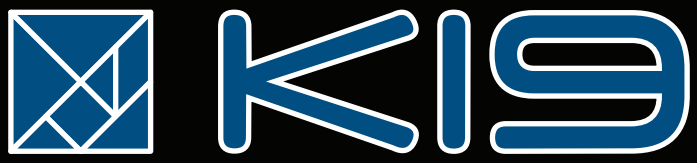

TREINAMENTOS

Desenvolvimento Web com HTML, CSS e Javascript

Desenvolvimento Web com HTML, CSS e Javascript

1 de agosto de 2012

Sumário i

Sobre a K19 1

Seguro Treinamento 2

Termo de Uso 3

Cursos 4

1 Introdução 1
- 1.1 Client side e server side . . . 1
- 1.2 HTML, CSS e Javascript . . . 1

2 HTML 3
- 2.1 Estrtutura Básica . . . 3
- 2.2 Exercícios de Fixação . . . 4
- 2.3 Semântica HTML . . . 5
- 2.4 Parágrafos . . . 6
- 2.5 Exercícios de Fixação . . . 7
- 2.6 Cabeçalhos . . . 7
- 2.7 Exercícios de Fixação . . . 8
- 2.8 Links . . . 9
- 2.9 Exercícios de Fixação . . . 10
- 2.10 Exercícios Complementares . . . 11
- 2.11 Âncoras . . . 11
- 2.12 Exercícios de Fixação . . . 12
- 2.13 Imagens . . . 12
- 2.14 Exercícios de Fixação . . . 13
- 2.15 Tabelas . . . 13
- 2.16 Exercícios de Fixação . . . 17
- 2.17 Listas . . . 17
- 2.18 Exercícios de Fixação . . . 19
- 2.19 Exercícios de Fixação . . . 20

2.20 Exercícios de Fixação . . . 21
2.21 Formulário . . . 21
2.22 Exercícios de Fixação . . . 27
2.23 Exercícios de Fixação . . . 28

3 CSS **31**
3.1 Estrutura de uma regra CSS . . . 34
3.2 Tipos de seletores . . . 35
3.3 Exercícios de Fixação . . . 37
3.4 Principais propiedades CSS . . . 38
3.5 Box model . . . 40
3.6 Posicionando elementos . . . 41
3.7 Cores em CSS . . . 43
3.8 Unidades de medida . . . 43
3.9 Desafios . . . 43

4 JavaScript **45**
4.1 Declarando e inicializando variáveis em JavaScript . . . 45
4.2 Operadores . . . 45
4.3 Controle de fluxo . . . 48
4.4 Exercícios de Fixação . . . 49
4.5 Exercícios Complementares . . . 51
4.6 Funções JavaScript . . . 52
4.7 Objetos JavaScript . . . 53
4.8 Arrays . . . 54
4.9 Exercícios de Fixação . . . 55

A Javascript Avançado **57**
A.1 Objetos . . . 57
A.2 Exercícios de Fixação . . . 61
A.3 Funções . . . 64
A.4 Exercícios de Fixação . . . 66
A.5 Arrays . . . 68
A.6 Métodos das Strings . . . 71
A.7 Exercícios de Fixação . . . 72

B JQuery **75**
B.1 Introdução . . . 75
B.2 Sintaxe . . . 76
B.3 Seletores . . . 76
B.4 Exercícios de Fixação . . . 77
B.5 Eventos . . . 78
B.6 Exercícios de Fixação . . . 79
B.7 Efeitos . . . 82
B.8 Exercícios de Fixação . . . 84
B.9 Exercícios Complementares . . . 86
B.10 HTML . . . 86
B.11 Exercícios de Fixação . . . 87

C Respostas **91**

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

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

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 idéia de desenvolvimento web ficou associada apenas à construção de páginas cuja função era simplesmente levar ao usuário um determinado conteúdo. Porém, com a popularização da internet, novas necessidades foram surgindo em diversas áreas como a do entretenimento, assim como a dos negócios. Cada vez mais jogos online foram aparecendo, redes sociais ganharam forças graças à grande interatividade permitida entre os usuários, gravadoras de música passaram a vender seus títulos através de canais especializados e ferramentas de produtividade começaram a rodar na tal da "nuvem". Enfim, necessidades antes inexistentes surgiram numa velocidade muito grande e 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 o responsável por produzir o cógido HTML, CSS, Javascript, PHP, SQL e de qualquer outra tecnologia que fosse necessário. Essa pessoa era chamada de webmaster. Com a evolução dos sites a figura do webmaster como era conhecida foi desaparecendo, pois a complexidade e volume de trabalho para o desenvolvimento de uma aplicação web foi ficando muito grande para apenas uma pessoa, ou para um grupo muito pequeno de desenvolvedores (webmasters). Hoje a figura do webmaster ainda existe, mas seu papel mudou um pouco, pois esse profissional atua mais como um gerente que possui bom conhecimento das diversas tecnologias empregadas nos desenvolvimento de uma aplicação web. Ele pode ou não participar diretamente do desenvolvimento, ou seja, pode ou não "botar a mão na massa".

Já que as tarefas antes de responsabilidade do webmaster foram delegadas a outros desenvolvedores, naturalmente foram aparecendo algumas especializações que podemos separar basicamente em dois grupos: desenvolvedores front-end e back-end. Em geral os desenvolvedores front-end são responsáveis pela interface com a qual o usuário irá interagir enquanto que os desenvolvedores back-end são responsáveis pela implantação das regras de negócio na aplicação.

Client side e server side

Com a diferenciação entre desenvolvedor front-end e back-end em mente, podemos dizer que o front-end de uma aplicação web irá trabalhar com as linguagens categorizadas como client side, ou seja, aquelas que são executadas no lado do cliente (navegador). Atualmente as principais linguagens/tecnologias client side são HTML, CSS, Javascript, Adobe Flash, Microsoft Silverlight e VBScript.

Já o desenvolvedor back-end irá trabalhar com linguagens como Java, C#, VB.NET, PHP, Ruby, Python, SQL entre outras que são categorizadas como server side, ou seja, serão executadas no lado do servidor. Isso não significa que os desenvolvedores front-end não precisam conhecer as linguagens utilizadas pelo back-end e vice-versa. O que ocorre na prática é a especialização do profissional 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/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 irá cuidar da formatação visual do conteúdo apresentado e o código Javascript permitirá que a página possua algum tipo de comportamento ("inteligência") e que alguma interação possa ser feita com o usuário.

Nos próximos capítulos iremos abordar mais a fundo cada uma dessas três tecnologias.

HTML

CAPÍTULO 2

Quando acessamos uma página web estamos interessados na informação contida nessa página, seja ela na forma de texto, imagem ou vídeo. O conteúdo de uma página web é, em geral, escrito na linguagem HTML (HyperText Markup Language), que é uma linguagem de marcação originalmente proposta por Tim Berners-Lee no final da década de 1980. Sua idéia era disseminar documentos científicos de uma maneira simples e com uma sintaxe flexível o suficiente para que mesmo aqueles sem muita familiaridade com a linguagem pudessem fazer o mesmo.

