Código Javascript 4.7: Comando `if`

O comando `if` permite que valores booleanos sejam testados. Se o valor passado como parâmetro para o comando `if` for `true`, o bloco do `if` é executado. Caso contrário, o bloco do **else** é executado.

O parâmetro passado para o comando `if` deve ser um valor booleano, caso contrário o código não compila. O comando `else` e o seu bloco são opcionais.

`while`

Em alguns casos, é necessário repetir um trecho de código diversas vezes. Suponha que seja necessário imprimir 10 vezes na página a mensagem: "Bom Dia". Isso poderia ser realizado colocando

10 linhas iguais a essa no código fonte:

```
1 document.writeln('Bom Dia');
```

Código Javascript 4.8: "Bom Dia"

Se ao invés de 10 vezes fosse necessário imprimir 100 vezes, já seriam 100 linhas iguais no código fonte. É muito trabalhoso utilizar essa abordagem para solucionar esse problema.

Através do comando **while**, é possível definir quantas vezes um determinado trecho de código deve ser executado pelo computador.

```
1 var contador = 0;
2
3 while(contador < 100) {
4   document.writeln('Bom Dia');
5   contador++;
6 }
```

Código Javascript 4.9: Comando `while`

A variável `contador` indica o número de vezes que a mensagem "Bom Dia" foi impressa na tela. O operador ++ incrementa a variável `contador` a cada rodada.

O parâmetro do comando `while` tem que ser um valor booleano. Caso contrário, ocorrerá um erro na execução do script.

`for`

O comando **for** é análogo ao `while`. A diferença entre esses dois comandos é que o `for` recebe três argumentos.

```
1 for(var contador = 0; contador < 100; contador++) {
2   document.writeln('Bom Dia');
3 }
```

Código Javascript 4.10: Comando `for`

Exercícios de Fixação

1 Crie uma pasta chamada **javascript** na sua pasta de exercícios. Crie uma página HTML vinculada a um arquivo JavaScript, que imprima na página o seu nome 100 vezes. Salve os documentos na pasta **javascript** e em seguida abra o arquivo HTML no navegador.

```
1 <html>
2   <head>
3     <title>Imprime nome</title>
4     <script type="text/javascript" src="imprime-nome.js"></script>
5   </head>
6   <body>
7   </body>
8 </html>
```

Código HTML 4.2: imprime-nome.html

```
for(var contador = 0; contador < 100; contador++) {
  document.writeln('Rafael Cosentino');
    document.writeln('<br/>');
}
```

Código Javascript 4.11: imprime-nome.js

2 Crie uma página HTML vinculada a um arquivo JavaScript que imprima na página os números de 1 até 100. Salve esses arquivos na pasta **javascript** e em seguida abra o arquivo HTML no navegador.

```
<html>
  <head>
    <title>Imprime ate 100</title>
    <script type="text/javascript" src="imprime-ate-100.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.3: imprime-ate-100.html

```
for(var contador = 1; contador <= 100; contador++){
  document.writeln(contador);
  document.writeln('<br />');
}
```

Código Javascript 4.12: imprime-ate-100.js

3 Crie um documento HTML vinculado a um documento JavaScript que percorra todos os números de 1 até 100. Para os números ímpares, deve ser impresso um "*", e para os números pares, deve ser impresso dois "**". Veja o exemplo abaixo:

*
**
*
**
*
**

Salve os documentos na pasta javascript e em seguida abra o arquivo html no navegador.

```
<html>
  <head>
    <title>Imprime padrão 1</title>
    <script type="text/javascript" src="imprime-padrao-1.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.4: imprime-padrao-1.html

```
for(var contador = 1; contador <= 100; contador++) {
  var resto = contador % 2;
```

```
3   if(resto == 1) {
4     document.writeln('*');
5   } else {
6     document.writeln('**')
7   }
8
9   document.writeln('<br />')
10 }
```

Código Javascript 4.13: imprime-padrao-1.js

4 Crie um documento HTML vinculado a um documento JavaScript que percorra todos os número de 1 até 100. Para os números múltiplos de 4, imprima a palavra "PI", e para os outros, imprima o próprio número. Veja o exemplo abaixo:

1
2
3
PI
5
6
7
PI

Salve os documentos na pasta javascript e em seguida abra o arquivo HTML no navegador.

```
1  <html>
2    <head>
3      <title>Imprime padrão 2</title>
4      <script type="text/javascript" src="imprime-padrao-2.js"></script>
5    <head>
6    <body>
7    </body>
8  <html>
```

Código HTML 4.5: imprime-padrao-2.html

```
1  for(var contador = 1; contador <= 100; contador++) {
2    var resto = contador % 4;
3    if(resto == 0) {
4      document.writeln('PI');
5    } else {
6      document.writeln(contador);
7    }
8
9    document.writeln('<br />');
10 }
```

Código Javascript 4.14: imprime-padrao-2.js

Salve os documentos na pasta **javascript** e em seguida abra o arquivo HTML no navegador.

Exercícios Complementares

1 Crie um documento HTML vinculado a um documento JavaScript que imprima os números de 1 até 50 duas vezes.

2 Crie um documento HTML vinculado a um documento JavaScript que imprima na página o nome da formação básica da K19, 5 vezes. E entre cada impressão o nome deste curso 3 vezes.

3 Crie um documento HTML vinculado a um documento JavaScript que percorra todos os números de 1 até 60. Para os números multiplo de, 3 imprima "**"; Para o restante imprima "*".

4 Crie um documento HTML vinculado a um documento JavaScript que imprima todos os números de 1 até 80 e imprimir “*” no lugar de todos os números múltiplos de 4 e 7.

5 Crie um documento HTML vinculado a um documento JavaScript que imprima na página um triângulo de “*”. Veja o exemplo abaixo:

```
*
**
***
****
*****
```

6 Crie um documento HTML vinculado a um documento JavaScript que imprima na tela vários triângulos de “*”. Observe o padrão abaixo.

```
*
**
***
****
*
**
***
****
```

7 Os números de Fibonacci são uma sequência de números definida recursivamente. O primeiro elemento da sequência é 0 e o segundo é 1. Os outros elementos são calculados somando os dois antecessores.

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233...
```

Crie um documento HTML vinculado a um documento JavaScript para imprimir os 30 primeiros números da sequência de Fibonacci.

Funções JavaScript

Uma função JavaScript é uma sequência de instruções JavaScript que serão executadas quando você chamá-la através do seu nome.

Definindo uma função

Definindo uma função simples em JavaScript:

```
1 function imprimeOlaMundo() {
2   document.writeln('Olá Mundo!');
3   document.writeln('<br />');
4 }
```

Código Javascript 4.22: Definindo uma função

Definindo uma função que recebe parâmetros:

```
1 function imprimeMensagem(mensagem) {
2   document.writeln(mensagem);
3   document.writeln('<br />');
4 }
```

Código Javascript 4.23: Definindo uma função que recebe parâmetros

Definindo uma função que retorna um valor:

```
1 function criaSaudacaoPersonalizada(nome) {
2   return 'Olá, ' + nome + '!';
3 }
```

Código Javascript 4.24: Definindo uma função que retorna um valor

Chamando uma função dentro do código JavaScript

Chamando uma função dentro do código JavaScript:

```
1 imprimeOlaMundo();
```

Código Javascript 4.25: Chamando uma função

Chamando uma função que recebe parâmetros:

```
1 imprimeMensagem('123 testando!!');
```

Código Javascript 4.26: Chamando uma função que recebe parâmetros

Chamando uma função que retorna um valor:

```
1 var saudacao = criaSaudacaoPersonalizada('Jonas');
```

Código Javascript 4.27: Chamando uma função que retorna um valor

Chamando uma função JavaScript pelo HTML

Os elementos HTML possuem alguns eventos que podem ser associados a funções JavaScript através de alguns atributos especiais cujos nomes começam com o prefixo on.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Chamando uma função JavaScript pelo HTML</title>
    <script type="text/javascript">
      function exibeOlaMundo() {
        alert('Olá Mundo!');
      }
    </script>
  </head>
  <body>
    <input type="button" value="Exibe saudação" onclick="exibeOlaMundo()" />
  </body>
</html>
```

Código HTML 4.13: Chamando uma função JavaScript pelo HTML

Objetos JavaScript

Qualquer desenvolvedor acostumado com linguagens orientadas a objetos como Java e C# pode estranhar um pouco a forma como trabalhamos com objetos em JavaScript. Apesar do JavaScript ser uma linguagem de script baseada em protótipos, ela oferece suporte à programação orientada a objetos. Portanto, muitos dos conhecimentos que um desenvolvedor tenha adquirido com Java ou C# com relação a orientação a objetos pode ser reaproveitado ao se programar em JavaScript.

Criando um objeto

Existe mais de uma maneira de se criar um objeto em JavaScript. A maneira mais simples podemos acompanhar no código abaixo:

```
var contaBancaria = new Object();

contaBancaria.numero = 1234;
contaBancaria.saldo = 1000;

contaBancaria.deposita = function(valor) {
  if(valor > 0) {
    this.saldo += valor;
  }
  else {
    alert('Valor inválido!');
  }
};
```

Código Javascript 4.28: Criando um objeto

Outra maneira de se criar um objeto é utilizando a notação literal mais conhecida como JSON (JavaScript Object Notation):

```
var contaBancaria = {
  numero: 1234,
  saldo: 1000,
  deposita: function(valor) {
    if(valor > 0) {
```

```
6         this.saldo += valor;
7       }
8       else {
9         alert('Valor inválido!');
10      }
11    }
12  }
```

Código Javascript 4.29: Criando um objeto utilizando a notação literal

Arrays

Os arrays em JavaScript são objetos e, portanto, possuem atributos e métodos para nos ajudar na manipulação de seus dados.

Declarando um array

Podemos declarar um array de três maneiras: através do protótipo `Array` sem parâmetros, através do protótipo `Array` com parâmetros e através da forma literal.

```
1  var numeros = new Array();
2  numeros[0] = 34;
3  numeros[1] = 52;
4  numeros[2] = 83;
5
6  var nomes = new Array('Jonas', 'Rafael', 'Marcelo');
7
8  var cidades = ['São Paulo', 'Rio de Janeiro', 'Curitiba'];
```

Código Javascript 4.30: Criando um array

Métodos do array

Um array possui diversos métodos para nos auxiliar nas tarefas mais comuns quando trabalhamos com arrays. Os mé todos são:

- concat() - concatena dois ou mais arrays e retorna uma cópia ro** resultado.
- indexOf() - procura por um objeto dentro do array e retorna o índice caso o encontre.
- join() - concatena todos os elementos de um array em uma string.
- lastIndexOf() - procura, de trás para frente, por um objeto dentro array e retorna o índice caso o encontre.
- pop() - remove o último objeto de um array e retorna o objeto removido.
- push() - adiciona um objeto no final do array e retorno o novo tamanho do array.
- reverse() - inverte a ordem dos objetos de um array.
- shift() - remove o primeiro objeto de um array e retorna o objeto removido.
- slice() - seleciona parte de um array e retorna um copia da parte selecionada.
- sort() - ordena os objetos de um array.
- splice() - adiciona e/ou remove objetos de um array.
- toString() - converte um array em uma string e retorna o resultado.

- unshift() - adiciona um objeto no começo do array e retorna o novo tamanho do array.
- valueOf() - retorna o valor primitivo de um array.

Exercícios de Fixação

5 Crie um documento HTML vinculado a um documento JavaScript que armazene 10 números inteiros em um array. Preencha todas as posições do array com valores sequenciais e em seguida imprima-os na tela. Em seguida, escolha duas posições aleatoriamente e troque os valores de uma posição pelo da outra. Repita essa operação 10 vezes. Ao final, imprima o array novamente.

```
<html>
  <head>
    <title> Arrays </title>
    <script type="text/javascript" src="arrays.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.14: arrays.html

```
var array = new Array(10);

for(var i = 0; i < array.length; i++){
  array[i] = i;
}

for(var i = 0; i < array.length; i++){
  document.writeln(array[i]);
  document.writeln('<br />');
}

for(var i = 0; i < 10; i++){
  var posicao1 = Math.floor(Math.random()*10);
  var posicao2 = Math.floor(Math.random()*10);
  var auxiliar = array[posicao1];

  array[posicao1] = array[posicao2];
  array[posicao2] = auxiliar;
}

document.writeln("--------------------");
document.writeln('<br />');

for(var i = 0; i < array.length; i++){
  document.writeln(array[i]);
  document.writeln('<br />');
}
```

Código Javascript 4.31: arrays.js

6 Crie um documento HTML vinculado a um documento JavaScript que armazene 10 números inteiros em um array. Preencha todas as posições do array com valores aleatórios e em seguida imprima-os na tela. Após imprimir o array, ordene o array do menor valor para o maior. Ao final, imprima o array ordenado.

```
<html>
  <head>
    <title>Ordena do número menor para o maior</title>
    <script type="text/javascript" src="ordena.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.15: ordena.html

```
var array = new Array(10);

for(var i = 0; i < array.length; i++){
  array[i] = Math.floor(Math.random()*10);
}

for(var i = 0; i < array.length; i++){
  document.writeln(array[i]);
}

array.sort();

document.writeln("--------------------");

for(var i = 0; i < array.length; i++){
  document.writeln(array[i]);
}
```

Código Javascript 4.32: ordena.js

Exercícios Complementares

8 Crie um documento HTML vinculado a um documento JavaScript que crie um array com 10 números sequenciais do tipo `int`. Feito isso, escolha duas posições deste array e aleatoriamente troque os valores dessas duas posições. Repita essa troca 15 vezes. No final, imprima esse array.

9 Crie um documento HTML vinculado a um documento JavaScript que preencha um array do tipo `int` com 15 números aleatórios. Imprima esse array. Após isso, ordene o array do menor valor para o maior valor. Em seguida, imprima esse array novamente com os números ordenados.

JAVASCRIPT AVANÇADO

APÊNDICE A

Objetos

Um objeto é um conjunto de propriedades. Toda propriedade possui nome e valor. O nome de uma propriedade pode ser qualquer sequência de caracteres. O valor de uma propriedade pode ser qualquer valor exceto `undefined`. Podemos adicionar uma nova propriedade a um objeto que já existe. Um objeto pode herdar propriedades de outro objeto utilizando a ideia de *prototype*.

Criando Objetos

Um objeto pode ser criado de forma literal. Veja o exemplo a seguir.

```
1 var objeto_vazio = {};
2 var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
```

Código Javascript A.1: Criando um objeto de forma literal

Um objeto pode se relacionar com outros objetos através de propriedades. Observe o código abaixo.

```
1 var formacao_java = {sigla: "K10", nome: "Formação Desenvolvedor Java",
2 cursos: [
3   {sigla: "K11", nome: "Orientação a Objetos em Java"},
4   {sigla: "K12", nome: "Desenvolvimento Web com JSF2 e JPA2"},
5     ]
6 };
```

Código Javascript A.2: Criando um objeto associado a outro

Recuperando o Valor de uma Propriedade

Para recuperar os valores das propriedades de um objeto, podemos utilizar o operador "." ou "[]". Veja o exemplo a seguir.

```
1 var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2 console.log(curso.sigla);
3 console.log(curso["sigla"]);
4
5 var sigla = "sigla";
6 console.log(curso[sigla]);
```

Código Javascript A.3: Recuperando o valor de uma propriedade

Alterando o Valor de uma Propriedade

Para alterar o valor de uma propriedade, basta atribuir um novo valor a propriedade do objeto.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  curso.sigla = "K12";
4  curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
5
6  console.log(curso.sigla);
7  console.log(curso.nome);
```

Código Javascript A.4: Alterando o valor de uma propriedade

Referências

Os objetos são acessados através de referências

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  // copiando uma referência
4  var x = curso;
5
6  x.sigla = "K12";
7  x.nome = "Desenvolvimento Web com JSF2 e JPA2";
8
9  // imprime K12
10 console.log(curso.sigla);
11
12 // imprime Desenvolvimento Web com JSF2 e JPA2
13 console.log(curso.nome);
```

Código Javascript A.5: Referência

Protótipos

Podemos criar um objeto baseado em outro objeto existente (protótipo). Para isso, podemos utilizar a propriedade especial `__proto__`. Observe o código abaixo.

```
1  // criando um objeto com duas propriedades
2  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
3
4  // criando um objeto sem propriedades
5  var novo_curso = {};
6
7  // definindo o primeiro objeto como protótipo do segundo
8  novo_curso.__proto__ = curso;
9
10 // imprime K11
11 console.log(novo_curso.sigla);
12
13 // imprime Orientação a Objetos em Java
14 console.log(novo_curso.nome);
```

Código Javascript A.6: Criando objeto com __proto__

Também podemos utilizar o método `create` de `Object` para criar objetos baseados em objetos existentes. Veja o exemplo abaixo.

```
1  // criando um objeto com duas propriedades
2  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
3
4  // criando um objeto sem propriedades
5  var novo_curso = {};
6
7  // definindo o primeiro objeto como protótipo do segundo
```

```
8  novo_curso = Object.create(curso);
9
10 // imprime K11
11 console.log(novo_curso.sigla);
12
13 // imprime Orientação a Objetos em Java
14 console.log(novo_curso.nome);
```

Código Javascript A.7: Criando objetos com Object.create()

Se uma propriedade for adicionada a um objeto, ela também será adicionada a todos os objetos que o utilizam como protótipo.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  curso.carga_horaria = 36;
6
7  // imprime K11
8  console.log(novo_curso.sigla);
9
10 // imprime Orientação a Objetos em Java
11 console.log(novo_curso.nome);
12
13 // imprime 36
14 console.log(novo_curso.carga_horaria);
```

Código Javascript A.8: Adicionando uma propriedade em um objeto que é utilizado como protótipo

Por outro lado, se uma propriedade for adicionada a um objeto, ela não será adicionada no protótipo desse objeto.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  novo_curso.carga_horaria = 36;
6
7  // imprime K11
8  console.log(curso.sigla);
9
10 // imprime Orientação a Objetos em Java
11 console.log(curso.nome);
12
13 // imprime undefined
14 console.log(curso.carga_horaria);
```

Código Javascript A.9: Adicionando uma propriedade em um objeto

Se o valor de uma propriedade de um objeto for modificado, os objetos que o utilizam como protótipo podem ser afetados.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  curso.sigla = "K12";
6  curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
7
8  // imprime K12
9  console.log(novo_curso.sigla);
10
11 // imprime Desenvolvimento Web com JSF2 e JPA2
```

```
12 console.log(novo_curso.nome);
```

Código Javascript A.10: Modificando o valor de uma propriedade de um objeto que é utilizado como protótipo

Por outro lado, alterações nos valores das propriedades de um objeto não afetam o protótipo desse objeto.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  novo_curso.sigla = "K12";
6  novo_curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
7
8  // imprime K11
9  console.log(curso.sigla);
10
11 // imprime Orientação a Objetos em Java
12 console.log(curso.nome);
```

Código Javascript A.11: Modificando o valor de uma propriedade de um objeto

Considere um objeto que foi construído a partir de um protótipo. Se o valor de uma propriedade herdada do protótipo for alterada nesse objeto, ela se torna independente da propriedade no protótipo. Dessa forma, alterações no valor dessa propriedade no protótipo não afetam mais o valor dela no objeto gerado a partir do protótipo.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  novo_curso.sigla = "K12";
6  novo_curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
7
8  curso.sigla = "K21";
9  curso.nome = "Persistência com JPA2 e Hibernate";
10
11 // imprime K12
12 console.log(novo_curso.sigla);
13
14 // imprime Desenvolvimento Web com JSF2 e JPA2
15 console.log(novo_curso.nome);
```

Código Javascript A.12: Sobrescrevendo uma propriedade

Removendo uma Propriedade

Podemos remover uma propriedade de um objeto com a função `delete`.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  // imprime K11
4  console.log(curso.sigla);
5
6  delete curso.sigla;
7
8  // imprime undefined
9  console.log(curso.sigla);
```

Código Javascript A.13: Removendo uma propriedade

Verificando a Existência de uma Propriedade

Podemos verificar se uma propriedade existe, podemos utilizar a função `in`.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  // imprime true
4  console.log("sigla" in curso);
5
6  // imprime false
7  console.log("carga_horaria" in curso);
```

Código Javascript A.14: Verificando a existência de uma propriedade

Exercícios de Fixação

1 Para fazer o exercício vamos utilizar o add-on Firebug do Firefox.

Para executar o código Javascript, devemos habilitar o `console` através do link "enable".

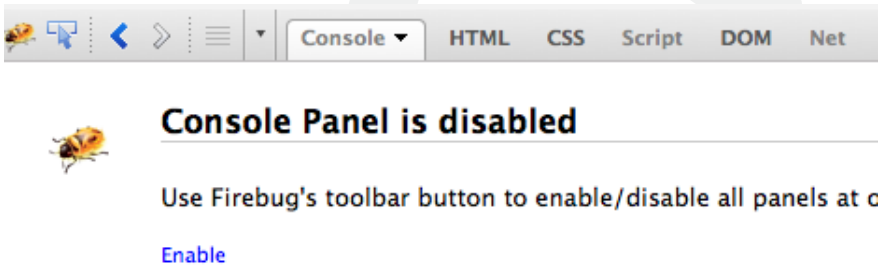

Figura A.1: Habilitando o console

Após o console ter sido habilitado, podemos executar o código Javascript conforme a figura abaixo.

Figura A.2: Executando código JavaScript

2 Crie objetos com propriedades chamadas `sigla` e `nome`. Imprima o valor dessas propriedades através do `console.log` do Firebug.

```
1 var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2 console.log(curso.sigla);
3 console.log(curso.nome);
4
5 var curso2 = {sigla: "K12", nome: "Desenvolvimento Web com JSF2 e JPA2"};
6 console.log(curso2.sigla);
7 console.log(curso2.nome);
```

Código Javascript A.15: Criando dois objetos

3 Verifique o funcionamento das referências em JavaScript.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  // imprime K11
4  console.log(curso.sigla);
5
6  // imprime Orientação a Objetos em Java
7  console.log(curso.nome);
8
9  var x = curso;
10
11 x.sigla = "K12";
12 x.nome = "Desenvolvimento Web com JSF2 e JPA2";
13
14 // imprime K12
15 console.log(curso.sigla);
16
17 // imprime Desenvolvimento Web com JSF2 e JPA2
18 console.log(curso.nome);
```

Código Javascript A.16: Referências

4 Crie um objeto a partir de outro objeto existente.

```
1 var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3 var novo_curso = Object.create(curso);
4
5 // imprime K11
6 console.log(novo_curso.sigla);
7
8 // imprime Orientação a Objetos em Java
9 console.log(novo_curso.nome);
```

Código Javascript A.17: Protótipo

5 Adicione uma propriedade em um objeto utilizado como protótipo e verifique que essa propriedade será adicionada nos objetos criados a partir desse protótipo.

```
1 var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3 var novo_curso = Object.create(curso);
4
5 curso.carga_horaria = 36;
6
7 // imprime K11
```

```
8  console.log(novo_curso.sigla);
9
10 // imprime Orientação a Objetos em Java
11 console.log(novo_curso.nome);
12
13 // imprime 36
14 console.log(novo_curso.carga_horaria);
```

Código Javascript A.18: Protótipo

6 Adicione uma propriedade em um objeto e verifique que o protótipo desse objeto não é afetado.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  novo_curso.carga_horaria = 36;
6
7  // imprime K11
8  console.log(curso.sigla);
9
10 // imprime Orientação a Objetos em Java
11 console.log(curso.nome);
12
13 // imprime undefined
14 console.log(curso.carga_horaria);
```

Código Javascript A.19: Protótipo

7 Altere o valor de uma propriedade de um objeto utilizado como protótipo e verifique que essa alteração afetará os objetos criados a partir desse protótipo.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
5  // imprime K11
6  console.log(novo_curso.sigla);
7
8  // imprime Orientação a Objetos em Java
9  console.log(novo_curso.nome);
10
11 curso.sigla = "K12";
12 curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
13
14 // imprime K12
15 console.log(novo_curso.sigla);
16
17 // imprime Desenvolvimento Web com JSF2 e JPA2
18 console.log(novo_curso.nome);
```

Código Javascript A.20: Protótipo

8 Reescreva em um objeto as propriedades herdadas de um protótipo e verifique que alterações nos valores dessas propriedades no protótipo não afetam mais os valores delas nesse objeto.

```
1  var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
2
3  var novo_curso = Object.create(curso);
4
```

```
5  novo_curso.sigla = "K12";
6  novo_curso.nome = "Desenvolvimento Web com JSF2 e JPA2";
7
8  // imprime K12
9  console.log(novo_curso.sigla);
10
11 // imprime Desenvolvimento Web com JSF2 e JPA2
12 console.log(novo_curso.nome);
13
14 curso.sigla = "K21";
15 curso.nome = "Persistência com JPA2 e Hibernate";
16
17 // imprime K12
18 console.log(novo_curso.sigla);
19
20 // imprime Desenvolvimento Web com JSF2 e JPA2
21 console.log(novo_curso.nome);
```

Código Javascript A.21: Protótipo

Exercícios Complementares

1. Crie objetos para as formações da K19 com as propriedades `sigla` e `nome` da formação. Através do console `.log` do FireBug, depois imprima o valor.

2. Verifique o funcionamento das referências do exercício anterior em JavaScript.

3. Crie um objeto com algumas propriedades. Dentro desse objeto crie um outro objeto sendo ele um protótipo. Crie também propriedades nesse novo objeto. Imprima todas as propriedades criadas e faça alterações nos valores das propriedades no protótipo para que afetem os valores delas.

Funções

As funções em JavaScript são objetos. Você pode armazená-las em variáveis, arrays e outros objetos. Elas podem ser passadas como argumento ou devolvidas por outra função. Veja o exemplo abaixo.

```
1 var multiplicacao = function(x, y) {
2   return x * y;
3 }
```

Código Javascript A.25: Criando uma função

Utilizando uma Função

Para utilizar a função `multiplicacao`, podemos chamá-la da seguinte forma.

```
1 var resultado = multiplicação(3,2);
```

Código Javascript A.26: Utilizando a função

Método

Quando uma função faz parte de um objeto, ela é chamada de método. Para executar um método, devemos utilizar a referência de um objeto e passar os parâmetros necessários. Observe o código abaixo.

```
var conta = {
  saldo: 0,
  deposita: function(valor) {
    this.saldo += valor;
  }
}

conta.deposita(100);
console.log(conta.saldo);
```

Código Javascript A.27: Método

Apply

Uma função pode ser associada momentaneamente a um objeto e executada através do método `apply`.

```
var deposita = function(valor) {
  this.saldo += valor;
}

var conta = {
  saldo: 0
}

deposita.apply(conta, [200]);
console.log(conta.saldo);
```

Código Javascript A.28: Método apply

Arguments

Os argumentos passados na chamada de uma função podem ser recuperados através do array `Arguments`. Inclusive, esse array permite que os argumentos excedentes sejam acessados.

```
var soma = function() {
  var soma = 0;

  for(var i = 0; i < arguments.length; i ++) {
    soma += arguments[i];
  }

  return soma;
}

var resultado = soma(2,4,5,6,1);

console.log(resultado);
```

Código Javascript A.29: Arguments

Exceptions

Quando um erro é identificado no processamento de uma função, podemos lançar uma exception para avisar que chamou a função que houve um problema.

```
var conta = {
  saldo: 0,
  deposita: function(valor) {
    if(valor <= 0) {
      throw {
        name: "ValorInvalido",
        message: "Valores menores ou iguais a 0 não podem ser depositados"
      }
    } else {
      this.saldo += valor;
    }
  }
}
```

Código Javascript A.30: Exceptions

Na chamada do método `deposita()`, podemos capturar um possível erro com o comando `try-catch`.

```
try {
  conta.deposita(0);
} catch(e) {
  console.log(e.name);
  console.log(e.message);
}
```

Código Javascript A.31: Exceptions

Exercícios de Fixação

9 Crie uma função que multiplicar dois números recebidos como parâmetro e devolve o resultado.

```
var multiplicacao = function(x, y) {
  return x * y;
}
```

Código Javascript A.32: multiplicacao()

10 Execute a função `multiplicacao()`

```
var resultado = multiplicacao(5, 3);
console.log(resultado);
```

Código Javascript A.33: Executando a função multiplicacao()

11 Crie um método para implementar a operação de depósito em contas bancárias.

```
var conta = {
  saldo: 0,
  deposita: function(valor) {
    this.saldo += valor;
  }
}
```

Código Javascript A.34: deposita()

12 Execute o método `deposita()`.

```
conta.deposita(500);
console.log(conta.saldo);
```

Código Javascript A.35: Executando o método deposita()

13 Crie uma função que soma todos os argumentos passados como parâmetro.

```
var soma = function() {
  var soma = 0;

  for(var i = 0; i < arguments.length; i ++) {
    soma += arguments[i];
  }

  return soma;
}
```

Código Javascript A.36: soma()

14 Execute o método `soma()`.

```
var resultado = soma(2,4,5,6,1);

console.log(resultado);
```

Código Javascript A.37: Executando o método soma()

15 Altere a lógica do método `deposita()` para evitar que valores incorretos sejam depositados.

```
var conta = {
  saldo: 0,
  deposita: function(valor) {
    if(valor <= 0) {
      throw {
        name: "ValorInvalido",
        message: "Valores menores ou iguais a 0 não podem ser depositados"
      }
    } else {
      this.saldo += valor;
    }
  }
}
```

Código Javascript A.38: Exceptions

16 Execute o método `deposita()` com valores incorretos e veja o resultado.

```
conta.deposita(0);
```

Código Javascript A.39: Executando o método deposita()

17 Adicione o comando `try-catch` para capturar a exceção gerada.

```
try {
  conta.deposita(0);
} catch(e) {
  console.log(e.name);
  console.log(e.message);
}
```

Código Javascript A.40: Capturando exceções com `try-catch`

Exercícios Complementares

4 Crie uma função que faça a operação de divisão e após isso execute a função `divisao()`.

5 Crie uma função para a operação saque em um caixa eletrônico. Fique atento com as exceções. Feito isso, execute o método `saque()`.

6 Crie uma função que multiplique todos os argumentos passados como parâmetro. Após isso, execute a função com os seguintes números: 3, 6, 2, 8.

Arrays

Javascript provê um objeto com características semelhantes a um array. Para criar o objeto array, podemos criá-lo de forma literal.

```
var vazio = [];
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];

console.log(vazio[0]);
console.log(cursos[0]);

console.log(vazio.length);
console.log(cursos.length);
```

Código Javascript A.44: Criando um array

Percorrendo um Array

Para percorrer um array, podemos utilizar o comando `for`.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.45: for

Adicionando Elementos

Para adicionar um elemento ao final do array, podemos utilizar a propriedade `length`.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
cursos[cursos.length] = "K01";
for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.46: Adicionando elementos ao final do array com length

Ou você pode adicionar os elementos ao final do array utilizando o método `push()`.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
cursos.push("K01");
for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.47: Adicionando elementos através do método push

Removendo Elementos

O método `delete()` permite remover elementos de um array.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];

delete cursos[0];

for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.48: Delete

O método `delete()` deixa uma posição indefinida no array. Para corrigir este problema, o array tem o método `splice()`. O primeiro parâmetro desse método indica qual é o primeiro elemento que desejamos remover. O segundo indica quantos elementos deve ser removidos a partir do primeiro.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];

cursos.splice(0,2);

for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.49: Utilizando o método splice()

Concatenando Arrays

O método `concat()` permite concatenar dois arrays.

```
var formacao_java = ["K11","K12"];
var formacao_java_avancado = ["K21","K22","K23"];

var formacao_completa = formacao_java.concat(formacao_java_avancado);

for(var i = 0; i < formacao_completa.length; i++) {
  console.log(formacao_completa[i]);
}
```

Código Javascript A.50: Concatenando

Gerando uma String com os Elementos de um Array

O método `join()` cria uma string a partir de um array.

```
1 var formacao_java = ["K11","K12"];
2
3 var resultado = formacao_java.join(",");
4 console.log(resultado);
```

Código Javascript A.51: Gerando uma string com os elementos de um array

Removendo o Último Elemento

O método `pop()` remove e retorna o último elemento.

```
1 var cursos = ["K11","K12","K21","K22","K23"];
2
3 var curso = cursos.pop();
4 console.log(curso);
```

Código Javascript A.52: Removendo o último elemento

Adicionando um Elemento na Última Posição

O método `push()` adiciona um elemento ao final do array.

```
1 var cursos = ["K11","K12","K21","K22"];
2
3 cursos.push("K23");
4
5 for(var i = 0; i < cursos.length; i++) {
6   console.log(cursos[i]);
7 }
```

Código Javascript A.53: Adicionando um elemento na última posição

Invertendo os Elementos de um Array

O método `reverse()` inverte a ordem dos elementos de um array.

```
1 var cursos = ["K11","K12","K21","K22","K23"];
2
3 cursos.reverse();
4
5 for(var i = 0; i < cursos.length; i++) {
6   console.log(cursos[i]);
7 }
```

Código Javascript A.54: Invertendo os elementos de um array

Removendo o Primeiro Elemento

O método `shift()` remove e retorna o primeiro elemento de um array.

```
1 var cursos = ["K11","K12","K21","K22","K23"];
2
3 var curso = cursos.shift();
4
5 console.log("Elemento removido: " + curso);
```

```
6
7  for(var i = 0; i < cursos.length; i++) {
8    console.log(cursos[i]);
9  }
```

Código Javascript A.55: Removendo o primeiro elemento

Copiando um Trecho de um Array

O método `slice()` cria uma cópia de uma porção de um array.

```
1  var cursos = ["K11","K12","K21","K22","K23"];
2
3  var formacao_java = cursos.slice(0,2);
4
5  for(var i = 0; i < formacao_java.length; i++) {
6    console.log(formacao_java[i]);
7  }
```

Código Javascript A.56: Copiando um trecho de um array

Removendo e Adicionando Elementos em um Array

O método `splice()` permite remover elementos do array e adicionar novos elementos.

```
1  var cursos = ["K11","K12","K21","K22","K23"];
2
3  cursos.splice(2,3,"K31","K32");
4
5  for(var i = 0; i < cursos.length; i++) {
6    console.log(cursos[i]);
7  }
```

Código Javascript A.57: Substituindo elementos de uma array

Adicionando um Elemento na Primeira Posição

O método `unshift()` adiciona elementos na primeira posição de um array.

```
1  var cursos = ["K12","K21","K22","K23"];
2
3  cursos.unshift("K11");
4
5  for(var i = 0; i < cursos.length; i++) {
6    console.log(cursos[i]);
7  }
```

Código Javascript A.58: Adicionando um elemento na primeira posição

Métodos das Strings

Acessando os Caracteres de uma String por Posição

O método `charAt()` retorna o caractere na posição especificada.

```
1  var curso = "K12";
2
```

```
3 console.log(curso.charAt(0));
```

Código Javascript A.59: Acessando os caracteres de uma string por posição

Recuperando um Trecho de uma String

O método `slice()` retorna uma porção de uma string.

```
1 var curso = "K12 - Desenvolvimento Web com JSF2 e JPA2";
2
3 console.log(curso.slice(0,3));
```

Código Javascript A.60: Recuperando um Trecho de uma String

Dividindo uma String

O método `split()` cria uma array de strings a partir de um separador.

```
1 var curso = "K12-Desenvolvimento Web com JSF2 e JPA2";
2 var aux = curso.split("-");
3
4 console.log(aux[0]);
5 console.log(aux[1]);
```

Código Javascript A.61: Dividindo uma string

Exercícios de Fixação

18 Crie dois arrays e imprima no console do Firebug o tamanho deles.

```
1 var vazio = [];
2 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
3
4 console.log(vazio[0]);
5 console.log(cursos[0]);
6
7 console.log(vazio.length);
8 console.log(cursos.length);
```

Código Javascript A.62: Criando dois arrays e imprimindo o tamanho

19 Imprima os elementos de um array linha a linha.

```
1 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
2 for(var i = 0; i < cursos.length; i++) {
3   console.log(cursos[i]);
4 }
```

Código Javascript A.63: for

20 Adicione elementos no final de um array utilizando a propriedade `length`.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];

cursos[cursos.length] = "K01";

for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.64: Adicionando elementos ao final do array com length

21 Adicione elementos no final de um array utilizando o método `push()`.

```
var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
cursos.push("K01");
for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.65: Adicionando elementos através do método push

22 Concatene dois arrays através do método `concat()`.

```
var formacao_java = ["K11","K12"];
var formacao_java_avancado = ["K21","K22","K23"];

var formacao_completa = formacao_java.concat(formacao_java_avancado);

for(var i = 0; i < formacao_completa.length; i++) {
  console.log(formacao_completa[i]);
}
```

Código Javascript A.66: Concatenando

23 Remove o último elemento de um array com o método `pop()`.

```
var cursos = ["K11","K12","K21","K22","K23"];

var curso = cursos.pop();
console.log(curso);
```

Código Javascript A.67: Removendo o último elemento

24 Adicione um elemento no final de um array. Para isso, aplique o método `push()`.