Desde sua proposta até os dias de hoje a linguagem sofreu diversas alterações em sua especificação de uma versão para outra, sendo a mais atual a especificação do HTML5 que está na fase de "trabalho em progresso"(working draft), ou seja, sua especificação ainda não foi concluída, porém já existem navegadores implementando alguns dos novos recursos definidos no HTML5.

As especificações do 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.

Estrtutura 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 estar 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

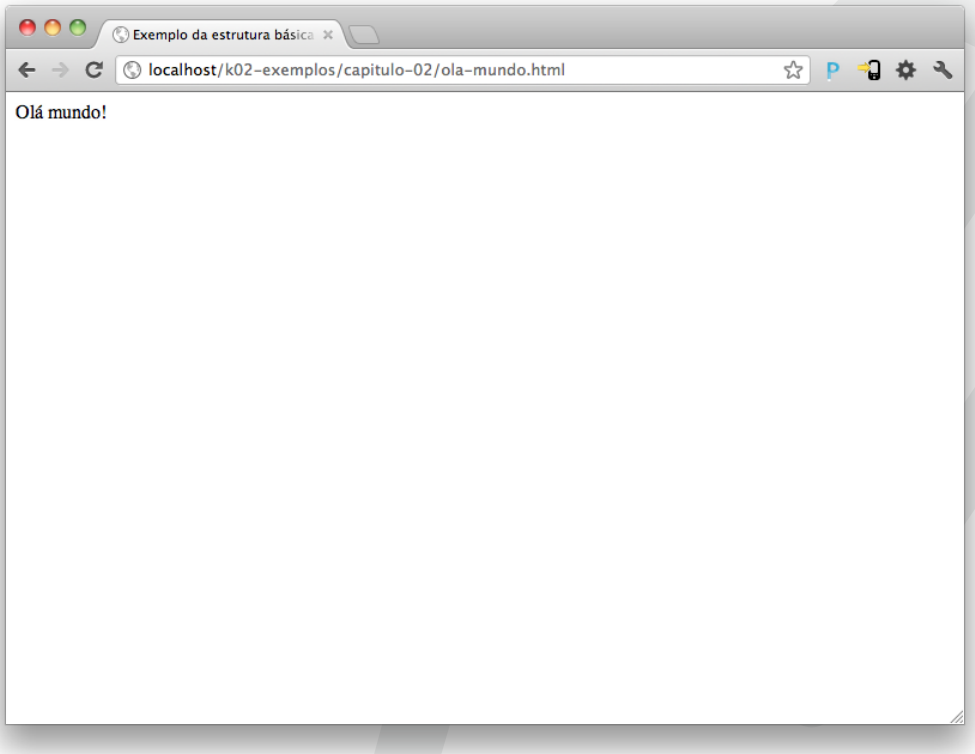

Figura 2.1: Exemplo da estrutura básica de um documento HTML

No exemplo acima temos um elemento HTML representado por sua tag "p"e um filho de texto simples "Olá Mundo!".

Exercícios de Fixação

1 Na pasta *Desktop* do seu usuário crie uma nova pasta com o seu primeiro nome. Dentro dessa pasta crie outra pasta com o nome *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 com o nome *ola-mundo.html* e salve 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.

Semântica HTML

De acordo com a especificação, cada tag possui um significado, isto é, o que o conteúdo de um determinado elemento representa. Muitos autores utilizam o termo semântica HTML ao se referirem ao uso correto dos significados de cada tag. 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 2.3: Exemplo de uso correto da semântica HTML

Neste exemplo utilizamos novamente a tag *p* e de acordo com a especificação o elemento p representa um parágrafo. Neste caso, o elemento de tag p foi utilizado de maneira correta. Vejamos outro exemplo:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Meus amigos - Site do 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 2.4: Exemplo de uso incorreto da semântica HTML

Dessa vez utilizamos a tag address e, de acordo com a especificação, "O elemento address deve ser utilizada pelos autores para fornecer informações de contato de um documento ou para a maior parte de um documento. Este elemento normalmente aparece no início ou no final de um documento".

Se observarmos o exemplo mais atentamente, trata-se de uma página do site do Jonas (repare no título da página), portanto provavelmente o autor da página é o Jonas e não o Rafael nem o Marcelo. Além disso, devemos evitar o uso da tag address para informar endereços postais a menos que ele seja relevante ao documento no qual ela foi inserida.

Parágrafos

Os parágrafos dentro de um documento HTML, em geral, são representados através da tag *p*. Uma das características da tag *p* é que ela ocupa horizontalmente todo o espaço definido pelo elemento pai. Esse é o comportamento dos **elementos de bloco** que discutiremos com mais detalhes no tópico sobre CSS.

Por enquanto o importante é termos em mente que, pelo fato da tag *p* se um elemento de bloco, o navegador irá ajustar o texto à largura do elemento pai realizando todas as quebras de linha necessárias. Caso seja necessário forçar uma quebra de linha no meio de um texto, podemos utilizar a tag *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 comportameto da quebra de linha automática, ou seja, sem utilizar
        nenhum recurso para que a quebra ocorra.</p>

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

Código HTML 2.5: Exemplo de quebra de linha em um parágrafo

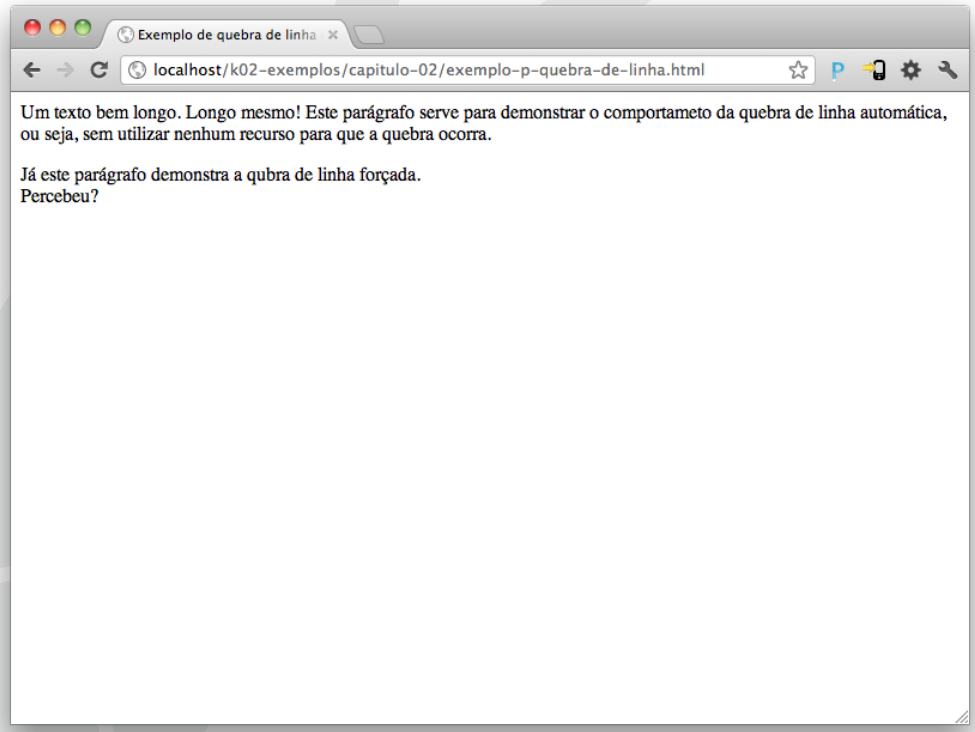

Figura 2.2: Exemplo de quebra de linha em um parágrafo

Exercícios de Fixação

3 Crie um novo documento HTML, insira o código abaixo e salve-o com o nome *p-quebra-de-linha.html* na pasta *html*. Em seguida abra o 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 comportameto da quebra de linha automática, ou seja, sem utilizar
        nenhum recurso para que a quebra ocorra.</p>

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

Código HTML 2.6: p-quebra-de-linha.html

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 2.7: Exemplo de cabeçalhos

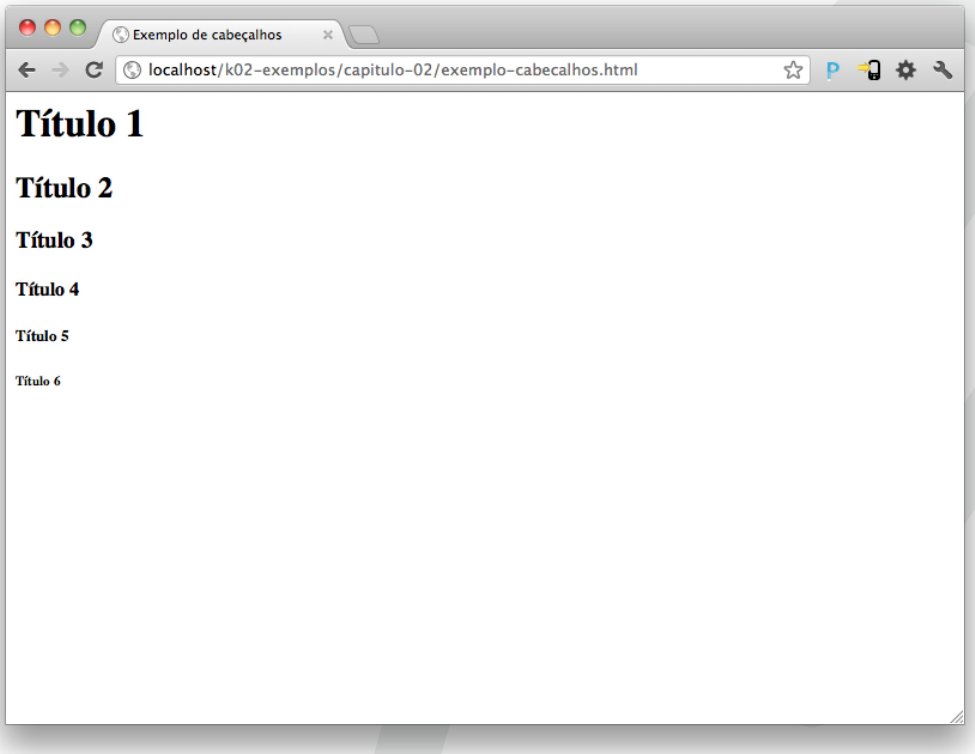

Figura 2.3: Exemplo de cabeçalhos

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 mais adiante.

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, insira o código abaixo e salve-o com o nome *cabecalhos-1.html* na pasta *html*. Em seguida abra o arquivo em um navegador.

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

```
4         <title>K02 - Desenvolvimento Web com HTML, CSS e Javascript</title>
5     </head>
6     <body>
7         <h1>K02 - Desenvolvimento Web com HTML, CSS e Javascript</h1>
8
9         <p>Atualmente, praticamente todos os sistemas corporativos possuem
10        interfaces web. Para quem deseja atuar no mercado de desenvolvimento
11        de software, é obrigatório o conhecimento das linguagens: HTML, CSS
12        e JavaScript.</p>
13
14    <p>Essas linguagens são utilizadas para o desenvolvimento da camada de
15    apresentação das aplicações web.</p>
16
17    <h2>Pré-requisitos</h2>
18
19    <p>Familiaridade com algum sistema operacional (Windows/Linux/Mac OS X)</p>
20    <p>Lógica de programação</p>
21
22    <h2>Agenda</h2>
23
24    <h3>Aos domingos</h3>
25
26    <p>xx/xx/xxxx das 08:00 às 14:00</p>
27    <p>xx/xx/xxxx das 14:00 às 20:00</p>
28
29    <h3>Aos sábados</h3>
30
31    <p>xx/xx/xxxx das 08:00 às 14:00</p>
32    <p>xx/xx/xxxx das 14:00 às 20:00</p>
33    </body>
34 </html>
```

Código HTML 2.8: 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 o seu documento. 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, a tag *a* sem atributos não irá criar 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 2.10: Exemplo de uso da tag a

Além do atributo `href` podemos utilizar atributo `target` no qual informamos onde iremos abrir o documento. 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.

```
<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="pagina1.html" target="_blank">Abre em outra janela/aba</a></p>
      <p><a href="pagina2.html" target="_self">Abre na mesma janela</a></p>
    <p><a href="pagina3.html">Abre na mesma janela</a></p>
  </body>
</html>
```

Código HTML 2.11: Exemplo de uso da tag a *com o atributo* `target`

Importante

Ao longo da evolução do HTML as tags `frame` e `iframe` foram caindo em desuso até que no HTML5 foram totalmente retiradas da especificação. Contudo a grande maioria dos navegadores ainda interpretam as duas tags 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 das tags `frame` e `iframe` ao máximo.

Exercícios de Fixação

6 Crie um documento 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.

Exercícios Complementares

1 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 pelo `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 o HTML nos permite 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">
        <title>Exemplo de uso da tag a como âncora</title>
    </head>
    <body>
        <p><a href="#mais_info">Veja mais informações</a></p>
        <p><a href="pagina2.html#outra_ancora">Âncora em outra página</a></p>

        ...
        ...
        ...

        <a name="mais_info">Mais informações</a>

        <p>
            ...
          ...
          ...
        </p>
    </body>
</html>
```

Código HTML 2.14: 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` para criarmos as âncoras. Porém, no HTML5, a recomendação do W3C é que se utilize o atributo `id`. 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 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.

8 Continuando o exercício anterior, crie um novo link que aponte para uma âncora localizada em outra página. Crie uma página com uma âncora para a qual o link que você acabou de criar irá apontar.

Imagens

Provavelmente os sites na internet seriam muito mais entediantes se não fosse possível adicionar algumas imagens ao conteúdo das páginas. Como não queremos que as nossas páginas fiquem muito monótonas, neste capítulo iremos utilizar a tag `img` e 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.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Exemplo de uso da tag img</title>
    </head>
    <body>
      <h1>K19 Treinamentos</h1>
        <img src="http://www.k19.com.br/css/img/main-header-logo.png" />

        <h2>Cursos</h2>
        <p>
          <img src="http://www.k19.com.br/css/img/k01-logo-large.png" />
          K01 - Lógica de Programação
        </p>
        <p>
          <img src="http://www.k19.com.br/css/img/k02-logo-large.png" />
          K02 - Desenvolvimento Web com HTML, CSS e JavaScript
        </p>
        <p>
          <img src="http://www.k19.com.br/css/img/k03-logo-large.png" />
          K03 - SQL e Modelo Relacional
        </p>
        <p>
          <img src="http://www.k19.com.br/css/img/k11-logo-large.png" />
          K11 - Orientação a Objetos em Java
        </p>
        <p>
          <img src="http://www.k19.com.br/css/img/k12-logo-large.png" />
          K12 - Desenvolvimento Web com JSF2 e JPA2
        </p>
        ...
        ...
        ...
    </body>
</html>
```