```
var cursos = ["K11","K12","K21","K22"];

cursos.push("K23");

for(var i = 0; i < cursos.length; i++) {
  console.log(cursos[i]);
}
```

Código Javascript A.68: Adicionando um elemento na última posição

25 Inverta a ordem dos elementos de um array com o método `reverse()`.

```
1  var cursos = ["K11","K12","K21","K22","K23"];
2
3  cursos.reverse();
4
5  for(var i = 0; i < cursos.length; i++) {
6    console.log(cursos[i]);
7  }
```

Código Javascript A.69: Invertendo os elementos de um array

26 Remova o primeiro elemento de um array através do método `shift()`.

```
1  var cursos = ["K11","K12","K21","K22","K23"];
2
3  var curso = cursos.shift();
4
5  console.log("Elemento removido: " + curso);
6
7  for(var i = 0; i < cursos.length; i++) {
8    console.log(cursos[i]);
9  }
```

Código Javascript A.70: Removendo o primeiro elemento

27 Faça uma cópia de um determinado trecho de um array.

```
1  var cursos = ["K11","K12","K21","K22","K23"];
2
3  var formacao_java = cursos.slice(0,2);
4
5  for(var i = 0; i < formacao_java.length; i++) {
6    console.log(formacao_java[i]);
7  }
```

Código Javascript A.71: Copiando um trecho de um array

28 Adicione um elemento na primeira posição de um array. Utilize o método `unshift()`.

```
1  var cursos = ["K12","K21","K22","K23"];
2
3  cursos.unshift("K11");
4
5  for(var i = 0; i < cursos.length; i++) {
6    console.log(cursos[i]);
7  }
```

Código Javascript A.72: Adicionando um elemento na primeira posição

29 Divida o conteúdo de uma string aplicando o método `split()`.

```
1  var curso = "K12-Desenvolvimento Web com JSF2 e JPA2";
2  var aux = curso.split("-");
3
4  console.log(aux[0]);
5  console.log(aux[1]);
```

Código Javascript A.73: Dividindo uma string

Exercícios Complementares

7 Crie dois arrays e imprima no console do Firebug o tamanho deles e também imprimir todos os elementos deste array linha a linha.

8 Crie um array com alguns valores e depois faça com que adicione um elemento no final do array utilizando a propriedade `length`.

9 Altere o arquivo do exercício anterior para adicionar elemento no final do array utilizando o método `push()`.

10 Crie dois arrays e concatene-os usando o método `concat()`.

11 Crie um array contendo alguns elementos e remova o último elemento através do método `pop()`.

12 Utilizando o arquivo do exercício anterior, adicione um elemento no final do array através do método `push()`.

13 Crie um array com alguns elementos e inverta a ordem dos elementos com o método `reverse()`.

14 Utilizando o arquivo do exercício anterior, remova o primeiro elemento do array utilizando o método `shift()`.

15 Utilizando o arquivo do exercício anterior, adicione novamente o elemento removido que estava na primeira posição do array através do método `unshift()`.

16 Crie um array contendo alguns elementos e faça uma cópia de um determinado trecho do desse array utilizando o método `slice()`.

17 Crie uma variável contendo uma string e divida o conteúdo utilizando o método `slipt()`.

jQuery

APÊNDICE B

Introdução

jQuery é uma biblioteca de funções JavaScript. Ela foi desenvolvida para simplificar e diminuir a quantidade de código JavaScript.

As principais funcionalidades da biblioteca JavaScript jQuery são:

- Seletores de elementos HTML
- Manipulação de elementos HTML
- Manipulação de CSS
- Funções de eventos HTML
- Efeitos e animações JavaScript
- AJAX

Para a lista completa de funcionalidades, acesse: http://docs.jquery.com/.

Para utilizar a biblioteca JavaScript jQuery, basta adicionar a referência para o arquivo `js` através da tag *<script>*. O download do arquivo `js` do jQuery pode ser feito através do seguinte endereço http://docs.jquery.com/Downloading_jQuery. Há duas opções de arquivo para download, o *Minified* e *Uncompressed*, você pode utilizar qualquer um.

```
1 <head>
2   <script type="text/javascript" src="jquery.js"></script>
3 </head>
```

Código HTML B.1: Referência pro jQuery

Caso você não queira fazer o download do arquivo `js` do jQuery, é possível utilizar a url de alguma empresa que hospede o arquivo jQuery e permita o uso público. Empresas como o Google e Microsoft proveem endereços para a utilização da biblioteca jQuery.

```
1 <head>
2   <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery↩
        /1.7.2/jquery.min.js">
3   </script>
4 </head>
```

Código HTML B.2: Referência pro jQuery através da url do Google

```
1 <head>
2   <script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.2.min.js">
3   </script>
4 </head>
```

Código HTML B.3: Referência pro jQuery através da url da Microsoft

Sintaxe

A sintaxe da biblioteca jQuery permite facilmente selecionar elementos HTML e executar alguma ação sobre eles.

A sintaxe básica para executar uma ação sobre determinados elementos é: `$(seletor).acao()`.

- O símbolo $ é um método de fabricação para criar o objeto jQuery;
- O `(seletor)` serve para consultar e encontrar os elementos HTML;
- A `acao()` define a operação jQuery que será executada nos elementos.

Mais Sobre
O método de fabricação é um padrão de projeto de criação. Para saber mais sobre padrões de projeto, confira a apostila de Design Patterns da K19 através do endereço http://www.k19.com.br/downloads/apostilas/java/k19-k51-design-patterns-em-java.

Exemplos:

```
1 $(this).hide() //esconde o elemento que o this faz referência
2
3 $("p").hide() //esconde todos os elementos <p>
4
5 $("p.curso").hide()//esconde todos os elementos <p> que tem a classe "curso"
6
7 $("#cursok31").hide()//esconde o elemento de id "cursok31"
```

Código Javascript B.1: Exemplos jQuery

Seletores

Os seletores do jQuery permitem manipular um conjunto de elementos ou um apenas elemento HTML.

O jQuery suporta os seletores CSS existentes mais os seus próprios seletores. Para conferir os seletores CSS existentes, acesse: http://www.w3.org/community/webed/wiki/CSS/Selectors.

Para selecionar um conjunto ou apenas um elemento HTML, a sintaxe utilizada contém o prefixo $ e os parênteses (): `$()`.

Exemplo:

```
$(this) //seleciona o elemento que o this referencia$

$("*") //seleciona todos elementos do documento HTML$

$("div") //seleciona todos os elementos <div>$

$(".curso") //seleciona todos os elementos cuja classe é "curso"$

$("#cursok31") //seleciona um único elemento de id "cursok31"$

$('input[name|="curso"]')$
  //seleciona todos os elementos <input> que contém o
  //atributo name igual a "curso" ou o prefixo "curso" seguido de traço (-).
$('input[name*="curso"]')$
  //seleciona todos os elementos <input> nas quais a
  //palavra "curso" faz parte do atributo name.
$('input[name~="curso"]')$
  //seleciona todos os elementos <input> nas quais a
  //palavra "curso" faz parte do atributo name delimitado por espaço.
$('input[name="k32"]')$
  //seleciona todos os elementos <input> cujo atributo
  //name termina com a palavra k32
$('input[name="Curso K32"]')$
  //seleciona todos os elementos <input> que o
  //atributo name tenha exatamente a palavra "Curso K32".
$('input[name!="curso"]')$
  //seleciona todos os elementos <input> que o atributo
  //name tenha valor diferente de "curso"
$('input[name^="curso"]')$
  //seleciona todos os elementos <input> que o atributo
  //name comece exatamente com a palavra "curso"
```

Código Javascript B.2: Seletores CSS

Para uma lista completa de seletores do jQuery, acesse: http://api.jquery.com/category/selectors/.

Exercícios de Fixação

1. Crie uma pasta na Área de Trabalho de trabalho com o seu nome.

2. Na pasta com o seu nome, crie uma pasta chamada **seletores**.

3. Crie um arquivo chamado **seletores.html** conforme o código abaixo.

```
<!DOCTYPE html>
<html>
  <head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
  <input name="curso-k31" class="curso"/>

  <input name="curso-k32" class="curso"/>
  <input name="formacao-net" class="formacao"/>
  <input name="formacao-java" class="formacao"/>
</body>
```

```
13  </html>
```

Código HTML B.4: seletores.html

4 Altere o arquivo `seletores.html` para que os campos cuja classe é `curso` tenha o valor igual a "K19".

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7     <input name="curso-k31" class="curso"/>
8
9     <input name="curso-k32" class="curso"/>
10    <input name="formacao-net" class="formacao"/>
11    <input name="formacao-java" class="formacao"/>
12  <script>$('.curso').val('K19');</script>$
13    </body>
14  </html>
```

Código HTML B.5: curso.html

5 Altere o exercício anterior, para selecionar os elementos cujo atributo `name` comece com a palavra `formacao` seguida de `traço (-)`.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7       <input name="curso-k31" class="curso"/>
8
9       <input name="curso-k32" class="curso"/>
10      <input name="formacao-net" class="formacao"/>
11      <input name="formacao-java" class="formacao"/>
12    <script>$('input[name|="formacao"]').val('K19');</script>$
13    </body>
14  </html>
```

Código HTML B.6: curso.html

6 Selecione todos os elementos <input> e atribua o valor "K19".

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7       <input name="curso-k31" class="curso"/>
8
9       <input name="curso-k32" class="curso"/>
10      <input name="formacao-net" class="formacao"/>
11      <input name="formacao-java" class="formacao"/>
12    <script>$('input').val('K19');</script>$
13    </body>
14  </html>
```

Código HTML B.7: curso.html

Exercícios Complementares

1. Crie um outro arquivo HTML chamado `seletores-2.html` na pasta **seletores**. Utilize a biblioteca `JavaScript JQuery` e adicione no mínimo 5 nomes com 5 classes.

2. Altere o arquivo `seletores-2.html` para que os campos cuja classe é "Gerente"mude para um valor igual a "Instrutor da K19".

3. Altere o arquivo `seletores-2.html` para selecionar os elementos cujo atributo `name` comece com vogais.

4. Altere o arquivo `seletores-2.html` para selecionar todos os elementos `<input>` e atribua o valor "K19".

Eventos

Os eventos são métodos que são chamados quando o usuário interage com o navegador.

Para registrar os eventos, podemos utilizar os seletores do jQuery visto na seção anterior.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
  <h2>Formação .NET K19:</h2>
  <ul>
  <li>Curso K31 - C# e Orientação a Objetos</li>
  <li>Curso K32 - Desenvolvimento Web com ASP .NET MVC</li>
  </ul>
  <script>$("li").click(function () {
      alert("Elemento li clicado: " + $(this).text());
    });
  </script>
  </body>
</html>
```

Código HTML B.12: Eventos jQuery

No exemplo acima, registramos todos os elementos <li> com o evento de clique. Quando o usuário clicar no elemento <li>, devemos definir uma função que será chamada, a esta função damos o nome de **função de callback**.

```
$("li").click( função de callback... )$
```

Código Javascript B.3: Evento de clique

A função de callback definida foi:

```
function () {
  alert("Elemento li clicado: " + $(this).text());$
}
```

Código Javascript B.4: Função de callback

Segue abaixo exemplos de eventos do jQuery:

Evento	Descrição
`$(document).ready(função de callback...)`	A função de callback é chamada quando o DOM é carregado por completo
`$(seletor).click(função de callback...)`	A função de callback será chamada quando o usuário clicar no elementos selecionados
`$(seletor).dblclick(função de callback...)`	A função de callback será chamada quando o usuário clicar 2X (duas) vezes nos elementos selecionados
`$(seletor).focus(função de callback...)`	A função de callback será chamada quando o foco estiver nos elementos selecionados
`$(seletor).change(função de callback...)`	A função de callback será chamada quando o usuário alterar o valor dos elementos selecionados

Para uma lista completa de eventos do jQuery, acesse http://api.jquery.com/category/events/.

Exercícios de Fixação

7. Crie uma pasta **eventos** dentro da pasta com o seu nome que foi criada na Área de Trabalho.

8. Crie um arquivo `eventos.html` conforme o código abaixo.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
  </body>
</html>
```

Código HTML B.13: eventos.html

9 Altere o arquivo `eventos.html` para adicionar o evento de clique aos elementos <li>. Quando o usuário clicar deverá ser mostrado uma mensagem de alerta com o conteúdo do elemento clicado.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
  <script>
  $('li').click(function(){ alert("Elemento li clicado: "
  +$(this).text());});
  </script>
  </body>
</html>
```

Código HTML B.14: eventos.html

10 Altere o exercício anterior para mostrar a mensagem de alerta após o usuário efetuar um duplo clique.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
 <script>
  $('li').dblclick(function(){ alert("Elemento clicado
  2X:"+$(this).text());});
 </script>
  </body>
</html>
```

Código HTML B.15: eventos.html

11 Altere o exercício anterior e altere o conteúdo de todos os elementos <li> após o carregamento completo do DOM.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script>
  $(document).ready(function () {
    $('li').text('DOM carregado por completo.');
  });
  </script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
</body>
</html>
```

Código HTML B.16: eventos.html

12 Altere o arquivo `eventos.html` e adicione uma caixa de seleção de cursos. Caso o usuário escolha um curso, mostre uma mensagem de alerta indicando o curso escolhido.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <label for="cursos">Selecione um curso:</label>
    <select name="cursos" id="cursos">
            <option>----</option>
      <option value="K31">K31 - C# e Orientação a Objetos</option>
  ***   <option value="K32">K32 - Desenvolvimento Web com ASP .NET MVC</option>
      <option value="K11">K11 - Java e Orientação a Objetos</option>
  ***   <option value="K12">K12 - Desenvolvimento Web com JSF2 e JPA2</option>
    </select>
  </div>
  <script>
  $('#cursos').change(function(){
  alert("Curso selecionado: "+$("#cursos option:selected")
  .text());
  });
  </script>
</body>
</html>
```

Código HTML B.17: eventos.html

Exercícios Complementares

5 Crie um outro arquivo HTML chamado `eventos-2.html` na pasta **eventos**, utilizando a biblioteca `JavaScript JQuery`. Nesse arquivo, crie uma lista ordenada contendo o nome das formações da K19.

6 Altere o arquivo `eventos-2.html` para adicionar o evento de clique aos elementos `<li>`. Com esse evento, mostre o alerta com a mensagem do conteúdo do elemento clicado.

7 Altere o arquivo `eventos-2.html` para adicionar o evento de alerta após o duplo clique no elemento selecionado.

8 No arquivo `eventos-2.html`, faça as alterações necessárias para que contenha uma caixa de seleção das formações da K19. Quando uma formação for escolhida, mostre uma mensagem de alerta indicando o nome da formação desejada.

Efeitos

A biblioteca jQuery contém vários métodos para adicionar animação para a página web.

jQuery Hide e Show

Com o jQuery você pode ocultar elementos da página ou torná-los visíveis através dos métodos `hide` e `show`.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
    <button id="mostrar">Mostrar</button>
    <button id="ocultar">Ocultar</button>

    <p style="display: none">Cursos K19</p>
  <script>
      $("#mostrar").click(function () {
      $("p").show("slow");
      });
      $("#ocultar").click(function () {
      $("p").hide("slow");
      });
    </script>
  </body>
</html>
```

Código HTML B.22: Métodos show e hide do jQuery

jQuery Toggle

O método jQuery Toggle altera a visibilidade dos elementos através dos métodos `show` e `hide`.

Elementos visíveis são ocultados e elementos ocultados tornam-se visíveis.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
  <button>Mostrar/Ocultar</button>

  <p style="display: none">Cursos K19</p>
  <script>
    $("button").click(function () {
      $("p").toggle("slow");
      });
    </script>
  </body>
</html>
```

Código HTML B.23: jQuery Toggle

jQuery Fade

O `jQuery fade` altera a opacidade dos elementos HTML. Os métodos `fade` do jQuery são três:

$(seletor).fadeIn(speed,callback)

$(seletor).fadeOut(speed,callback)

$(seletor).fadeTo(speed,opacidade,callback)

O primeiro parâmetro speed aceita os seguintes valores: "slow", "fast", "normal"ou milissegundos.

O parâmetro callback define a função que será chamada após o evento de “slide” terminar.

O parâmetro de opacidade do método `fadeTo` define em porcentagem a opacidade do elemento HTML.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
  <button id="fadein">Fade in</button>
  <button id="fadeout">Fade out</button>
  <button id="fadeto">Fade to</button>
  <p>Cursos K19</p>
  <script>
    $("#fadein").click(function () {
    $("p").fadeIn();
    });
```

```
18      $("#fadeout").click(function () {
19      $("p").fadeOut();
20      });
21      $("#fadeto").click(function () {
22      $("p").fadeTo("normal", 0.30);
23      });
24    </script>
25    </body>
26  </html>
```

Código HTML B.24: jQuery fade

jQuery Slide

O `jQuery Slide` permite alterarmos a altura dos elementos HTML. O jQuery Slide tem 3 (três) métodos:

$(selector).slideDown(speed,callback)

$(selector).slideUp(speed,callback)

$(selector).slideToggle(speed,callback)

O primeiro parâmetro speed aceita os seguintes valores: "slow", "fast", "normal"ou milissegundos.

O parâmetro callback define a função que será chamada após o evento de “slide” terminar.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4     <style>
5       p { background:green; }
6     </style>
7       <script src="http://code.jquery.com/jquery-latest.js"></script>
8     </head>
9     <body>
10      <button id="slidedown">Slide Down</button>
11      <button id="slideup">Slide Up</button>
12      <button id="slidetoggle">Slide Toggle</button>
13
14      <p>Cursos K19</p>
15    <script>
16      $("#slidedown").click(function () {
17      $("p").slideDown();
18      });
19      $("#slideup").click(function () {
20      $("p").slideUp();
21      });
22      $("#slidetoggle").click(function () {
23      $("p").slideToggle("slow");
24      });
25    </script>
26    </body>
27  </html>
```

Código HTML B.25: jQuery Slide

Para uma lista completa de efeitos do jQuery, acesse http://api.jquery.com/category/effects/.