Código HTML 2.18: Exemplo de uso da tag `img`

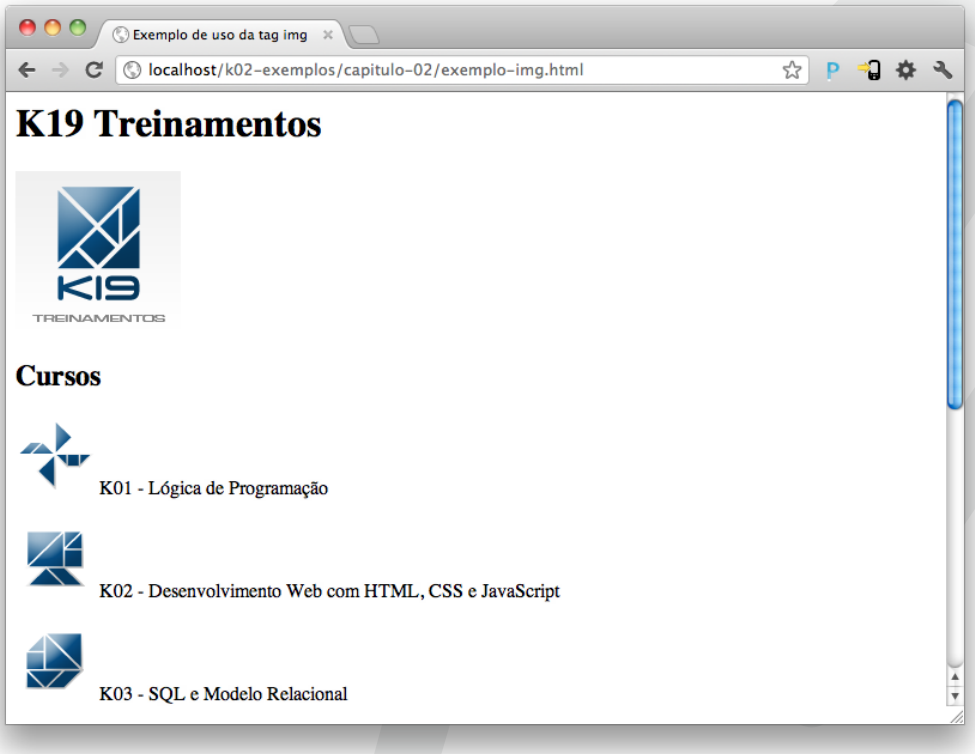

Figura 2.4: Exemplo de uso da tag `img`

Exercícios de Fixação

9 Escolha uma ou mais imagens quaisquer no computador ou na internet. Crie um documento HTML que contenha um ou mais elementos com a tag `img` para exibir as imagens escolhidas.

Tabelas

Suponha que você esteja desenvolvendo o site de uma empresa que necessita divulgar alguns relatórios em uma de suas páginas. Existe uma grande chance que os dados dos relatórios venham de planilhas eletrônicas. Como os navegadores interpretam apenas código HTML, você ficará encarregado de tranferir para o formato HTML as informações dos relatórios que estão no formato da planilha eletrônica. Surge aí uma necessidade: exibir no navegador um conjunto de informações de forma organizada.

Para resolver esse problema temos a tag `table` do HTML que nos premite apresentar um conjunto de dados em 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 2.20: Exemplo de uso da tag `table`

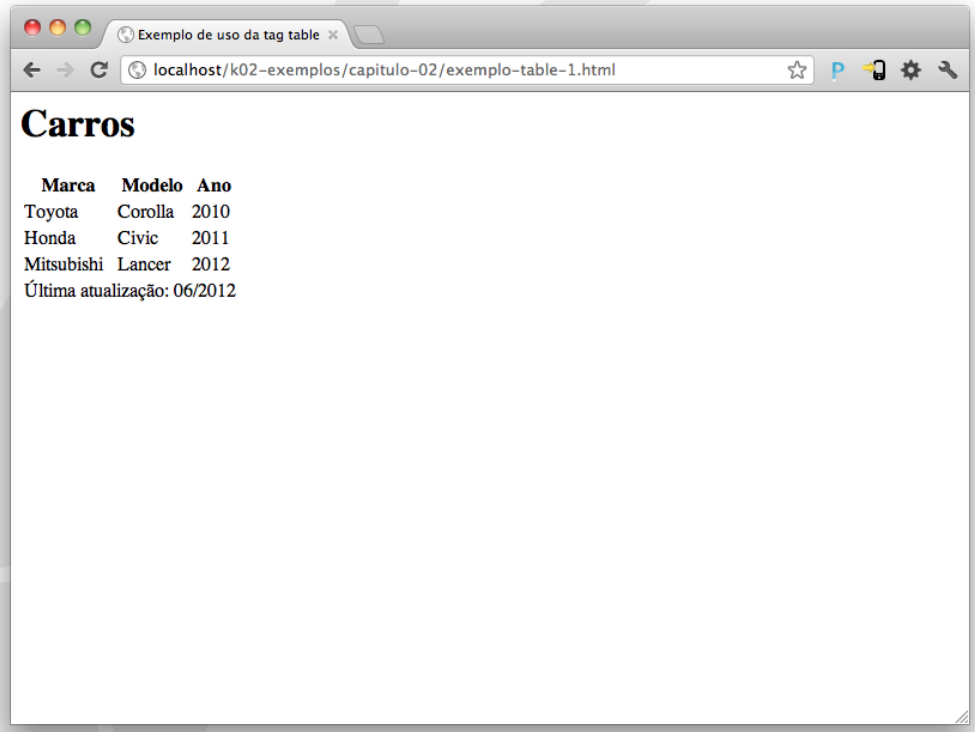

Figura 2.5: Exemplo de uso da tag `table`

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

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

      <table>
        <thead>
          <tr>
            <th>Marca</th>
            <th>Modelo</th>
            <th>Ano</th>
          </tr>
        </thead>
        <tfoot>
          <tr>
            <td colspan="3">Última atualização: 06/2012</td>
          </tr>
        </tfoot>
        <tbody>
          <tr>
            <td>Toyota</td>
            <td>Corolla</td>
            <td>2010</td>
          </tr>
          <tr>
            <td>Honda</td>
            <td>Civic</td>
            <td>2011</td>
          </tr>
          <tr>
            <td>Mitsubishi</td>
            <td>Lancer</td>
            <td>2012</td>
          </tr>
        </tbody>
      </table>
    </body>
</html>
```

Código HTML 2.21: Exemplo de uso da tag `table`

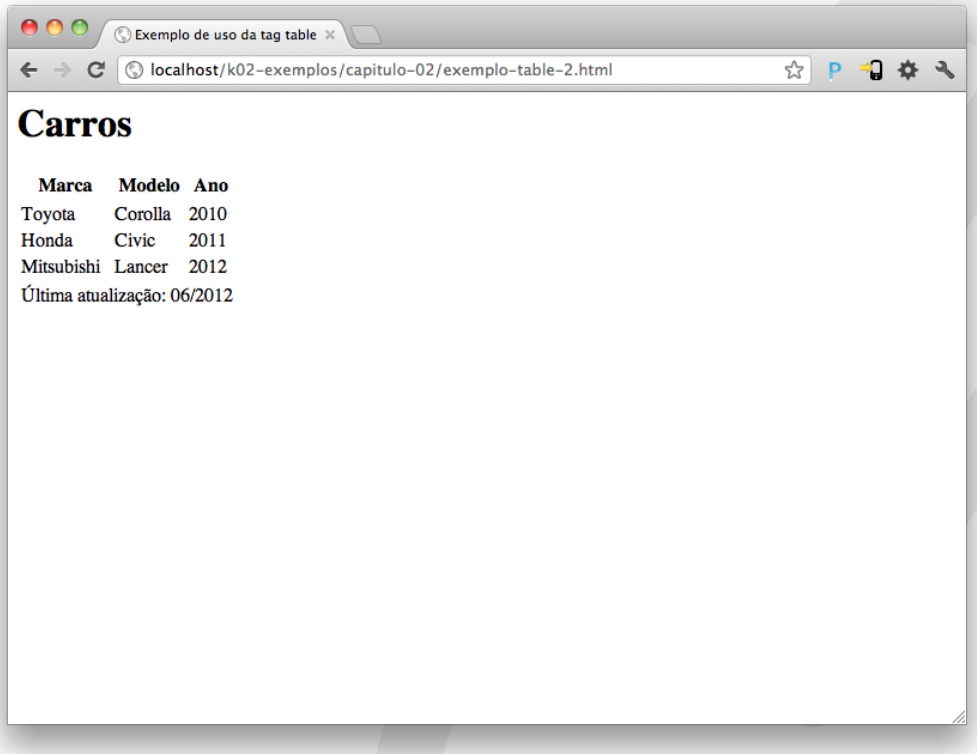

Figura 2.6: Exemplo de uso da tag `table`

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 com a tag `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

10 Crie uma página que contenha uma tabela de acordo com a imagem abaixo:

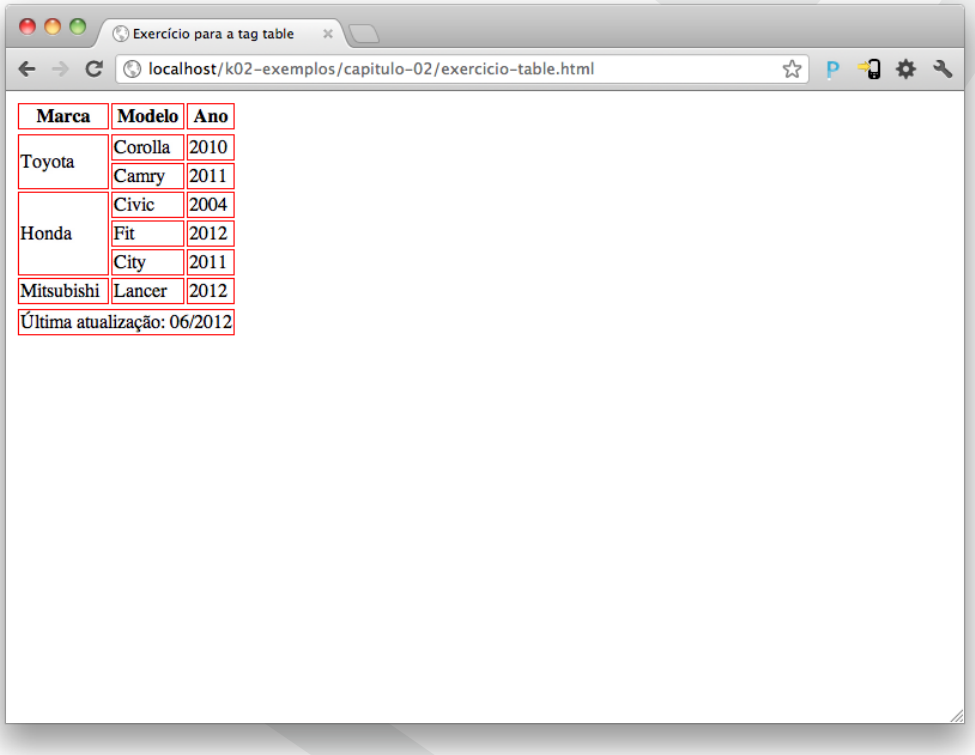

Figura 2.7: Exercício para a tag `table`

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

Listas

Em um documeno 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 com a tag `dl` deve possuir pelo menos um filho com a tag `dt` seguido de um elemento com a tag dd, isto é, um item na lista de definição é composto por um par de elementos com as tags `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>K19 Treinamentos</h1>
      <h2>Cursos</h2>

      <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 2.23: Exemplo de uso da tag `dl`

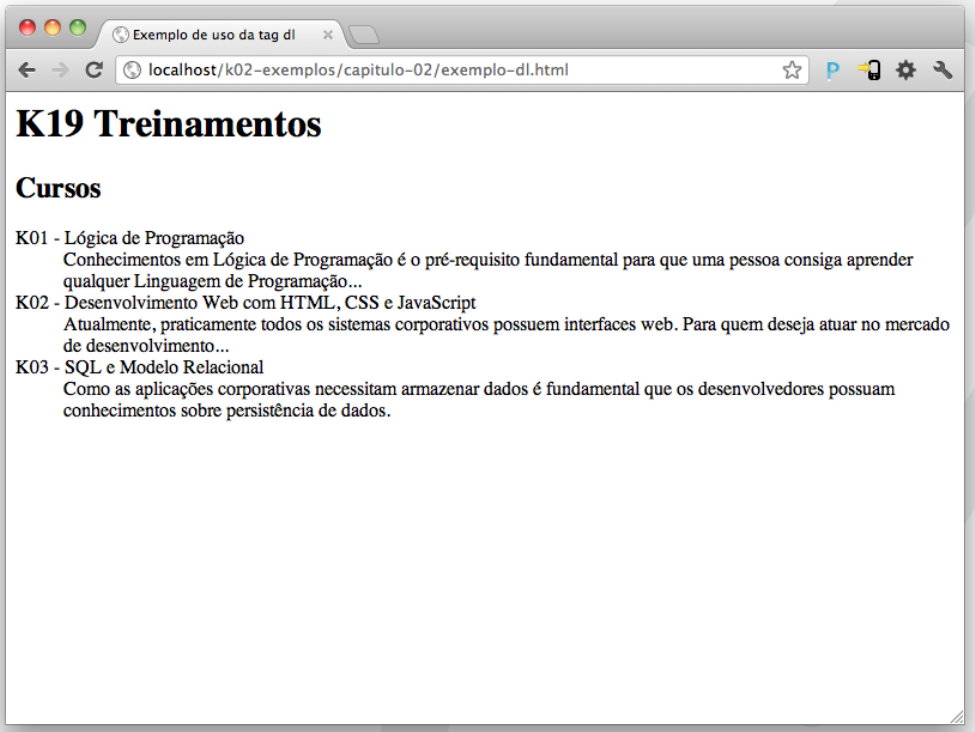

Figura 2.8: Exemplo de uso da tag `dl`

Exercícios de Fixação

11 Crie um documento HTML que contenha o cardápio de um restaurante com os nomes dos seus pratos e uma breve descrição sobre os mesmos.

Lista ordenada

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