Exercícios de Fixação

13 Crie uma pasta **efeitos** dentro da pasta com o seu nome.

14 Crie um arquivo `efeitos.html` dentro da pasta **efeitos** conforme o código abaixo.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
  </body>
</html>
```

Código HTML B.26: efeitos.html

15 Altere o arquivo `efeitos.html` e adicione botões para ocultar e mostrar os elementos <li> da página.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
  <div>
    <button id="mostrar">Mostrar</button>
    <button id="ocultar">Ocultar</button>
  </div>
  <script>
  $("#mostrar").click(function(){ $("li").show("slow");});
  $("#ocultar").click(function(){ $("li").hide("slow");});
  </script>
  </body>
</html>
```

Código HTML B.27: efeitos.html

16 Altere o exercício anterior e adicione um botão que mostra os elementos <li> ocultos e oculta os

elementos <li> visíveis.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>

  <div>
  <button id="mostrar-ocultar">Mostrar/Ocultar</button>
  </div>
  <script>
  $("#mostrar-ocultar").click(function(){ $("li").toggle
  ("slow"); });
  </script>
  </body>
</html>
```

Código HTML B.28: efeitos.html

17 Altere o exercício anterior para ocultar e mostrar os elementos <li> através dos métodos do jQuery que alteram a opacidade.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <h1>Cursos K19</h1>
  <div>
    <ul>
      <li>K31 - C# e Orientação a Objetos</li>
      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
      <li>K11 - Java e Orientação a Objetos</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
  </div>
  <div>
    <button id="mostrar">Mostrar</button>
    <button id="ocultar">Ocultar</button>
  </div>
  <script>
  $("#mostrar").click(function(){ $("li").fadeIn("slow"); });
  $("#ocultar").click(function(){ $("li").fadeOut("slow"); });
  </script>
  </body>
</html>
```

Código HTML B.29: efeitos.html

Exercícios Complementares

9 Altere o exercício anterior e acrescente um botão para diminuir a opacidade dos elementos <li> para 0.2.

10 Altere o arquivo efeitos.html para ocultar e mostrar os elementos alterando a altura deles.

11 Altere o arquivo efeitos.html e adicione um botão que oculta os elementos <li> visíveis e mostra os elementos <li> ocultos. Para mostrar e ocultar, utilize o método do jQuery que altera a altura dos elementos.

12 Crie um outro arquivo HTML na pasta **efeitos** e crie uma lista ordenada com as formações da K19. Faça as alterações necessárias para adicionar botões para ocultar e mostrar os elementos que estão no <li>.

13 No arquivo criado anteriormente, altere os elementos <li> para ocultar e mostrar através do método de fade-in/fade-out.

HTML

A biblioteca jQuery contém métodos para alterar e manipular os elementos HTML da página.

Para alterar o conteúdo dos elementos HTML da página, podemos usar o método html().

```
1 $("p").html("K19 Treinamentos");$
```

Código Javascript B.5: Método html()

Para adicionar conteúdo HTML, podemos usar os métodos append() e prepend().

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4    <style>
5      p { background:green; }
6    </style>
7    <script src="http://code.jquery.com/jquery-latest.js"></script>
8    </head>
9    <body>
10     <button id="prepend">Prepend</button>
11     <button id="append">Append</button>
12
13   <p>Cursos K19</p>
14   <script>
15       $("#prepend").click(function () {
16       $("p").prepend("K19 Treinamentos - ");
17       });
18       $("#append").click(function () {
19       $("p").append(" - K31");
20       });
21   </script>
22   </body>
23 </html>
```

Código HTML B.35: Método append() e prepend()

Para adicionar o conteúdo antes ou depois dos elementos HTML, podemos utilizar os métodos `after()` e `before()`.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
  p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
   <button id="before">Before</button>
   <button id="after">After</button>

  <p>Cursos K19</p>
  <script>
     $("#before").click(function () {
      $("p").before("K19 Treinamentos - ");
    });
    $("#after").click(function () {
      $("p").after(" - K31");
    });
  </script>
  </body>
</html>
```

Código HTML B.36: Métodos after() e before()

Para uma lista completa de métodos para manipular e alterar elementos HTML com o jQuery, acesse http://api.jquery.com/category/manipulation/.

Exercícios de Fixação

18 Crie uma pasta `html` dentro da pasta com o seu nome.

19 Crie um arquivo `html.html` dentro da pasta `html` conforme o código abaixo.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <p>Cursos K19</p>
  </body>
</html>
```

Código HTML B.37: html.html

20 Altere o arquivo `html.html` e adicione um botão alterar o conteúdo do elemento <p> para "Treinamentos da K19".

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

  <p>Cursos K19</p>
    <div>
    <button id="alterar">Alterar conteúdo</button>
    </div>
    <script>
      $("#alterar").click(function(){$("p").html("Treinamentos da K19");});
    </script>
  </body>
</html>
```

Código HTML B.38: html.html

21 Altere o arquivo `html.html` e adicione botões para adicionar conteúdo antes e depois do conteúdo do elemento <p>.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
    <p>Cursos K19</p>
  <div>
    <button id="prepend">Prepend</button>
    <button id="append">Append</button>
  </div>
  <script>
  $("#prepend").click(function(){$("p").prepend("Formação Desenvolvedor Java - ");});
  $("#append").click(function(){$("p").append(" - Formação Desenvolvedor .NET");});
  </script>
  </body>
</html>
```

Código HTML B.39: html.html

22 Altere o exercício anterior para adicionar o conteúdo antes e depois do elemento <p>.

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    p { background:green; }
  </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
<body>
  <p>Cursos K19</p>
```

```
11    <div>
12    <button id="before">Before</button>
13    <button id="after">After</button>
14    </div>
15    <script>
16    $("#before").click(function(){$("p").before("Formação Desenvolvedor Java - ");});
17    $("#after").click(function(){$("p").after(" - Formação Desenvolvedor .NET");});
18    </script>
19    </body>
20  </html>
```

Código HTML B.40: html.html

Exercícios Complementares

14 Crie um outro arquivo HTML chamado `html-2.html` na pasta **html** contendo a biblioteca `JavaScript JQuery`. Nesse arquivo, coloque um elemento contendo uma frase qualquer. Após isso, adicione um botão para alterar o conteúdo desse elemento para "K19 Treinamentos."

15 Faça alterações no arquivo do exercício anterior para criar dois botões que quando clicados adicionam um texto como conteúdo. Crie um antes e outro depois do elemento criado. Dica: não é necessário apagar o botão criado anteriormente.

16 Altere no arquivo `html-2.html` para adicionar o conteúdo antes e depois do elemento <p>.

HTML5

APÊNDICE C

O HTML5 é uma linguagem utilizada para apresentar e estruturar o conteúdo de páginas web. Trata-se da quinta versão da linguagem HTML e com ela vieram diversas mudanças como novas tags, novos atributos e novas APIs. As diferenças com relação à versão anterior não ficam presas apenas àquilo que é novo, pois muito daquilo que já existia também foi alterado. Algumas tags tiveram sua semântica alterada e algumas propriedades deixaram de existir, por exemplo.

Com relação a sintaxe do HTML5, podemos seguir o padrão do HTML ou do XHTML, porém não podemos misturá-los em um mesmo documento.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5    </head>
6    <body>
7      ...
8    </body>
9  </html>
```

Código HTML C.1: Exemplo de um documento HTML5 utilizando a sintaxe HTML

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml">
4    <head>
5      <meta charset="ISO-8859-1" />
6    </head>
7    <body>
8      ...
9    </body>
10 </html>
```

Código HTML C.2: Exemplo de um documento HTML5 utilizando a sintaxe XHTML

Lembre-se
A linha destacada no exemplo do uso da sintaxe XHTML é necessária apenas quando o encode do documento for diferente de UTF-8. Quando o encode for UTF-8 o W3C recomenda que essa linha seja omitida.

Importante
Documentos HTML5 que utilizarem a sintaxe HTML deverão ser transmitidos com o Content-Type `text/html` enquanto que os que utilizarem a sintaxe XHTML deverão ser transmitidos com o Content-Type `application/xhtml+xml`.

article

Representa uma composição independente em um documento, página ou aplicação e que é, a princípio, independentemente distribuível ou reutilizável. Pode ser um post de um fórum, artigo de uma revista ou jornal, post de um blog, comentário enviado por um usuário, widget interativo ou qualquer item independente de conteúdo.

Quando encontramos elementos `article` aninhados, os elementos `article` internos representam um conteúdo que esteja relacionado ao conteúdo do ancestral `article` mais próximo. Por exemplo, elementos `article` internos podem representar os comentários de um post de um blog Neste caso o post do blog seria o elemento `article` ancestral mais próximo.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <article>
      <h1>Primeiro post do blog</h1>
      <p>Loren ipsum...</p>

      <h2>Comentários</h2>
      <article>Legal este post!</article>
      <article>Bacana este post!</article>
      <article>Da hora este post!</article>
    </article>
  </body>
</html>
```

Código HTML C.3: article.html

section

Representa uma seção genérica de um documento. Neste contexto, uma seção é o agrupamento de um conteúdo dentro de um tema que, normalmente, possui um cabeçalho.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <section>
      <h1>Últimos Posts</h1>
      <article>
        <h1>Primeiro post do blog</h1>
        <p>Loren ipsum...</p>
      </article>

      <article>
        <h1>Segundo post do blog</h1>
        <p>Loren ipsum...</p>
      </article>

      <article>
        <h1>Terceiro post do blog</h1>
        <p>Loren ipsum...</p>
```

```
22        </article>
23      </section>
24    </body>
25  </html>
```

Código HTML C.4: section.html

header

O elemento header é utilizado para definir um conteúdo de introdução ou de navegação. Normalmente encontramos elementos de cabeçalho (h1 ~ h6), em seu conteúdo. É importante lembrar que, apesar de geralmente ser empregado no começo da página, seu uso pode ser feito também em diferentes seções do mesmo documento.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8">
5     </head>
6     <body>
7       <header>
8         <h1>Blog da K19</h1>
9       </header>
10      <section>
11        <h1>Meus Posts</h1>
12
13        ...
14      </section>
15    </body>
16  </html>
```

Código HTML C.5: header.html

footer

Representa o rodapé do ancestral de seccionamento mais próximo. É muito comum encontrarmos em seu conteúdo informações sobre a seção a qual ele pertence como quem a escreveu, links relacionados ao conteúdo da seção e informações legais, por exemplo.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8">
5     </head>
6     <body>
7       <header>
8         <h1>Blog da K19</h1>
9       </header>
10
11      <article>
12        <header>
13          <h1>Primeiro post</h1>
14        </header>
15
16        <p>Loren ipsum...</p>
17
18        <footer>Postado por: Jonas Hirata</footer>
19      </article>
```

```
20
21      <footer>&copy;2013 K19 Treinamentos.</footer>
22    </body>
23  </html>
```

Código HTML C.6: footer.html

nav

A tag nav representa uma seção da página que contém links para outras páginas ou para outras partes do documento, ou seja, uma seção com links de navegação.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8">
5     </head>
6     <body>
7       <header>
8         <h1>Blog da K19</h1>
9
10        <nav>
11          <a href="#">Home</a>
12          <a href="#">Últimos posts</a>
13          <a href="#">Arquivo</a>
14        </nav>
15      </header>
16
17      <section>
18        <h1>Meus Posts</h1>
19
20        ...
21      </section>
22
23      <footer>&copy;2013 K19 Treinamentos.</footer>
24    </body>
25  </html>
```

Código HTML C.7: nav.html

aside

O elemento aside representa uma seção que consiste em um conteúdo que esteja tangencialmente relacionado ao conteúdo que está à sua volta. Essa seção é frequentemente representada como uma coluna lateral em relação ao conteúdo principal de uma página.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8">
5     </head>
6     <body>
7       <header>
8         <h1>Blog da K19</h1>
9
10        <nav>
11          <a href="#">Home</a>
12          <a href="#">Últimos posts</a>
13          <a href="#">Arquivo</a>
```

```
14        </nav>
15      </header>
16
17      <section>
18        <h1>Meus Posts</h1>
19
20        ...
21      </section>
22
23      <aside>
24        <h1>Posts relacionados</h1>
25        ...
26      </aside>
27
28      <footer>&copy;2013 K19 Treinamentos.</footer>
29    </body>
30  </html>
```

Código HTML C.8: aside.html

figure

Representa um conteúdo que é auto-suficiente e tipicamente referenciado como uma unidade singular do fluxo principal do documento. Opcionalmente o conteúdo pode possuir uma legenda.

Pode ser utilizado para exibir ilustrações, diagramas, fotos, vídeos, códigos fonte, etc, que são referenciados no conteúdo principal do documento, porém podem, sem afetar o fluxo do documento, ser removidas do conteúdo principal e colocadas, por exemplo, como anotações no canto da página ou em forma de apêndice.

figcaption

A tag `figcaption` deve ser filha de um elemento `figure` e representa a legenda para o resto do conteúdo do elemento pai. Além disso, a tag `figcaption` deve aparecer como primeira ou última filha de um elemento `figure`.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8">
5     </head>
6     <body>
7       <header>
8         <h1>Blog da K19</h1>
9
10        <nav>
11          <a href="#">Home</a>
12          <a href="#">Últimos posts</a>
13          <a href="#">Arquivo</a>
14        </nav>
15      </header>
16
17      <section>
18        <h1>Meus Posts</h1>
19
20        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum enim est,↩
              ultrices ac
21        vehicula vitae, pharetra id mauris. Quisque sit amet nisl mollis sem fermentum ↩
              accumsan.
```

```
22        Sed sed quam nisi, cursus sodales metus. Curabitur dapibus, massa sed ↩
              sollicitudin viverra,
23        odio justo dignissim metus, vel tempor turpis neque id erat. Aenean fermentum ↩
              ultricies
24        ante a luctus.</p>
25
26        <figure>
27          <img src="#" title="Aenean fermentum" alt="Mauris convallis, leo sollicitudin"↩
                >
28          <figcaption>Figura 3b. Curabitur dapibus, massa sed sollicitudin.</figcaption>
29        </figure>
30
31        <p>Mauris fermentum lorem nec nisi euismod elementum. Aenean nec magna dolor, ↩
              vel fermentum
32        turpis. Mauris convallis, leo sollicitudin egestas malesuada, nunc est ultrices ↩
              enim, eget
33        varius odio felis et velit. Sed ac lorem nibh, ut convallis ante.</p>
34      </section>
35
36      <footer>&copy;2013 K19 Treinamentos.</footer>
37    </body>
38  </html>
```

Código HTML C.9: figure-e-figcaption.html

CSS3

APÊNDICE D

A terceira versão da especificação do CSS, normalmente referida como CSS3, traz novos recursos que facilitará a vida dos desenvolvedores front-end.

Criar bordas arredondadas, aplicar sombra ou rotacionar elementos HTML e usar fontes diferentes em textos sempre foi uma tarefa trabalhosa que, quando possível resolver apenas em código, necessitava muito código JavaScript ou "gambiarras" em CSS.

Neste capítulo iremos abordar algumas das novas propriedades introduzidas pelo CSS3. Devemos nos lembrar que, assim como o HTML5, a especificação do CSS3 ainda não foi concluída, portanto qualquer recurso do CSS3 deve ser usado com cuidado para evitar resultados inesperados em diferentes navegadores.

Bordas arredondadas

A propriedade CSS `border-radius` é responsável por aplicar bordas arredondadas em um elemento HTML.

```
1  #main {
2    border-radius: 10px;
3  }
4
5  #menu {
6    border-radius: 10px 20px;
7  }
8
9  #side {
10   border-radius: 40px 30px 20px 10px;
11 }
```

Código CSS D.1: Propriedade border-radius

No exemplo acima, o elemento `#main` terá bordas arredondadas com 10 pixels de raio nos quatro cantos.

O elemento `#menu` terá bordas arredondadas com 10 pixels de raio no canto superior esquerdo, assim como no canto inferior direito. No canto superior direito e no canto inferior esquerdo o elemento terá bordas arredondadas com 20 pixels de raio.

O elemento `#side` terá bordas arredondadas com 40 pixels de raio no canto superior esquerdo, 30 pixels de raio no canto superior direito, 20 pixels de raio no canto inferior direito e 10 pixels no canto inferior esquerdo.

Além da propriedade `border-radius`, o CSS3 nos oferece as propriedades da lista abaixo para ajustar o raio da borda de cada um dos cantos de um elemento de maineira independente:

- border-top-left-radius: define o raio da borda do canto superior esquerdo.
- border-top-right-radius: define o raio da borda do canto superior direito.
- border-bottom-right-radius: define o raio da borda do canto inferior direito.
- border-bottom-left-radius: define o raio da borda do canto inferior esquerdo.