```
1  <html>
2      <head>
3          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4          <title>Exemplo de uso da tag ol</title>
5      </head>
6      <body>
7        <h1>K19 Receitas</h1>
8        <h2>Macarrão instantâneo</h2>
9        <h3>Modo de preparo</h3>
10
11       <ol>
12         <li>Ferver 600ml de água em uma panela.</li>
13         <li>Retirar o macarrão do pacote.</li>
14         <li>Colocar o macarrão na panela no fogo baixo.</li>
15         <li>Cozinhar o macarrão por 3min.</li>
```

```
16          <li>Temperar a gosto.</li>
17        </ol>
18      </body>
19  </html>
```

Código HTML 2.25: Exemplo de uso da tag `ol`

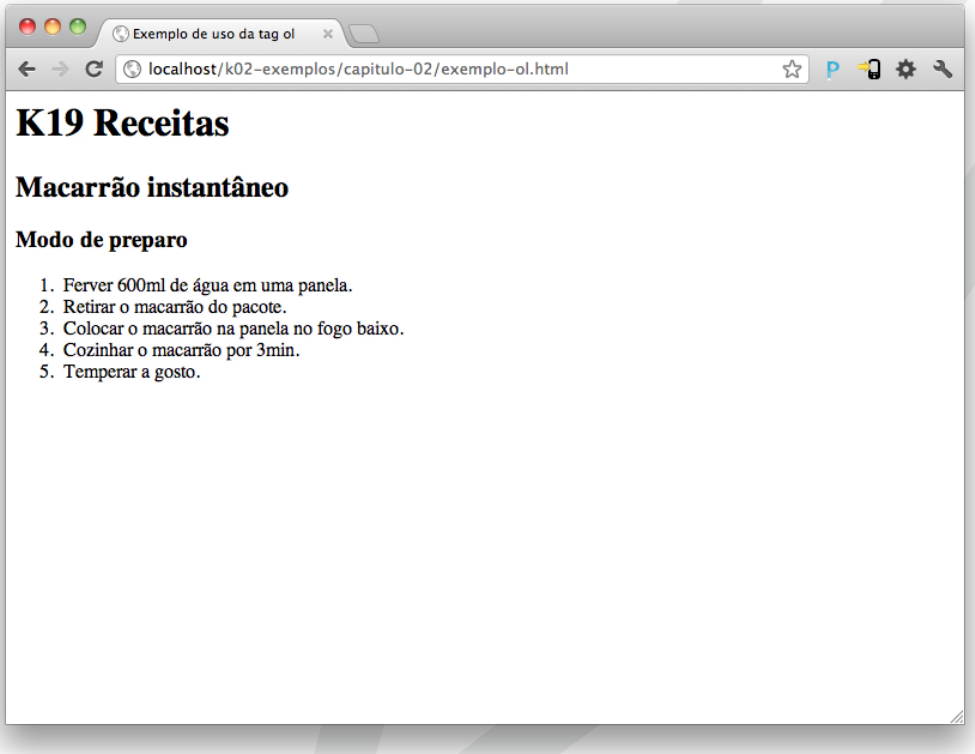

Figura 2.9: Exemplo de uso da tag `ol`

Exercícios de Fixação

12 Crie um documento HTML que contenha um manual que explica passo-a-passo o uso de um caixa eletrônico para a operação de saque.

Lista sem ordem

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

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

```
7       <h1>K19 Treinamentos</h1>
8       <h2>K02 - Desenvolvimento Web com HTML, CSS e JavaScript</h2>
9       <h3>Pré-requisitos</h3>
10
11      <ul>
12        <li>Conhecimento de algum sistema operacional (Windows/Linux/MacOS X)</li>
13        <li>Lógica de programação</li>
14      </ul>
15    </body>
16  </html>
```

Código HTML 2.27: Exemplo de uso da tag `ul`

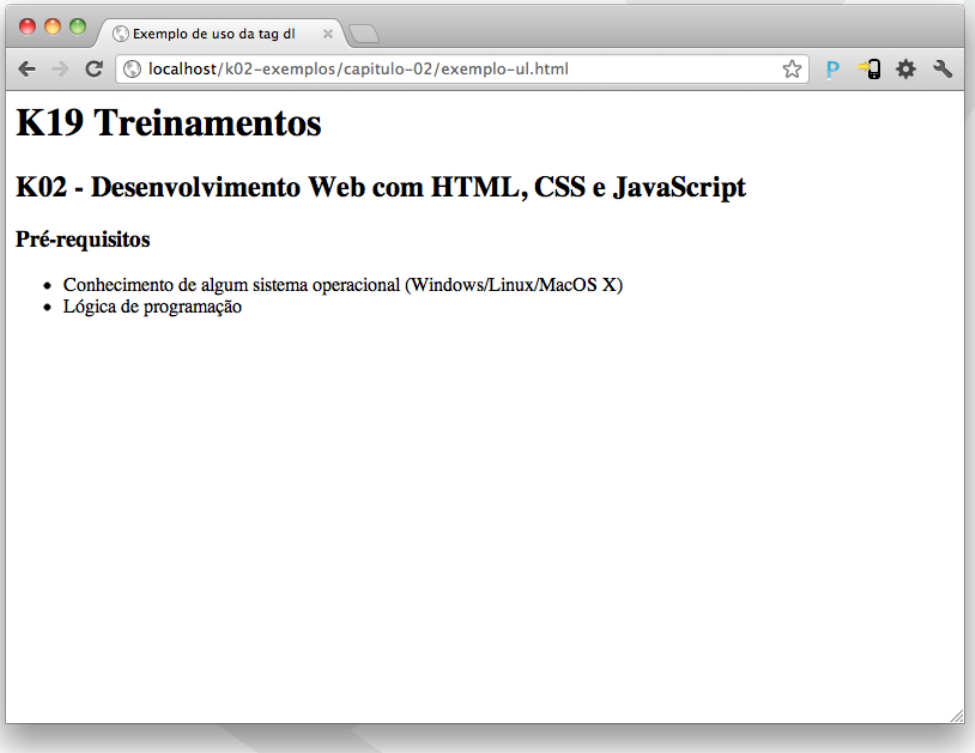

Figura 2.10: Exemplo de uso da tag `ul`

Exercícios de Fixação

13 Crie um documento HTML que contenha a lista dos cursos da Formação Básica 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.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Exemplo de uso da tag input</title>
    </head>
    <body>
      <form action="pagina.html" method="get">
        <p>
          text:
          <input type="text" />
        </p>
        <p>
          password:
          <input type="password" />
        </p>
        <p>
          checkboxes:
          <input type="checkbox" name="checkgroup" />
          <input type="checkbox" name="checkgroup" />
          <input type="checkbox" name="checkgroup" />
        </p>
        <p>
          radios:
          <input type="radio" name="checkgroup" />
          <input type="radio" name="checkgroup" />
          <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 2.29: Exemplo de uso da tag `input`

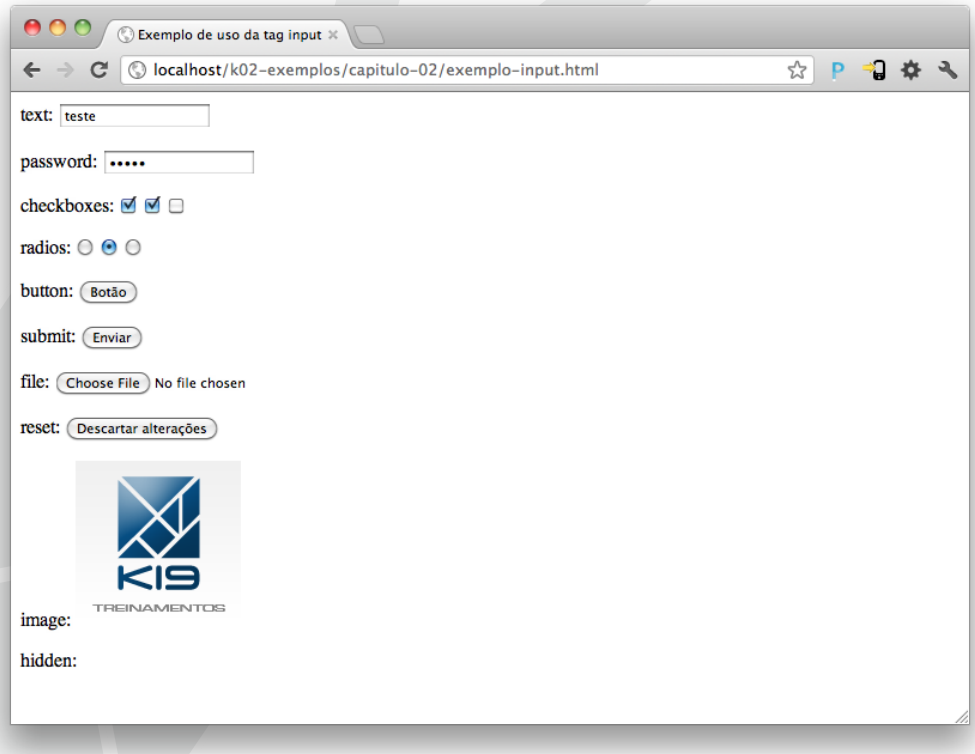

Figura 2.11: Exemplo de uso da tag `input`

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 com a tag `option`. Tais elementos devem ser filhos diretos ou indiretos do elemento com a tag `select`. Além disso, cada item pode conter o atributo `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">
            <option value="desktops">Desktops</option>
            <option value="notebooks">Notebooks</option>
            <option value="tablets">Tablets</option>
            <option value="celulares">Celulares</option>
          </select>
        </p>
      </form>
    </body>
</html>
```

Código HTML 2.30: Exemplo de uso da tag `select`

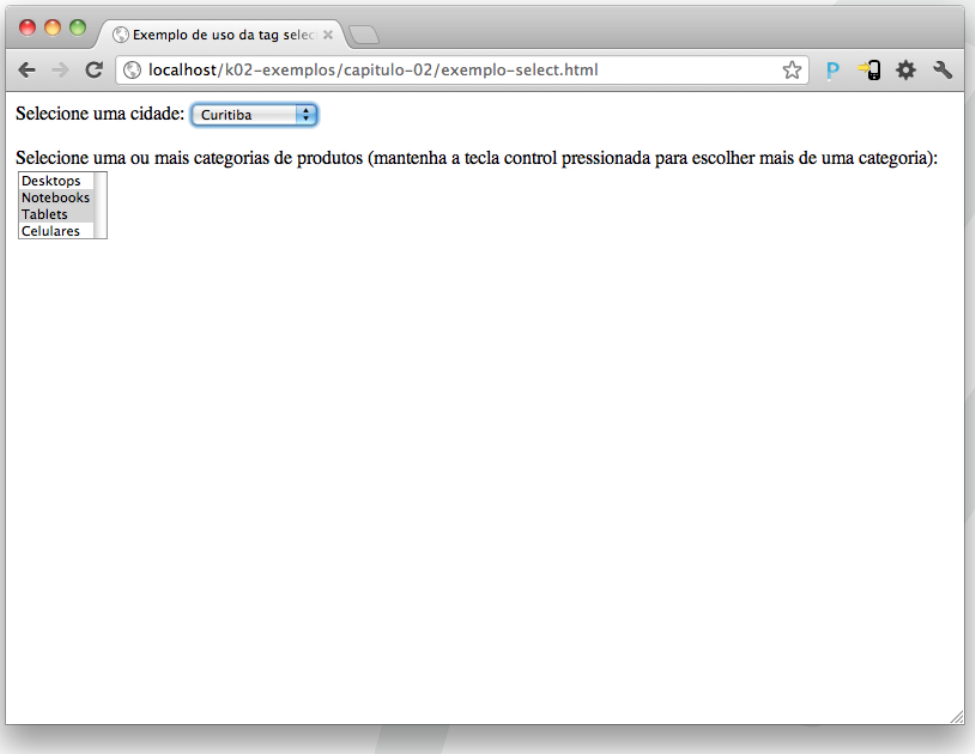

Figura 2.12: Exemplo de uso da tag `select`

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

```
<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>
            <optgroup label="Região Sudeste">
              <option value="sao-paulo">São Paulo</option>
              <option value="rio-de-janeiro">Rio de Janeiro</option>
            </optgroup>
            <optgroup label="Região Sul">
              <option value="porto-alegre">Porto Alegre</option>
              <option value="curitiba">Curitiba</option>
            </optgroup>
          </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">
            <optgroup label="De mesa">
              <option value="desktops">Desktops</option>
            </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 2.31: Exemplo de uso da tag `select`

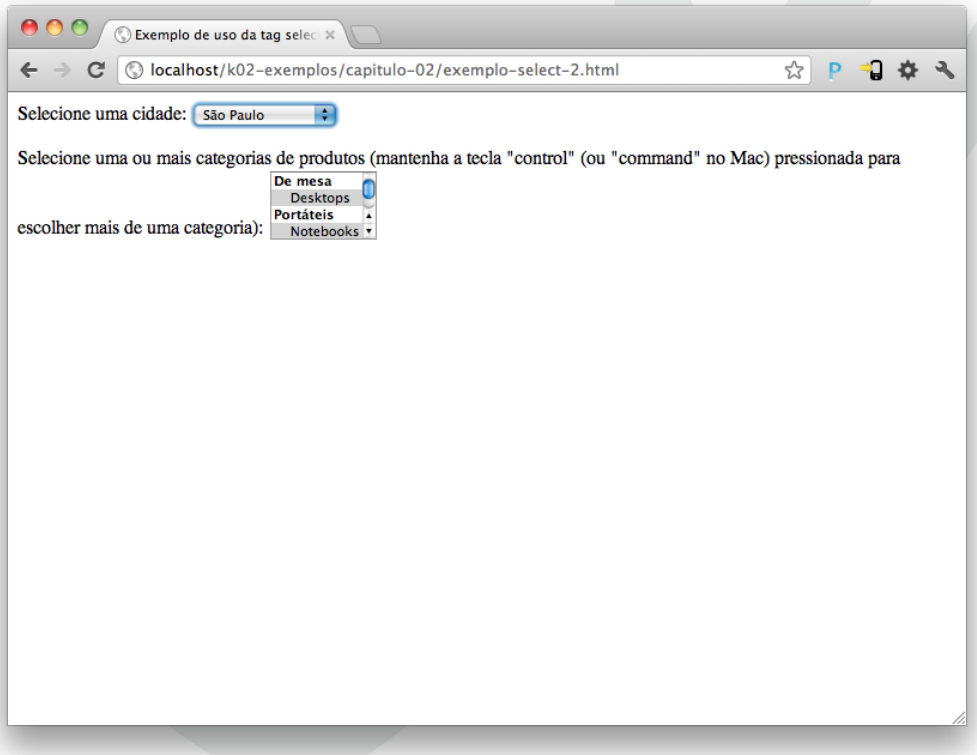

Figura 2.13: Exemplo de uso da tag `select`

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.

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