```
1 #content {
2   border-top-left-radius: 10px;
3   border-top-right-radius: 20px 10px; /* 20px na horizontal e 10px na vertical */
4 }
```

Código CSS D.2: Propriedades border-top-left-radius e border-top-right-radius

Um recurso interessante que ganhamos ao ajustarmos as bordas de maneira independente é a possibilidade de criarmos bordas com mais de um raio, como podemos observar na linha 3 do código acima. O efeito disso é uma borda elíptica ao invés de circular.

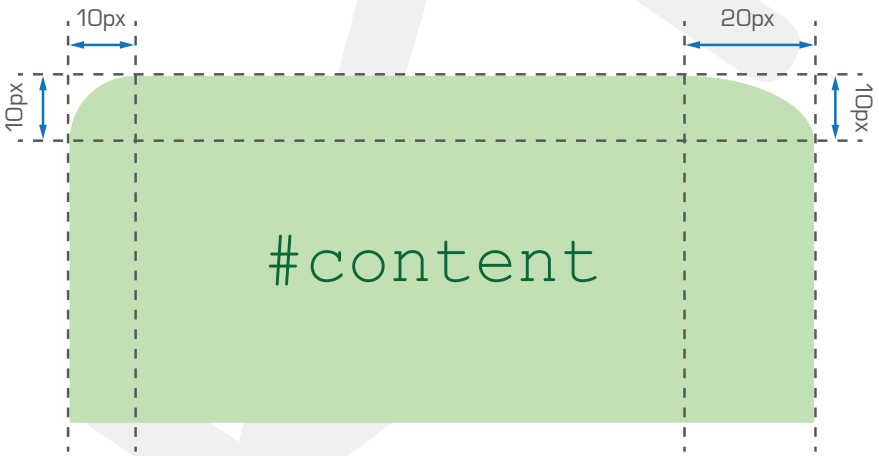

Figura D.1: Exemplo de borda com raios diferentes

Sombras

No CSS3 existem dois tipos de sombras: a sombra que é aplicada à "caixa" do elemento e a sombra que é aplicada no texto de um elemento.

box-shadow

A propriedade `box-shadow` é responsável por aplicar uma sombra na parte externa ou interna da "caixa" de um elemento. A "caixa" é limitada pelo retângulo definido pela borda do elemento de acordo com o box model. Além disso, o valor da propriedade pode ser definido de diversas maneiras. Confira a ilustração abaixo na qual mostramos como atribuir um valor para a propriedade `box-shadow`.

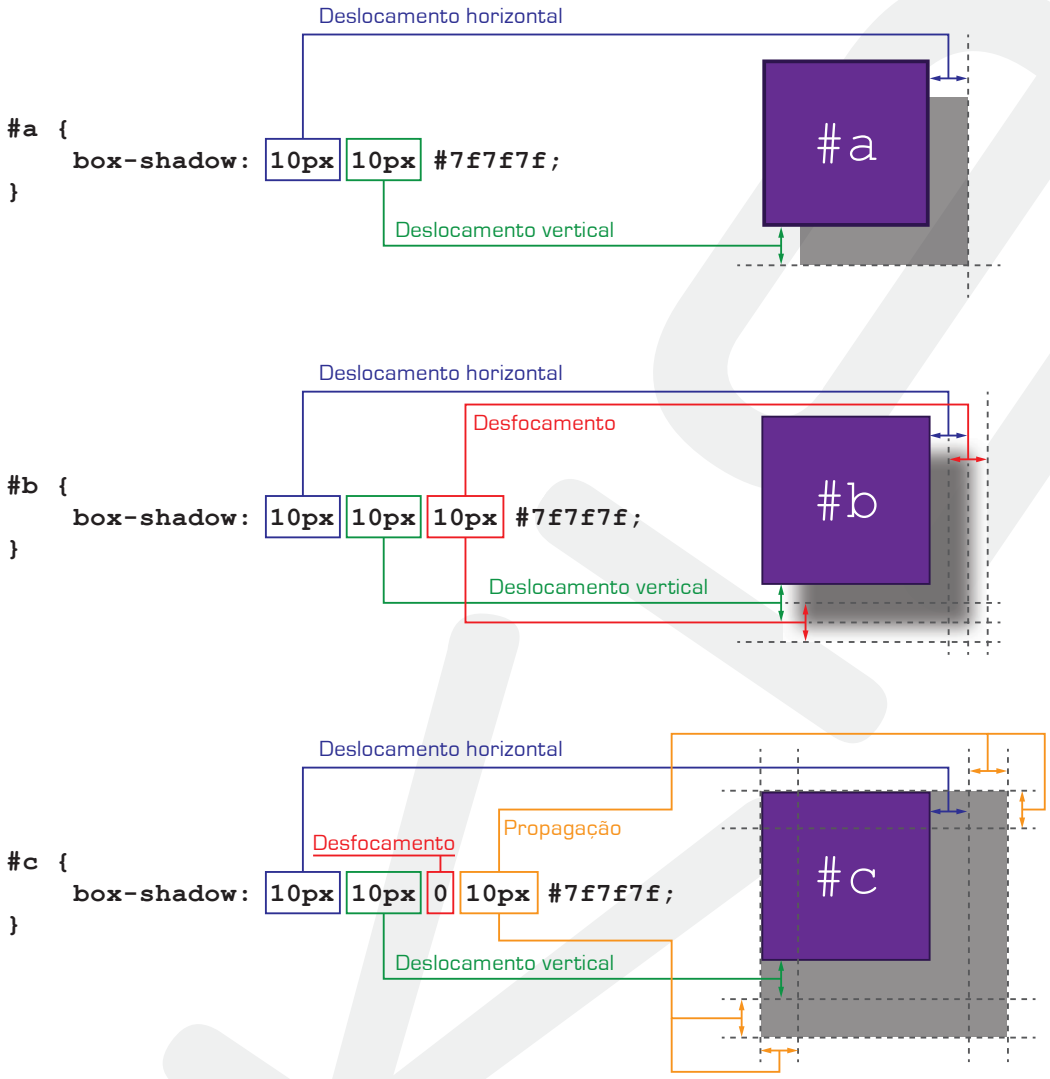

Figura D.2: Funcionamento da propriedade box-shadow para sombra externa

Os valores para os deslocamentos vertical e horizontal podem assumir valores negativos fazendo com que a sombra seja projetada para cima e para a esquerda.

Observe no código CSS abaixo valores aplicados na propriedade box-shadow.

```
1  #a {
2    box-shadow: 10px 10px #000000;
3  }
4
5  #b {
6    box-shadow: 10px 10px 10px #000000;
7  }
8
9  #c {
10   box-shadow: 10px 10px 0 10px #000000;
11 }
12
13 #d {
14   box-shadow: -10px -10px #000000;
15 }
16
```

```
#e {
  box-shadow: -10px -10px 10px #000000;
}

#f {
  box-shadow: -10px -10px 0 10px #000000;
}

#g {
  box-shadow: 0 0 10px #000000;
}

#h {
  box-shadow: 0 0 10px 10px #000000;
}
```

Código CSS D.3: Exemplos de uso da propriedade box-shadow para sombra externa

Em um navegador, o resultado da aplicação das regras CSS acima seria:

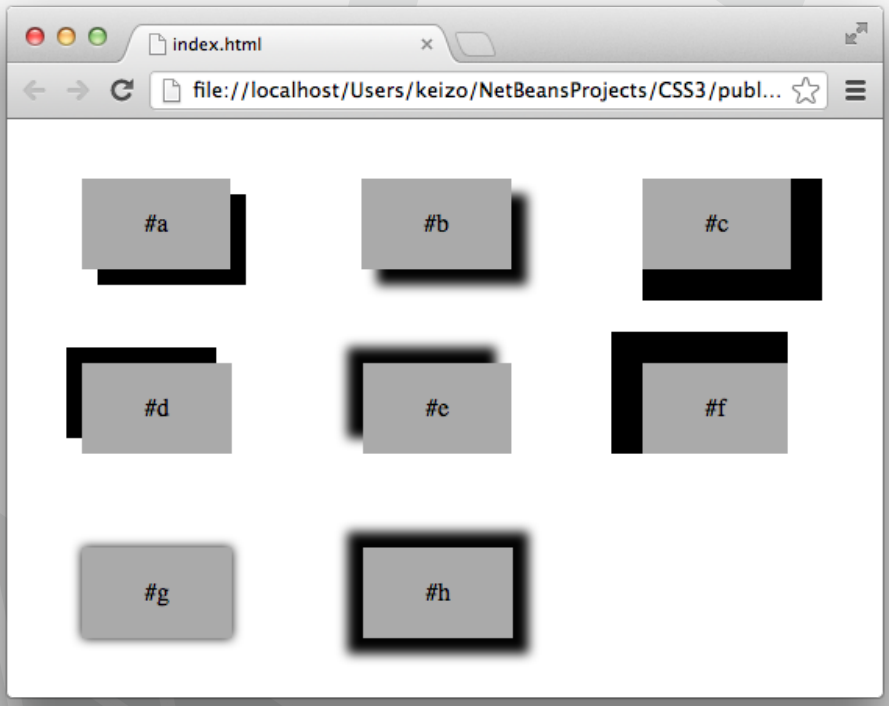

Figura D.3: Aplicações diferentes da propriedade box-shadow

As regras para a aplicação da sombra interna são as mesmas da sombra externa. A única diferença é que devemos começar a atribuição do valor com a palavra `inset`.

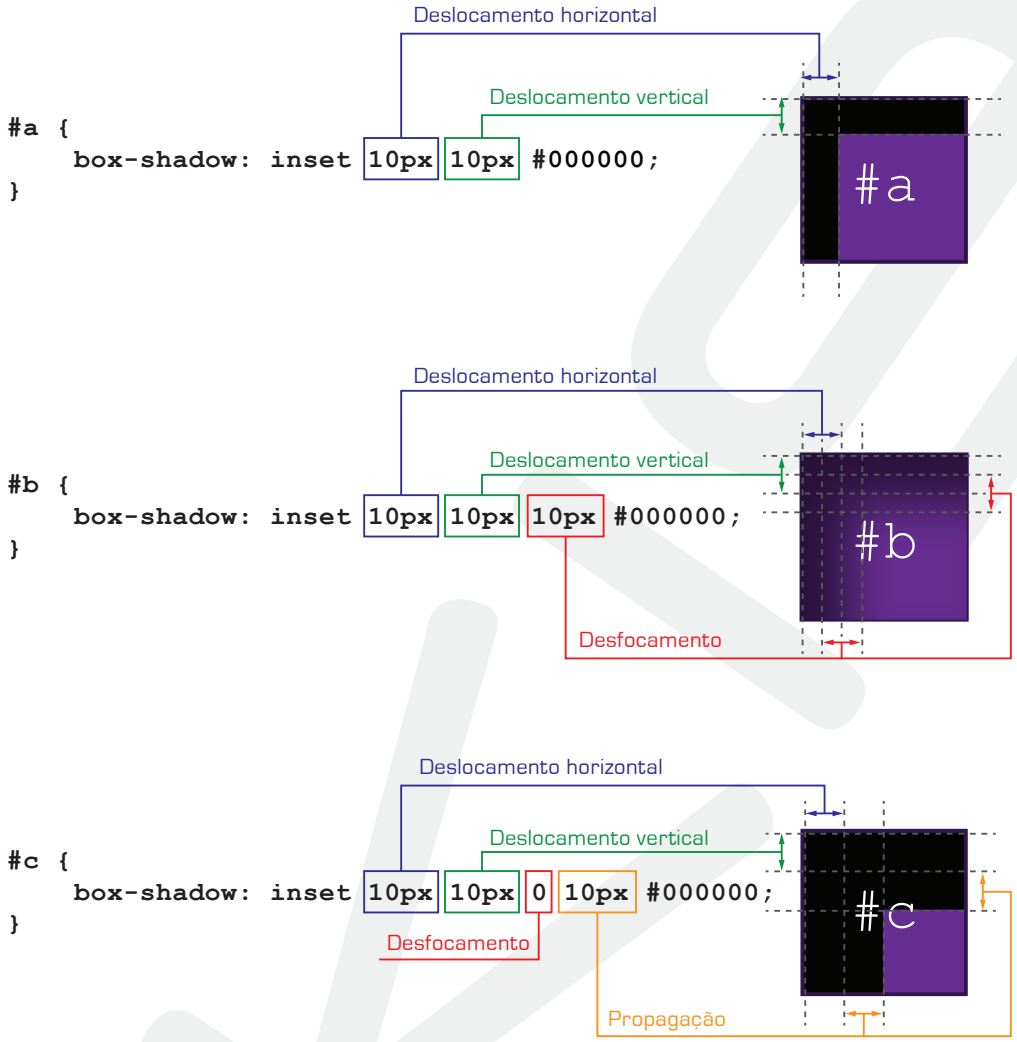

Figura D.4: Funcionamento da propriedade box-shadow para sombra interna

Observe no código CSS abaixo valores aplicados na propriedade `box-shadow` para criar sombras internas.

```
1  #a {
2    box-shadow: inset 10px 10px #000000;
3  }
4
5  #b {
6    box-shadow: inset 10px 10px 10px #000000;
7  }
8
9  #c {
10   box-shadow: inset 10px 10px 0 10px #000000;
11 }
12
13 #d {
14   box-shadow: inset -10px -10px #000000;
15 }
16
17 #e {
18   box-shadow: inset -10px -10px 10px #000000;
```

```
19 }
20
21 #f {
22   box-shadow: inset -10px -10px 0 10px #000000;
23 }
24
25 #g {
26   box-shadow: inset 0 0 10px #000000;
27 }
28
29 #h {
30   box-shadow: inset 0 0 10px 10px #000000;
31 }
```

Código CSS D.4: Exemplos de uso da propriedade box-shadow para sombra interna

Em um navegador, o resultado da aplicação das regras CSS acima seria:

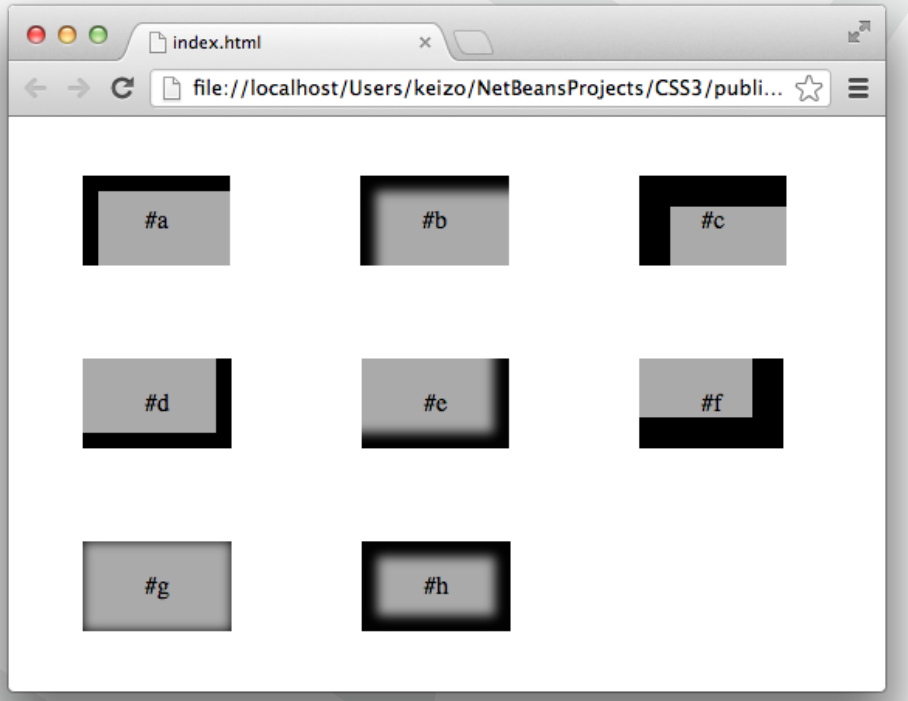

Figura D.5: Aplicações diferentes da propriedade box-shadow para sombras internas

text-shadow

Para aplicarmos uma sombra no texto de um elemento devemos utilizar a propriedade `text-shadow`. O seu funcionamento é muito semelhante ao da propriedade `box-shadow`, exceto pelo fato que não podemos criar uma sombra interna e não existe o parâmetro de propagação da sombra.

```
1 #a {
2   text-shadow: 10px 10px #ff0000;
3 }
4
```

```
5  #b {
6    text-shadow: 10px 10px 10px #ff0000;
7  }
8
9  #c {
10   text-shadow: -10px -10px #ff0000;
11 }
12
13 #d {
14   text-shadow: -10px -10px 10px #ff0000;
15 }
16
17 #e {
18   text-shadow: 0 0 10px #ff0000;
19 }
```

Código CSS D.5: Exemplos de uso da propriedade text-shadow para sombra de texto

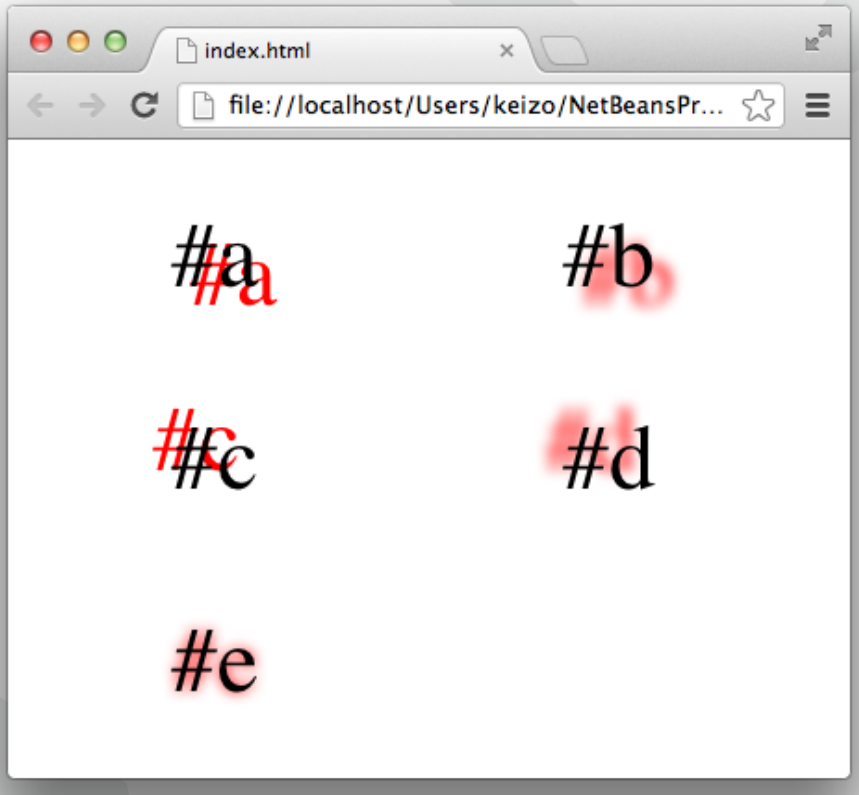

Figura D.6: Aplicações diferentes da propriedade text-shadow para sombra de texto

Mais Sobre

As propriedades `box-shadow` e `text-shadow` podem receber valores para definir mais de uma sombra. Para isso devemos separar os valores por vírgula. Observe o exemplo:

```
box-shadow:  10px 10px 10px #000000, -10px -10px 10px #ff0000
```

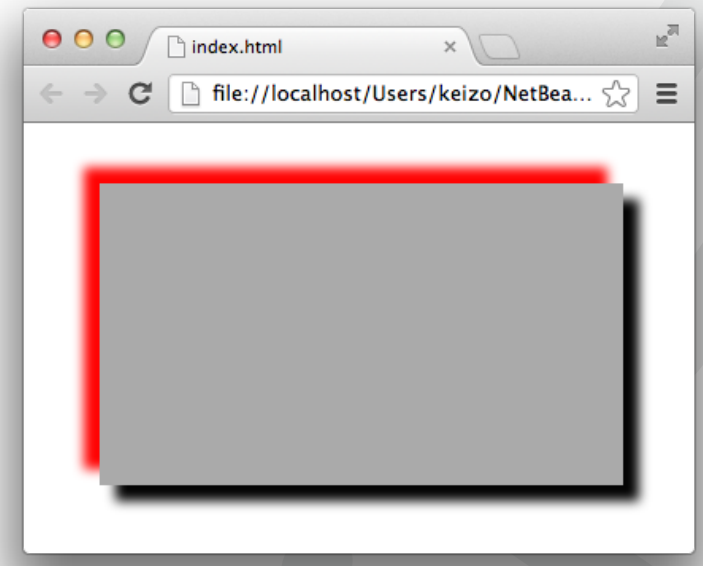

Figura D.7: Múltiplas sombras na propriedade box-shadow

Transformações

No CSS3 foram introduzidas algumas funções para realizar as transformações de translado, escalonamento, distorção e rotação nos elementos de uma página HTML. Essas funções são utilizadas em conjunto com a propriedade `transform` do CSS3.

translate()

O resultado da aplicação da função `translate(m, n)` é semelhante ao resultado obtido ao mover um elemento através do atributo `position` com o valor `relative`. Ao utilizar a função `translate(m, n)` um elemento é transladado a `m` unidades de medida da esquerda e `n` unidades de medida do topo.

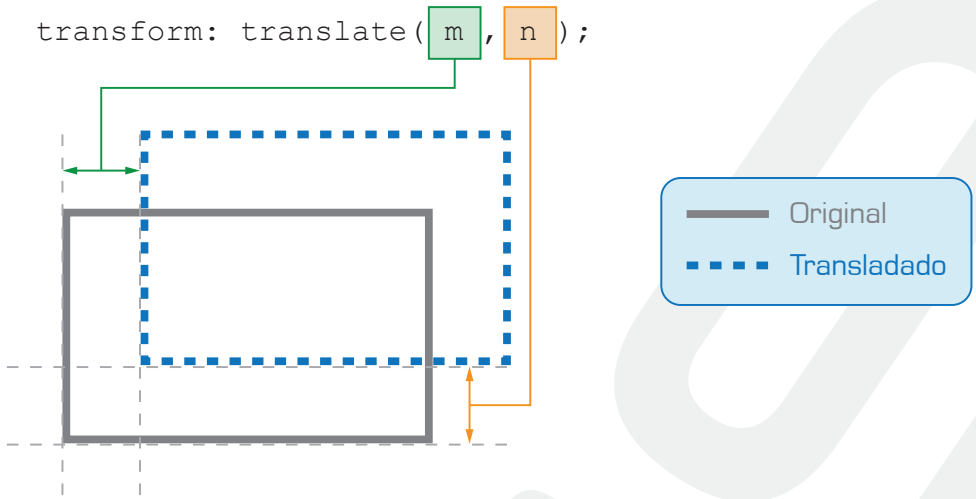

Figura D.8: Função translate()

scale()

A função `scale(m, n)` escalonará as dimensões de um elemento. O escalonamento será aplicado sobre a largura a uma taxa definida por **m** e sobre a altura a uma taxa definida por **n**. Caso a função seja chamada com apenas um parâmetro, a mesma taxa será aplicada na altura e largura do elemento.

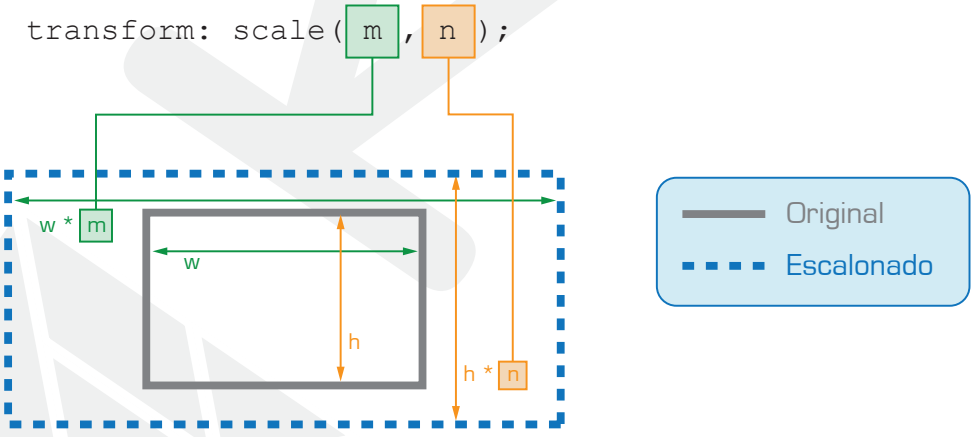

Figura D.9: Função scale()

rotate()

A função `rotate(m)` rotacionará um elemento em torno do seu ponto de origem. O valor de `m` deve ser dado em graus, voltas ou grado.

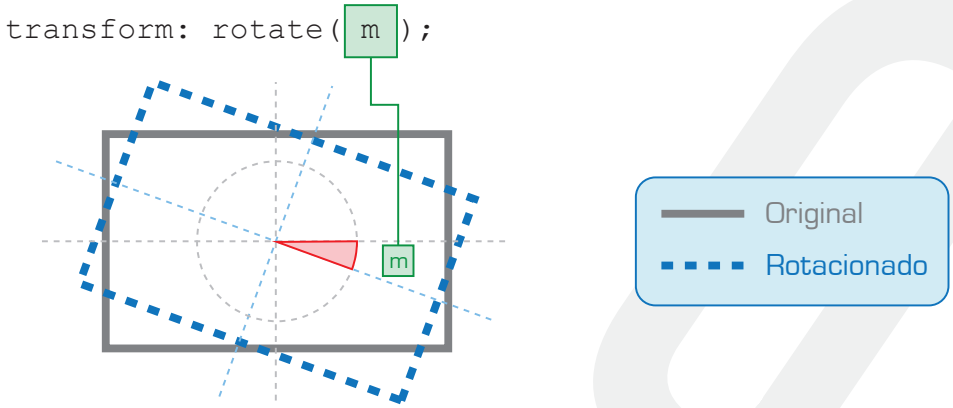

Figura D.10: Função rotate()

skew()

A função `skew(m, n)` distorcerá um elemento em relação aos eixos x e y. `m` e `n` definem as distorções aplicadas nos eixos x e y respectivamente. Os valores de `m` e `n` devem ser dados em graus, voltas ou grado.

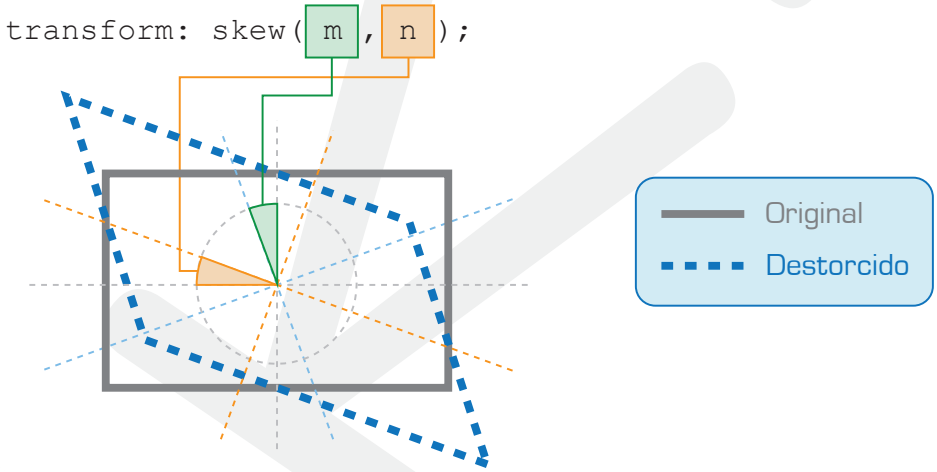

Figura D.11: Função skew()

Fontes no CSS3

Antes do CSS3 os webdesigners estavam limitados a utilizarem apenas as fontes disponíveis nos computadores dos usuários. Quando era necessário utilizar alguma fonte diferente o webdesigner precisava recorrer a alguma técnica como Cufón (Custom Font) ou sIFR (Scalable Inman Flash Replacement).

Durante muito tempo essas técnicas foram amplamente utilizadas por trazerem uma liberdade muito maior para os designers no momento da elaboração da parte artística de uma página. Porém, essa flexibilidade tinha o seu preço: exigiam um pouco mais de processamento e/ou o uso de plugins de terceiros no momento da renderização da página. Além disso, tais técnicas não apresentavam

resultados muito satisfatórios quando era necessário exibir uma grande quantidade de texto com a fonte escolhida.

@font-face

A regra `@font-face` nos permite ativar automaticamente uma fonte quando ela necessária. A fonte pode estar um arquivo local (máquina do usuário) ou remoto (servidor). Veja o exemplo abaixo.

```
1  @font-face {
2    font-family: 'NomeDaFonte';
3    src: url('nome-da-fonte.eot');
4    src: url('nome-da-fonte.eot?#iefix') format('embedded-opentype'),
5       url('nome-da-fonte.woff') format('woff'),
6       url('nome-da-fonte.ttf') format('truetype'),
7       url('nome-da-fonte.svg#webfontregular') format('svg');
8  }
```

Código CSS D.6: font-face.css

Os formatos de fontes para os quais os navegadores oferecem suporte variam de navegador para navegador. Por isso, o código do exemplo acima possui diversas declarações para a mesma fonte. Cada declaração é válida para um grupo diferente de navegadores.

- Linhas 3 à 4: Internet Explorer 6 9. `?#iefix` é um *Hack CSS* para que no Internet Explorer 6 o restante do valor atribuído à propriedade `src` seja ignorado. Além disso, o charactere # desse Hack CSS evita que falhas devido ao excesso de caracteres possam ocorrer.
- Linha 5: formato utilizado por todas as versões dos navegadores mais modernos. É o formato recomendado pelo W3C.
- Linha 6: formato aceito nos navegadores Firefox, Chrome, Safari e Opera. Usado também nos navegadores padrão dos sistemas Android e iOS.
- Linha 7: formato aceito no navegador padrão dos sistemas iOS mais antigos.

A função `url()` deve ser utilizada quando queremos utilizar uma fonte que está armazenada remotamente (no servidor). Caso queiramos carregar uma fonte disponível na máquina do usuário, devemos utilizar a função `local()`, passando como parâmetro o nome da fonte desejada.

```
1  @font-face {
2    font-family: 'NomeDaFonte';
3    src: local('nome-da-fonte');
4  }
```

Código CSS D.7: font-face-local.css

Para utilizarmos as fontes definidas na regra `@font-face` devemos utilizar a propriedade `font-family` dentro de uma regra CSS.

```
1  @font-face {
2    font-family: 'MinhaFonteBacana';
3    src: url('minha-fonte-bacana.eot');
4    src: url('minha-fonte-bacana.eot?#iefix') format('embedded-opentype'),
5       url('minha-fonte-bacana.woff') format('woff'),
6       url('minha-fonte-bacana.ttf') format('truetype'),
7       url('minha-fonte-bacana.svg#webfontregular') format('svg');
8  }
9
10 p {
```

```
11    font-family: 'MinhaFonteBacana';
12  }
```

Código CSS D.8: font-family.css

No exemplo acima todos os elementos **p** de uma página utilizarão a fonte **MinhaFonteBacana**.

MAIS PROPRIEDADES CSS

APÊNDICE E

Propriedades de tabela

border-spacing

A propriedade `border-spacing` define a distância entre as bordas das células de uma tabela, assim como a distância entre a borda de uma célula e a borda da própria tabela. Veja o exemplo abaixo.

```
1 table {
2   border-collapse: separate;
3   border-spacing: 10px;
4 }
```

Código CSS E.1: border-spacing

Também podemos definir valores diferentes para o espaçamento horizontal e vertical, como podemos verificar no exemlo a seguir.

```
1 table {
2   border-collapse: separate;
3   border-spacing: 10px 20px;
4 }
```

Código CSS E.2: border-spacing

caption-side

A propriedade `caption-side` define em que posição a legenda de uma tabela deverá aparecer. Por padrão, a propriedade `caption-side` possui o valor `top`, fazendo com que a legenda fique na parte superior da tabela. O valor `bottom` faz com que a legenda fique na parte inferior da tabela. Veja o exemplo abaixo.

```
1 caption {
2   caption-side: bottom;
3 }
```

Código CSS E.3: caption-side

empty-cells

A propriedade `empty-cells` define se uma célula vazia da tabela deve ser exibida ou não. Por padrão, a propriedade `empty-cells` possui o valor `show`. Para que uma célula vazia não seja exibida devemos atribuir o valor `hide`. Veja o exemplo abaixo.

```
...
<table>
  <tr>
    <th>Nome</th>
    <th>E-mail</th>
  </tr>
  <tr>
    <td>Jonas</td>
    <td>jonas@k19.com.br</td>
  </tr>
  <tr>
    <td>Jundy</td>
    <td></td>
  </tr>
</table>
...
```

Código HTML E.1: empty-cells.html

```
table {
  border: 1px solid black;
  border-collapse:separate;
  empty-cells:hide;
}

table th, table td {
  border: 1px solid black;
}
```

Código CSS E.4: empty-cells.css

Propriedades de borda

As bordas de um elemento podem ser definidas de maneira simplificada através da propriedade `border`. Porém, se desejarmos podemos utilizar as variantes da propriedade `border` para definir cada característica das bordas. Veja o exemplo abaixo.

```
div {
  border: 1px solid red;
}

p {
  border-width: 1px;
  border-style: solid;
  border-color: red;
}
```

Código CSS E.5: Propriedade border e suas variantes

Ao utilizarmos as variantes podemos definir valores diferentes para cada um dos lados do elemento. Veja o exemplo a seguir.

```
p {
  border-width: 1px 2px 3px 4px;
  border-style: solid dotted double dashed;
  border-color: red blue green yellow;
}
```

Código CSS E.6: Variantes da propriedade border com valores diferentes para cada lado do elemento

No exemplo acima, para cada propriedade os valores são atribuídos respectivamente às bordas superior, inferior, esquerda e direita. O mesmo resultado pode ser obtido utilizando as propriedades individuais de cada borda. Confira a lista das propriedades individuais.

- border-top
- border-top-color
- border-top-style
- border-top-width
- border-right
- border-right-color
- border-right-style
- border-right-width
- border-bottom
- border-bottom-color
- border-bottom-style
- border-bottom-width
- border-left
- border-left-color
- border-left-style
- border-left-width

Propriedades de conteúdo gerado

content

A propriedade `content` deve ser utilizada em conjunto com os pseudo-elementos `:before` ou `:after` para inserir um conteúdo qualquer antes ou depois do conteúdo do elemento. Podemos utilizar a propriedade da seguinte maneira:

```
1 ...
2 <a href="http://www.k19.com.br">K19 Treinamentos</a>
3 ...
```

Código HTML E.2: content.html

```
1 a:after
2 {
3   content: " (clique para conhecer a empresa)";
4 }
```

Código CSS E.7: content.css

No exemplo acima, o texto do link após a renderização da página seria: **K19 Treinamentos (clique para conhecer a empresa)**.

counter-increment e counter-reset

A propriedade `counter-increment` incrementa um ou mais valores de um contador. Essa propriedade normalmente trabalha em conjunto com as propriedades `counter-reset` e `content`. Veja o exemplo abaixo:

```
1  body {
2    counter-reset:secao;
3  }
4
5  h1 {
6    counter-reset: subsecao;
7  }
8
9  h2:before {
10   counter-increment: subsecao;
11   content: counter(secao) "." counter(subsecao) " ";
12 }
```

Código CSS E.8: counter-increment

Repare que no exemplo acima a função `counter()` do CSS foi utilizada para obtermos os valores dos contadores `secao` e `subsecao`. Além disso, observe que temos uma regra para o elemento `h1` que zera o contador `subsecao` através da propriedade `counter-reset`. Portanto, toda vez que um elemento `h1` aparecer no código HTML o contador `subsecao` será zerado.

Propriedades de posicionamento

float

A propriedade `float` faz com que um elemento tenha um posicionamento flutuante em relação ao conteúdo ao seu redor. Essa propriedade pode assumir os valroes right, left e none. Veja o exemplo abaixo.

```
1  img  {
2    float:right;
3  }
```

Código CSS E.9: float

No exemplo acima, qualquer elemento que for definido no código HTML após um elemento `img` será posicionado à esquerda da imagem. Isso se deve ao fato da imagem estar "flutuando" à direita do conteúdo.

clear

O elemento com a propriedade `clear` faz com que os elementos com a propriedade `float` definidos anteriormente a ele no código HTML percam o comportamento de elementos "flutuantes". Se a propriedade assumir o valor `left`, apenas os elementos com `float: left` perderão o comportamento "flutuante". Se a propriedade assumir o valor `right`, apenas os elementos com `float: right` perderão o comportamento "flutuante".

Além dos valores `left` e `right`, a propriedade `clear` pode assumir o valor `both`, o que fará com que os elementos anterirormente definidos com a propriedade `float: left` e/ou `float: right`

percam o comportamento “flutuante”.

A propriedade `clear` aceita e tem como padrão o valor `none`, que não afeta o comportamento “flutuante” dos elementos anteriormente definidos.

```
...
<img src="imagem.jpg" />
<p>Texto de exemplo</p>
<p class="clear">Outro texto</p>
...
```

Código HTML E.3: clear.html

```
img {
  float:right;
}

p.clear {
  clear:both;
}
```

Código CSS E.10: clear.css

cursor

A propriedade `cursor`, define o tipo do cursor que será mostrado quando se passa o mouse por cima de um elemento. A propriedade pode assumir os seguintes valores: auto, crosshair, default, e-resize, help, move, n-resize, ne-resize, nw-resize, pointer, progress, s-resize, se-resize, sw-resize, text, w-resize e wait. Veja o exemplo abaixo.

```
div {
  cursor: text;
}
```

Código CSS E.11: cursor

overflow

A propriedade `overflow` define o que deve acontecer quando o conteúdo for maior que o limite da caixa do elemento que o contém. Os valores que a propriedade pode assumir são:

- visible (padrão): o conteúdo do elemento é exibido mesmo que ele ultrapasse os limites do elemento no qual está contido.
- hidden: a parte excedente do conteúdo não ficará visível.
- scroll: no elemento aparecerão barras de rolagem.
- auto: no elemento aparecerão barras de rolagem para que seja possível ver todo o conteúdo, caso o conteúdo seja maior que o elemento.

Veja o exemplo abaixo:

```
div {
  width:150px;
  height:150px;
  overflow:scroll;
}
```

Código CSS E.12: overflow

visibility

A propriedade `visibility` define se um elemento está visível ou não, podendo assumir os valores `visible`, `hidden` ou `collapse`, sendo o último válido apenas para elementos de tabela.

Ao utilizar a propriedade `visibility` com o valor `hidden`, repare que apesar do elemento não ser exibido seu espaço continua sendo ocupado.

```
1  h2 {
2    visibility:hidden;
3  }
```

Código CSS E.13: visibility

z-index

A propriedade `z-index` especifica a ordem de empilhamento de um elemento. O elemento com um valor maior de empilhamento, sempre estará na frente de um elemento com menor valor. Essa propriedade somente é válida para elementos posicionados através da propriedade `position` com os valores `absolute`, `relative` ou `fixed`. Veja o exemplo abaixo.

```
1  img {
2    position:absolute;
3    left:0px;
4    top:0px;
5    z-index: 1;
6  }
7
8  img.layer1 {
9    top: 2px;
10   left: 2px;
11   z-index: 2;
12 }
```

Código CSS E.14: z-index

QUIZZES

APÊNDICE F

Quiz 1

Considere uma página HTML contendo um com largura (width) 200px, margem interna (padding) 10px e borda de 3px. Visualmente, qual é o espaço horizontal ocupado por esse elemento?

a) 200px

b) 203px

c) 210px

d) 213px

e) 226px

De acordo com o Box Model do CSS visto no Capítulo 3, ao atribuirmos margens internas e bordas em um elemento com largura definida fazemos com que, visualmente, a largura ocupada por esse elemento seja a soma das propriedades **width**, **padding-left**, **padding-right**, **border-left** e **border-right**. Como utilizamos a propriedade **padding** com o valor 10px para definir as margens internas, podemos considerar que temos 10px nas propriedades **padding-left** e **padding-right**. A mesma idéia se aplica à propriedade **border**, portanto temos **border-left** e **border-right** com 3px cada.

Fazendo a soma temos: 200px (width) + 10px (padding-left) + 10px (padding-right) + 3px (border-left) + 3px (border-right) = **226px**

Portanto o espaço horizontal visualmente ocupado pelo elemento é de **226px**.

RESPOSTAS

APÊNDICE G

Resposta do Exercise 2.1

```
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exibe o nome do curso duas vezes</title>
  </head>
  <body>
    <p>K02 - Desenvolvimento Web com HTML, CSS e JavaScript.</p>
    <p>K02 - Desenvolvimento Web com HTML, CSS e JavaScript.</p>
  </body>
</html>
```

Código HTML 2.3: imprime-2x-curso.html

Resposta do Exercise 2.2

```
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title> Exercício de quebra de linha forcada.</title>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Nam tempor ante non sem euismod at pharetra leo bibendum.
    Nunc quis scelerisque risus. Fusce tristique tortor sit<br />
    amet metus mattis vitae imperdiet arcu porta. Maecenas
    tempor placerat est non tincidunt.<br />
    In dignissim adipiscing iaculis.</p>
  </body>
</html>
```

Código HTML 2.8: quebra-de-linha-forcada.html

Observe a utilização da tag **
** *para quebrar a linha.*

Resposta do Exercise 2.3

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Curiosidades do Mundo</title>
  </head>
```

```
  <body>
    <h1>Curiosidades do Mundo</h1>

    <h2>1. Europa</h2>
    <p>A Europa é o segundo menor continente em superfície do mundo, cobrindo
    cerca de 10 180 000 quilômetros quadrados ou 2% da superfície da Terra
    e cerca de 6,8% da área acima do nível do mar.</p>

    <h3>1.1 Alemanha</h3>
    <p>Com 81,8 milhões de habitantes em janeiro de 2010, o país tem a maior
    população entre os Estados membros da União Europeia e é também o lar da
    terceira maior população de migrantes internacionais em todo o mundo.</p>

    <h4>1.1.1 Hesse</h4>
    <p>A capital é Wiesbaden e a maior cidade Francoforte do Meno (Frankfurt am Main),
    onde está localizado um dos maiores aeroportos do mundo e um centro financeiro
    de grande importância.</p>

    <h5>1.1.1.1 Frankfurt</h5>
    <p>Frankfurt am Main ou Francoforte do Meno, mais conhecida simplesmente como
    Frankfurt, é a maior cidade do estado alemão de Hesse e a quinta maior cidade
    da Alemanha, com uma população 700.000 habitantes em 2012.</p>

    <h3>1.2 França</h3>
    <p>É o país mais visitado no mundo, recebendo 82 milhões de turistas estrangeiros
    por ano.</p>

    <h4>1.2.1 Île-de-France</h4>
    <p>Ilha de França (em francês: Île-de-France) é uma das 26 regiões administrativas
    da França.</p>

    <h5>1.2.1.1 Paris</h5>
    <p>Paris é a capital e a mais populosa cidade da França, bem como a capital da
    região administrativa de Île-de-France.</p>

    <h2>2. Ásia</h2>
    <p>A Ásia é o maior dos continentes, tanto em área como em população.</p>

    <h3>2.1 Japão</h3>
    <p>O país é um arquipélago de 6 852 ilhas, cujas quatro maiores são Honshu,
    Hokkaido, Kyushu e Shikoku, representando em conjunto 97% da área
    terrestre nacional.</p>

    <h4>2.1.1 Okinawa</h4>
    <p>Antigamente, Okinawa fazia parte de um reino independente, o reino Ryukyu,
    o que foi decisivo para o desenvolvimento de uma cultura própria do desenrolar
    de uma história particular e significativamente diferenciada do resto do
    Japão.</p>

    <h5>2.1.1.1 Nago</h5>
    <p>De 21 de julho até 23 de julho de 2000, foi sede do encontro anual do G8.</p>

    <p><small>Fonte: wikipedia.org</small></p>
  </body>
</html>
```

Código HTML 2.12: geografia.html

Resposta do Exercise 2.4

```
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title> Exercício produto K19 </title>
```

```
  </head>
  <body>
    <h1> Cafeteira - K19 </h1>
    <h2> Gosto Doce </h2>

    <h3>Recursos</h3>
    <p>Prático, uso de capsulas</p>
    <p>Dois tipos: quente e frio</p>
    <p>Faz seu expresso em poucos segundos</p>
    <p>Mais de 10 sabores</p>
    <p>Silencioso</p>

    <h3>Observações</h3>
    <p>Verificar a quantidade de água no reservatório</p>
    <p>Verificar o led para a inicialização</p>
    <p>Desligar aparelho após uso</p>
    <p>Sempre limpar corretamente o suporte para a capsula</p>

    <h4>Itens relacionados</h4>
    <p>Capuccino - 16 capsulas</p>
    <p>Expresso - 16 capsulas </p>
    <p>Expresso Intenso - 8 capsulas</p>
    <p>Caneca K19 - 300ml</p>
  </body>
</html>
```

Código HTML 2.13: produto.html

Resposta do Exercise 2.5

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title> Exercício com links e target</title>
  </head>
  <body>
    <p><a href="http://www.amazon.com/" target="_blank">Site Amazon</a></p>
    <p><a href="http://www.google.com.br/" target="_top">Site Google</a></p>
    <p><a href="https://www.facebook.com/" target="_self">Site FaceBook</a></p>
    <p><a href="https://www.orkut.com/" target="_parent">Site Orkut</a></p>
  </body>
</html>
```

Código HTML 2.18: links-target.html

Resposta do Exercise 2.6

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exercícios com iframe - p1</title>
  </head>
  <body>
    <p><a href="http://www.k19.com.br">K19</a></p>
    <div>
      <iframe src="iframe-p2.html"></iframe>
    </div>
  </body>
```

```
12 </html>
```

Código HTML 2.19: iframe-p1.html

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exercícios com iframe - p2</title>
5    </head>
6    <body>
7      <p><a href="http://www.k19.com.br" target="_parent">K19</a></p>
8      <div>
9        <iframe src="iframe-p3.html"></iframe>
10     </div>
11   </body>
12 </html>
```

Código HTML 2.20: iframe-p2.html

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exercícios com iframe - p3</title>
5    </head>
6    <body>
7      <p><a href="http://www.k19.com.br/">K19</a></p>
8    </body>
9  </html>
```

Código HTML 2.21: iframe-p3.html

Resposta do Exercise 2.7

```
1  <html>
2    <head>
3    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exercício de âncora</title>
5    </head>
6    <body>
7      <h1>Documento 1</h1>
8
9      <p><a href="http://www.k19.com.br">Site K19</a></p>
10
11     <p> ... </p>
12     <p> ... </p>
13     <a name="ancora"></a>
14     <p>Âncora</p>
15     <p> ... </p>
16   </body>
17 </html>
```

Código HTML 2.26: exercicio-ancora.html

```
1  <html>
2    <head>
3    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Exercício de âncora 2</title>
5    </head>
6    <body>
7      <h1>Documento 2</h1>
```

```
8
9       <p> ... </p>
10      <p> ... </p>
11      <p> ... </p>
12
13      <p><a href="http://www.google.com" target="_blank">Site do Google</a></p>
14
15      <a href="exercicio-ancora.html#ancora">Ir para o documento 1</a>
16    </body>
17  </html>
```

Código HTML 2.27: exercicio-ancora-2.html

Resposta do Exercise 2.8

```
1   <html>
2     <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício de links</title>
5     </head>
6     <body>
7       <h1 id="title-info">Informações sobre a Europa</h1>
8
9       <p>
10        <a href="http://pt.wikipedia.org/wiki/Europa#Geografia" target="_blank">
11          Geografia da Europa
12        </a>
13      </p>
14
15      <p>
16        <a href="http://pt.wikipedia.org/wiki/Europa#Economia" target="_blank">
17          Economia da Europa
18        </a>
19      </p>
20
21      <p> ... </p>
22      <p> ... </p>
23      <p> ... </p>
24
25      <p><a href="exercicio-ancora-3.html#title-info">Ir para o topo da página</a></p>
26    </body>
27  </html>
```

Código HTML 2.28: exercicio-ancora-3.html

Resposta do Exercise 2.9

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício - Imagens</title>
5     </head>
6     <body>
7       <h1>Imagens:</h1>
8       <p>
9         Apostilas K19
10        <img src="http://www.k19.com.br/app/webroot/css/img/books-entry-point.png" />
11      </p>
```

```
12      <p>
13        Logo K19
14        <img src="http://www.k19.com.br/app/webroot/css/img/main-header-logo.png" />
15      </p>
16    </body>
17  </html>
```

Código HTML 2.31: exercicio-imagens.html

Resposta do Exercise 2.10

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type"content="text/html;charset=UTF-8">
4       <title>Exercício - Tabela</title>
5       <style type="text/css">
6       table, th, td {
7         border: 1px solid red;
8       }
9       </style>
10    </head>
11    <body>
12      <table>
13        <thead>
14          <tr>
15            <th>Continente/Subcontinente</th>
16            <th>Cidade</th>
17            <th>Idioma</th>
18          </tr>
19        </thead>
20        <tfoot>
21          <tr>
22            <td colspan="3">Última atualização: 11/2012</td>
23          </tr>
24        </tfoot>
25        <tbody>
26          <tr>
27            <td rowspan="2">América do Sul</td>
28            <td>São Paulo</td>
29            <td>Português</td>
30          </tr>
31          <tr>
32            <td>Cidade do México</td>
33            <td>Espanhol</td>
34          </tr>
35          <tr>
36            <td rowspan="3">Ásia</td>
37            <td>Tóquio</td>
38            <td>Japonês</td>
39          </tr>
40          <tr>
41            <td>Xangai</td>
42            <td>Mandarim</td>
43          </tr>
44          <tr>
45            <td>Nova Délhi</td>
46            <td>Hindi</td>
47          </tr>
48          <tr>
49            <td>América do Norte</td>
50            <td>Nova Iorque</td>
51            <td>Inglês</td>
52          </tr>
53
```

```
54        </tbody>
55      </table>
56    </body>
57  </html>
```

Código HTML 2.35: exercicio-tabela.html

Resposta do Exercise 2.11

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício Complementar - Pontos Turísticos</title>
5     </head>
6     <body>
7       <h1>Pontos Turísticos do Brasil</h1>
8       <h2>Lista dos pontos</h2>
9
10      <dl>
11        <dt>Ilha Bela - SP</dt>
12        <dd>
13          Praias, Trilhas e Mergulho em Náufrago.
14        </dd>
15        <dt>Bonito - MS</dt>
16        <dd>
17          Mergulho em rios de águas transparentes, cachoeiras, grutas e cavernas.
18        </dd>
19        <dt>Museu de Arte de São Paulo - SP</dt>
20        <dd>
21          Grande acervo com diversas obras de artistas consagrados.
22        </dd>
23      </dl>
24    </body>
25  </html>
```

Código HTML 2.38: exercicio-pontos-turisticos.html

Resposta do Exercise 2.12

```
1   <html>
2     <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4       <title>Exercício Complementar - Como instalar o seu XPTO?</title>
5     </head>
6     <body>
7       <h1>Como instalar o seu XPTO? - K19 Eletronics</h1>
8
9       <ol>
10        <li>Verifique se todos os acessórios listados <a href="#">nesta página</a>
11        estão presentes.</li>
12        <li>Coloque o aparelho na horizontal sobre uma superfície estável.</li>
13        <li>Conectar o aparelho ao computador ou notebook utilizando o cabo USB.</li>
14        <li>Usar o CD-ROM para instalação do software.</li>
15        <li>Conectar o aparelho à fonte de energia com o adaptador AC.</li>
16        <li>Ligar o aparelho e esperar o reconhecimento no computador.</li>
17      </ol>
18    </body>
19  </html>
```

Código HTML 2.41: exercicio-manual-passoapasso.html

Resposta do Exercise 2.13

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exercício - Formação Desenvolvedor Java K19</title>
  </head>
  <body>
    <dl>
      <h1>K10 - Formação Desenvolvedor Java</h1>

    <ul>
      <li>K11 - Orientação a Objetos em Java</li>
      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
    </ul>
    </dl>
  </body>
</html>
```

Código HTML 2.44: exercicio-formacao-k19.html

Resposta do Exercise 2.14

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exercício - Formulário</title>
  </head>
  <body>
    <form action="teste.html" method="get">
      <input type="hidden" name="hash" value="4w587nt3" />

      <div>
        <label for="nome">Nome: </label>
        <input type="text" id="nome" name="nome" />
      </div>

      <div>
        <label for="sobrenome"> Sobrenome: </label>
        <input type="text" id="sobrenome" name="sobrenome" />
      </div>

      <div>
        <label for="senha">Senha:</label>
        <input type="password" id="senha" name="senha" />
      </div>

      <div>
        <label for="foto">Escolha uma foto:</label>
        <input type="file" id="foto" name="foto" />
      </div>

      <div>
        Sexo:
```

```
32          <input type="radio" name="sexo" id="sexo-m" />
33          <label for="sexo-m">Masculino</label>
34          <input type="radio" name="sexo" id="sexo-f" />
35          <label for="sexo-f">Feminino</label>
36        </div>
37
38        <div>
39          <label for="estado">Selecione um Estado:</label>
40          <select id="estado" name="estado">
41            <option value="sp">São Paulo</option>
42            <option value="rj">Rio de Janeiro</option>
43            <option value="mg">Minas Gerais</option>
44            <option value="es">Espírito Santo</option>
45            <option value="pr">Paraná</option>
46          </select>
47        </div>
48
49        <div>
50          Escolha um ou mais meios de comunicação:
51          <input type="checkbox" name="comunicacao[]" id="comunicacao-musica" />
52          <label for="comunicacao-musica">Música</label>
53          <input type="checkbox" name="comunicacao[]" id="comunicacao-televisao" />
54          <label for="comunicacao-televisao">Televisão</label>
55          <input type="checkbox" name="comunicacao[]" id="comunicacao-radio" />
56          <label for="comunicacao-radio">Rádio</label>
57        </div>
58
59        <div>
60          <label for="obs">Observações:</label>
61          <textarea id="obs" name="observacoes"></textarea>
62        </div>
63
64        <div>
65          <input type="submit" value="Enviar" />
66          <input type="reset" value="Desfazer alterações" />
67        </div>
68      </form>
69    </body>
70  </html>
```

Código HTML 2.51: exercicio-formulario.html

Resposta do Exercise 4.1

```
1  <html>
2    <head>
3      <title> Imprime numeros de 1 ate 50 - 2x </title>
4      <script type="text/javascript" src="imprime-numeros-1-50-2x.js"></script>
5    </head>
6    <body>
7    </body>
8  </html>
```

Código HTML 4.6: imprime-numeros-1-50-2x.html

```
1  for (var x = 0; x < 2; x++) {
2    for (var y = 1; y <= 50; y++) {
3      document.writeln(y);
4      document.writeln('<br/>');
5    }
6  }
```

Código Javascript 4.15: imprime-numeros-1-50-2x.js

Resposta do Exercise 4.2

```
<html>
  <head>
    <title> Imprime formacao e curso </title>
    <script type="text/javascript" src="imprime-formacao-curso.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.7: imprime-formacao-curso.html

```
for (var x = 0; x < 5; x++) {
  document.writeln('K00 - Formação Básica');
  document.writeln('<br/>');

  for (var y = 0; y < 3; y++) {
    document.writeln('K02 - Desenvolvimento Web com HTML, CSS e JavaScript');
    document.writeln('<br/>');
  }
}
```

Código Javascript 4.16: imprime-formacao-curso.js

Resposta do Exercise 4.3

```
<html>
  <head>
    <title> Imprima *** para multiplos de 3 </title>
    <script type="text/javascript" src="imprime-***-multiplo3.js"></script>
  </head>
  <body>
  </body>
</html>
```

*Código HTML 4.8: imprime-***-multiplo3.html*

```
for (var x = 1; x <= 60; x++) {
  if (x % 3 != 0) {
    document.writeln ('*');
  } else {
    document.writeln ('***')
  }
  document.writeln ('<br/>');
}
```

*Código Javascript 4.17: imprime-***-multiplo3.js*

Resposta do Exercise 4.4

```
<html>
  <head>
    <title> Imprimir * entre os numeros multiplos de 4 e 7 </title>
    <script type="text/javascript" src="imprime-*-no-lugar-multiplo4e7.js"></script> <↩
        /head>
  <body>
  </body>
</html>
```

Código HTML 4.9: imprime--no-lugar-multiplo4e7.html*

```
for (var x = 1; x <= 80; x++) {
  var resto = x%4;
  var resto2 = x%7;

  if (resto == 0) {
    document.writeln("*");
  } else if (resto2 == 0) {
    document.writeln("*");
  } else {
    document.writeln(x);
  }
  document.writeln('<br/>');
}
```

Código Javascript 4.18: imprime--no-lugar-multiplo4e7.js*

Resposta do Exercise 4.5

```
<html>
  <head>
    <title>Imprime padrão 3</title>
    <script type="text/javascript" src="imprime-padrao-3.js"></script>
  <head>
  <body>
  </body>
</html>
```

Código HTML 4.10: imprime-padrao-3.html

```
var linha = '*';
for(var contador = 1; contador <= 10; contador++) {
  document.writeln(linha);
  document.writeln('<br />');
  linha += '*';
}
```

Código Javascript 4.19: imprime-padrao-3.js

Resposta do Exercise 4.6

```
<html>
  <head>
    <title>Imprime padrão 4</title>
    <script type="text/javascript" src="imprime-padrao-4.js"></script>
```

```
5     </head>
6     <body>
7     </body>
8   </html>
```

Código HTML 4.11: imprime-padrao-4.html

```
1   var linha = '*';
2   for(var contador = 1; contador <= 8; contador++) {
3     document.writeln(linha);
4     document.writeln('<br />');
5     var resto = contador % 4;
6     if(resto == 0) {
7       linha = '*';
8     } else {
9       linha += '*';
10    }
11  }
```

Código Javascript 4.20: imprime-padrao-4.js

Resposta do Exercise 4.7

```
1   <html>
2     <head>
3       <title>Imprime padrão 5</title>
4       <script type="text/javascript" src="imprime-padrao-5.js"></script>
5     </head>
6     <body>
7     </body>
8   </html>
```

Código HTML 4.12: imprime-padrao-5.html

```
1   var penultimo = 0;
2   var ultimo = 1;
3
4   document.writeln(penultimo);
5   document.writeln('<br />');
6   document.writeln(ultimo);
7   document.writeln('<br />');
8
9   for(var contador = 0; contador < 28; contador++) {
10    var proximo = penultimo + ultimo;
11    document.writeln(proximo);
12    document.writeln('<br />');
13
14    penultimo = ultimo;
15    ultimo = proximo;
16  }
```

Código Javascript 4.21: imprime-padrao-5.js

Resposta do Exercise 4.8

```
<html>
  <head>
    <title> Embaralhando Array </title>
    <script type="text/javascript" src="embaralha-array.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.16: embaralha-array.html

```
var array = new Array(10);
for (var x = 0; x < array.length; x++) {
  array[x] = x;
}

for (var y = 0; y < array.length; y++) {
  document.writeln(array[y]);
  document.writeln('<br/>');
}

for(var z = 0; z < 15; z++) {
  var posicao1 = Math.floor(Math.random()*10);
  var posicao2 = Math.floor(Math.random()*10);
  var auxiliar = array[posicao1];

  array[posicao1] = array[posicao2];
  array[posicao2] = auxiliar;
}

document.writeln("------------------");
document.writeln('<br/>');

for(var w = 0; w < array.length; w++) {
  document.writeln(array[w]);
  document.writeln('<br/>');
}
```

Código Javascript 4.33: embaralha-array.js

Resposta do Exercise 4.9

```
<html>
  <head>
    <title> Ordenando Array </title>
    <script type="text/javascript" src="ordenando-array.js"></script>
  </head>
  <body>
  </body>
</html>
```

Código HTML 4.17: ordenando-array.html

```
var array = new Array(15);
for(var x = 0; x < array.length; x++) {
  array[x] = Math.floor(Math.random()*10);
}

for(var y = 0; y < array.length; y++) {
  document.writeln(array[y]);
}

```

```
10 document.writeln('<br/>');
11
12 array.sort();
13
14 document.writeln("----------------------------------------");
15 document.writeln('<br/>');
16
17 for(var z = 0; z < array.length; z++) {
18   document.writeln(array[z]);
19 }
```

Código Javascript 4.34: ordenando-array.js

Resposta do Exercise A.1

```
1 var formacao = {sigla: "K00", nome: "Formação Básica"};
2 console.log(formacao.sigla);
3 console.log(formacao.nome);
4
5 var formacao_2 = {sigla: "K10", nome: "Formação Desenvolvedor Java"};
6 console.log(formacao_2.sigla);
7 console.log(formacao_2.nome);
```

Código Javascript A.22: formacao-1

Resposta do Exercise A.2

```
1 var formacao = {sigla: "K00", nome: "Formação Básica"};
2 console.log(formacao.sigla);
3 console.log(formacao.curso);
4
5 var x = formacao;
6
7 x.sigla = "K00";
8 x.nome = "Formação Básica";
9
10 console.log(formacao.sigla);
11 console.log(formacao.nome);
```

Código Javascript A.23: formacao-2

Resposta do Exercise A.3

```
1 var formacao = {sigla: "K20", nome: "Formação Desenvolvedor Java Avançado"};
2
3 var nova_formacao = Object.create(curso);
4
5 nova_formacao.sigla = "K30";
6 nova_formacao.nome = "Formação Desenvolvedor .NET";
7
8 console.log(formacao.sigla);
```

```
9 console.log(formacao.nome);
```

Código Javascript A.24: prototipo

Resposta do Exercise A.4

```
1 var divisao = function(a, b) {
2   return a / b;
3 }
4
5 var resultado = divisao(10, 2);
6 console.log(resultado);
```

Código Javascript A.41: divisão()

Resposta do Exercise A.5

```
1  var conta = {
2    saldo: 0,
3    saque: function(valor) {
4      if (conta.saldo < 600){
5        throw {
6          message: "Saldo insuficiente."
7        }
8      } else if (valor <= 0) {
9        throw {
10         message: "Valores menores ou iguais a 0 não podem ser sacados."
11       }
12     } else {
13       this.saldo -= valor;
14     }
15     try {
16       conta.saque(0);
17     } catch(e) {
18       console.log(e.name);
19       console.log(e.message);
20     }
21   }
22 }
23
24 conta.saque(600)
25 console.log (conta.saque);
```

Código Javascript A.42: saque()

Resposta do Exercise A.6

```
1 var multiplicacao = function() {
2   var multiplicacao = 0;
3
4   for(var x = 0; x < arguments.length; x++) {
```

```
5       multiplicacao *= arguments[x];
6     }
7
8     return multiplicacao;
9   }
10
11  var resultado = multiplicacao(3, 6, 2, 8);
12
13  console.log(resultado);
```

Código Javascript A.43: mutiplicacao-2()

Resposta do Exercise A.7

```
1   var vazio = [];
2   var formacoes = ["K00", "K10", "K20", "K30", "K40"];
3
4   console.log(vazio[0]);
5   console.log(formacoes[0]);
6
7   console.log(vazio.length);
8   console.log(formacoes.length);
9
10  for (var x = 0, x < formacoes.length; x++) {
11    console.log(formacoes[x]);
12  }
```

Código Javascript A.74: imprime-array

Resposta do Exercise A.8

```
1   var formacoes = ["K10", "K20", "K30", "K40"];
2
3   formacoes[formacoes.length] = "K00";
4
5   for(var x = 0; x < formacao.length; x++) {
6     console.log(cursos[x]);
7   }
```

Código Javascript A.75: formacoes-array

Resposta do Exercise A.9

```
1   var formacoes = ["K10", "K20", "K30", "K40"];
2
3   formacoes.push("K00");
4
5   for(var x = 0; x < formacao.length; x++) {
6     console.log(cursos[x]);
7   }
```

Código Javascript A.76: metodo-push()

Resposta do Exercise A.10

```
1  var letras = ["a", "b", "c"];
2  var numeros = ["1", "2", "3"];
3
4  var letras_numeros = letras.concat(numeros);
5
6  for(var x = 0; x < letras_numeros.length; x++) {
7    console.log(letras_numeros[x]);
8  }
```

Código Javascript A.77: metodo-concat()

Resposta do Exercise A.11

```
1  var formacoes = ["K00", "K10", "K20", "K30", "K40"];
2
3  var formacoes = formacoes.pop();
4  console.log(formacoes);
```

Código Javascript A.78: metodo-pop()

Resposta do Exercise A.12

```
1  var formacoes = ["K00", "K10", "K20", "K30", "K40"];
2
3  var formacoes = formacoes.pop();
4  console.log(formacoes);
5
6  formacoes.push("K40");
7
8  for(var x = 0; formacoes.length; x++) {  //highlights !!
9    console.log(formacoes[x]);
10 }
```

Código Javascript A.79: metodo2-push()

Resposta do Exercise A.13

```
1  var letras = ["a", "b", "c", "d", "e"];
2
3  letras.reverse();
4
5  for(var x = 0; x < letras.length; x++) {
6    console.log(letras[x]);
7  }
```

Código Javascript A.80: metodo-reverse()

Resposta do Exercise A.14

```
1  var letras = ["a", "b", "c", "d", "e"];
2
3  letras.reverse();
4
5  for(var x = 0; x < letras.length; x++) {
6    console.log(letras[x]);
7  }
8
9  var letras = letras.shift();
10
11 console.log("Elemento removido: " + letras);
12
13 for(var x = 0; x < letras.length; x++) {
14   console.log(letras[x]);
15 }
```

Código Javascript A.81: metodo-shift()

Resposta do Exercise A.15

```
1  var letras = ["a", "b", "c", "d", "e"];
2
3  letras.reverse();
4
5  for(var x = 0; x < letras.length; x++) {
6    console.log(letras[x]);
7  }
8
9  var letras = letras.shift();
10
11 console.log("Elemento removido: " + letras);
12
13 for(var x = 0; x < letras.length; x++) {
14   console.log(letras[x]);
15 }
16
17 letras.unshift("e");
18
19 for(var x = 0; x < letras.length; x++) {
20   console.log(letras[x]);
21 }
```

Código Javascript A.82: metodo-unshift()

Resposta do Exercise A.16

```
1  var formacoes = ["K00", "K10", "K20", "K30", "K40"];
2
3  var formacoes  = cursos.slice(0,2);
4
5  for(var x = 0; x < formacoes.length; x++) {
6    conosole.log(formacao[x]);
```

```
7 }
```

Código Javascript A.83: metodo-slice()

Resposta do Exercise A.17

```
1 var formacao = "K10 - Formação Desenvolvedor Java";
2 var aux = formacao.split("-");
3
4 console.log(aux[0]);
5 console.log(aux[1]);
```

Código Javascript A.84: metodo-split()

Resposta do Exercise B.1

```
1  <DOCTYPE html>
2  <html>
3    <head>
4      <script src="http://code.jquery.com/jquery-latest.js"></script>
5    </head>
6    <body>
7      <input name="Carlos Alberto" class="Funcionario"/>
8      <input name="Ana Maria" class="Funcionario"/>
9      <input name="Paulo Soares" class="Supervisor"/>
10     <input name="Oscar Schneider" class="Gerente"/>
11     <input name="Iris Okamoto" class="Gerente"/>
12   </body>
13 </html>
```

Código HTML B.8: seletores-2.html

Resposta do Exercise B.2

```
1  <DOCTYPE html>
2  <html>
3    <head>
4      <script src="http://code.jquery.com/jquery-latest.js"></script>
5    </head>
6    <body>
7      <input name="Carlos Alberto" class="Funcionario"/>
8      <input name="Ana Maria" class="Funcionario"/>
9      <input name="Paulo Soares" class="Supervisor"/>
10     <input name="Oscar Schneider" class="Gerente"/>
11     <input name="Iris Okamoto" class="Gerente"/>
12   <script>$('.Gerente').val('Instrutor da K19');</script>$
13   </body>
14 </html>
```

Código HTML B.9: seletores-2.html

Resposta do Exercise B.3

```
<DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
    <input name="Carlos Alberto" class="Funcionario"/>
    <input name="Ana Maria" class="Funcionario"/>
    <input name="Paulo Soares" class="Supervisor"/>
    <input name="Oscar Schneider" class="Gerente"/>
    <input name="Iris Okamoto" class="Gerente"/>

    <script>$('input[name|="a","e", "i", "o",
    "u"]').val('K19');</script> $
  </body>
</html>
```

Código HTML B.10: seletores-2.html

Resposta do Exercise B.4

```
<DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>
    <input name="Carlos Alberto" class="Funcionario"/>
    <input name="Ana Maria" class="Funcionario"/>
    <input name="Paulo Soares" class="Supervisor"/>
    <input name="Oscar Schneider" class="Gerente"/>
    <input name="Iris Okamoto" class="Gerente"/>

    <script>$('input').val('K19');</script>$
  </body>
</html>
```

Código HTML B.11: seletores-2.html

Resposta do Exercise B.5

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
  </head>
  <body>

    <h1> Formações K19 </h1>
    <div>
      <ul>
        <li> K00 - Formação Básica </li>
```

```
12          <li> K10 - Formação Desenvolvedor Java </li>
13          <li> K20 - Formação Desenvolvedor Java Avançado </li>
14          <li> K30 - Formação Desenvolvedor .NET </li>
15          <li> K40 - Formação Desenvolvedor Android </li>
16        </ul>
17      </div>
18    </body>
19  </html>
```

Código HTML B.18: eventos-2.html

Resposta do Exercise B.6

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7       <h1> Formações K19 </h1>
8       <div>
9         <ul>
10          <li> K00 - Formação Básica </li>
11          <li> K10 - Formação Desenvolvedor Java </li>
12          <li> K20 - Formação Desenvolvedor Java Avançado </li>
13          <li> K30 - Formação Desenvolvedor .NET </li>
14          <li> K40 - Formação Desenvolvedor Android </li>
15        </ul>
16      </div>
17      <script>
18      $('li').click(function(){ alert("Elemento li clicado:
19      "+$(this).text());});
20      </script>
21    </body>
22  </html>
```

Código HTML B.19: eventos-2.html

Resposta do Exercise B.7

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7
8       <h1> Formações K19 </h1>
9       <div>
10        <ul>
11          <li> K00 - Formação Básica </li>
12          <li> K10 - Formação Desenvolvedor Java </li>
13          <li> K20 - Formação Desenvolvedor Java Avançado </li>
14          <li> K30 - Formação Desenvolvedor .NET </li>
15          <li> K40 - Formação Desenvolvedor Android </li>
16        </ul>
17      </div>
18      <script>
```

```
19    $('li').dbclick(function(){ alert("Elemento clicado
20    2x:"+$(this).text());});
21      </script>
22    </body>
23  </html>
```

Código HTML B.20: eventos-2.html

Resposta do Exercise B.8

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7
8       <h1> Formações K19 </h1>
9       <div>
10        <label for="formacoes"> Selecione uma formação:</label>
11        <select name="formacoes" id="formacoes">
12          <option>----</option>
13        <option value="K00"> K00 - Formação Básica</option>
14        <option value="K10"> K10 - Formação Desenvolvedor Java </option>
15    ***   <option value="K20"> K20 - Formação Desenvolvedor Java Avançado </option>
16        <option value="K30"> K30 - Formação Desenvolvedor .NET </option>
17        <option value="K40"> K40 - Formação Desenvolvedor Android </option>
18        </select>
19      </div>
20    <script>
21      $('#formacoes').change(function(){
22         alert("Formação selecionado: "+$("#formacoes
23         option:selected").text()); });
24    </script>
25    </body>
26  </html>
```

Código HTML B.21: eventos-2.html

Resposta do Exercise B.9

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7
8     <h1>Cursos K19</h1>
9     <div>
10      <ul>
11        <li>K31 - C# e Orientação a Objetos</li>
12        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
13        <li>K11 - Java e Orientação a Objetos</li>
14        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
15      </ul>
16    </div>
17    <div>
```

```
18      <button id="mostrar">Mostrar</button>
19      <button id="ocultar">Ocultar</button>
20      <button id="fadeto">FadeTo</button>
21    </div>
22    <script>
23      $("#mostrar").click(function(){ $("li").fadeIn("slow"); });
24      $("#ocultar").click(function(){ $("li").fadeOut("slow"); });
25      $("#fadeto").click(function(){ $("li").fadeTo("slow", 0.2); });
26    </script>
27    </body>
28  </html>
```

Código HTML B.30: efeitos.html

Resposta do Exercise B.10

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7
8     <h1>Cursos K19</h1>
9     <div>
10      <ul>
11        <li>K31 - C# e Orientação a Objetos</li>
12        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
13        <li>K11 - Java e Orientação a Objetos</li>
14        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
15      </ul>
16    </div>
17    <div>
18      <button id="mostrar">Mostrar</button>
19      <button id="ocultar">Ocultar</button>
20    </div>
21    <script>
22      $("#mostrar").click(function(){ $("li").slideDown("slow"); });
23      $("#ocultar").click(function(){ $("li").slideUp("slow"); });
24    </script>
25    </body>
26  </html>
```

Código HTML B.31: efeitos.html

Resposta do Exercise B.11

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6   <body>
7
8     <h1>Cursos K19</h1>
9     <div>
10      <ul>
11        <li>K31 - C# e Orientação a Objetos</li>
```

```
12        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
13        <li>K11 - Java e Orientação a Objetos</li>
14        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
15      </ul>
16    </div>
17    <div>
18      <button id="mostrar-ocultar">Mostrar/Ocultar</button>
19    </div>
20    <script>
21      $("#mostrar-ocultar").click(function(){ $("li").slideToggle("slow"); });
22    </script>
23    </body>
24  </html>
```

Código HTML B.32: efeitos.html

Resposta do Exercise B.12

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7
8       <h1> K19 treinamentos </h1>
9       <div>
10        <ul>
11          <li> K00 - Formação Básica </li>
12          <li> K10 - Formação Desenvolvedor Java </li>
13          <li> K20 - Formação Desenvolvedor Java Avançado </li>
14          <li> K30 - Formação Desenvolvedor .NET </li>
15          <li> K40 - Formação Desenvolvedor Android </li>
16        </ul>
17      </div>
18      <div>
19        <button id="mostrar"Mostrar</button>
20        <button id="ocultar">Ocultar</button>
21      </div>
22      <script>
23        $("#mostrar").click(function(){ $("li").show("slow");});
24        $("#ocultar").click(function(){ $("li").hide("slow");});
25      </script>
26    </body>
27  </html>
```

Código HTML B.33: efeitos-2.html

Resposta do Exercise B.13

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7       <h1> K19 treinamentos </h1>
8       <div>
```

```
9       <ul>
10        <li> K00 - Formação Básica </li>
11        <li> K10 - Formação Desenvolvedor Java </li>
12        <li> K20 - Formação Desenvolvedor Java Avançado </li>
13        <li> K30 - Formação Desenvolvedor .NET </li>
14        <li> K40 - Formação Desenvolvedor Android </li>
15      </ul>
16    </div>
17    <div>
18      <button id="mostrar-ocultar">Mostrar/Ocultar</button>
19    </div>
20    <script>
21      $("#mostrar-ocultar").click(function(){
22      $("li").toggle("slow");});
23    </script>
24  </body>
25 </html>
```

Código HTML B.34: efeitos-2.html

Resposta do Exercise B.14

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src="http://code.jquery.com/jquery-latest.js"></script>
5    </head>
6    <body>
7      <p> Lorem Ipsum Dolor </p>
8      <div>
9      <button id="Alterar conteúdo" /button>
10     </div>
11       <script>
12       $("alterar").click(function(){$("p").html("K19 - Treinamentos);});
13       </script>
14   </body>
15 </html>
```

Código HTML B.41: html-2.html

Resposta do Exercise B.15

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src="http://code.jquery.com/jquery-latest.js"></script>
5    </head>
6    <body>
7      <p> Lorem Ipsum Dolor </p>
8      <div>
9      <button id="Alterar conteúdo</button>
10     </div>
11       <script>
12       $("alterar").click(function(){$("p").html("K19 - Treinamentos");});
13       </script>
14     <div>
15     <button id="prepend"> Prepend</button>
16     <button id="append"> Append</button>
```

```
17      </div>
18      <script>
19        $("#prepend").click(function(){$("p").prepend("");});
20        $("#append").click(function(){$("p").append("");});
21      </script>
22    </body>
23  </html>
```

Código HTML B.42: html-2.html

Resposta do Exercise B.16

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <script src="http://code.jquery.com/jquery-latest.js"></script>
5     </head>
6     <body>
7       <p> Lorem Ipsum Dolor </p>
8       <div>
9       <button id="before">Before</button>
10      <button id="after">After</button>
11      </div>
12      <script>
13      $("#before").click(function(){$("p").before("");});
14      $("#after").click(function(){$("p").after("");});
15      </script>
16    </body>
17  </html>
```

Código HTML B.43: html-2.html