```
11            </textarea>
12          </p>
13        </form>
14      </body>
15  </html>
```

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

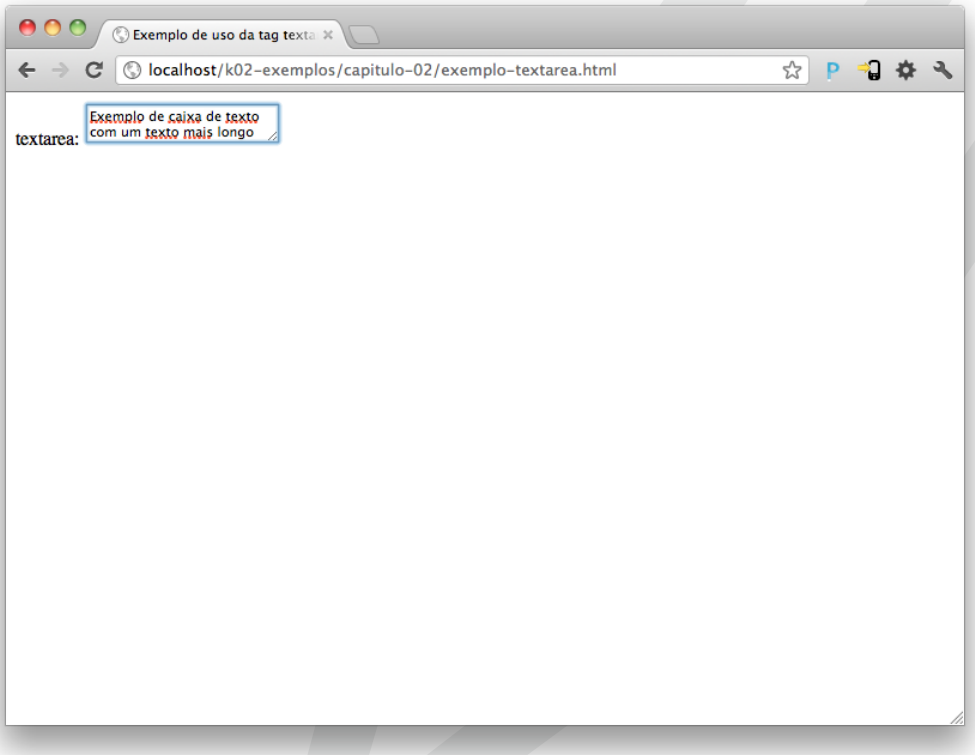

Figura 2.14: Exemplo de uso da tag `textarea`

Exercícios de Fixação

14 Crie um documento HTML com formulário que contenha uma caixa de texto que aceite mais de uma linha.

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

Código HTML 2.34: Exemplo de uso da tag `label`

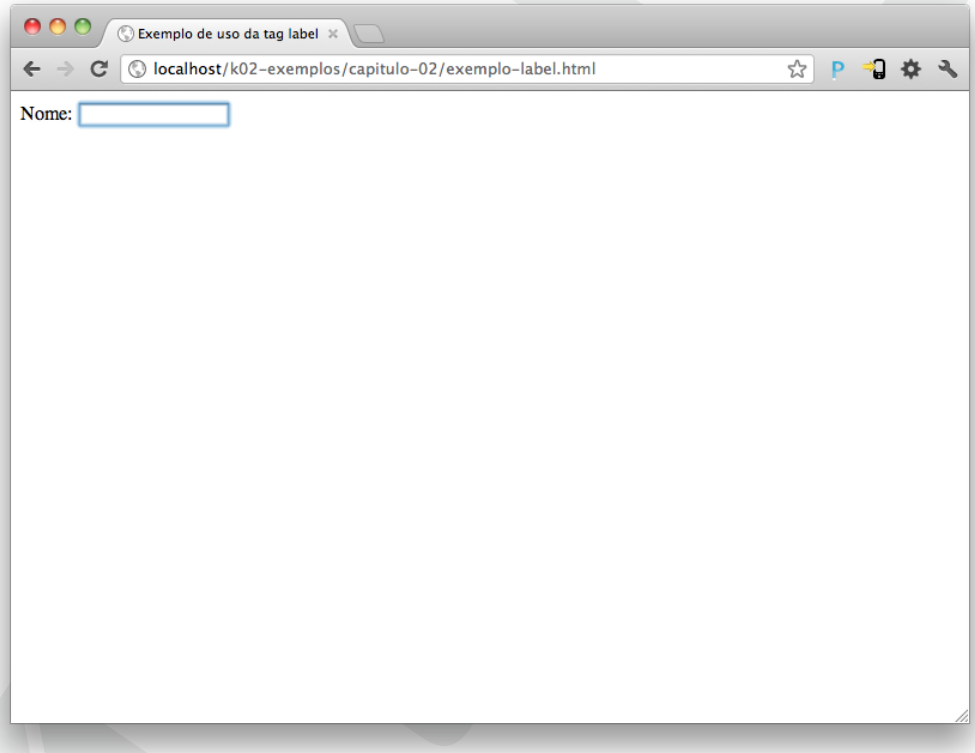

Figura 2.15: Exemplo de uso da tag `label`

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

Exercícios de Fixação

15 Crie um documento HTML com diversos elementos de formulário e para cada elemento crie um rótulo. Cada rótulo deve focar o elemento de formulário correspondente.

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 indideros 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).

CSS

CAPÍTULO 3

Até o momento trabalhamos apenas com os elementos HTML sem nos preocuparmos com a questão visual dos mesmos. Durante os exemplos mostrados no capítulo anterior, cada elemento estava utilizando a formatação padrão fornecida pelo navegador.

A formatação padrão pode variar de navegador para navegador, pois apesar de todos os navegadores tentarem seguir as sugestões do W3C, às vezes ocorrem erros de interpretação da especificação ou erros de implementação. Além disso, o W3C sugere, ou seja, não obriga. Portanto, seria 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 cairam 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

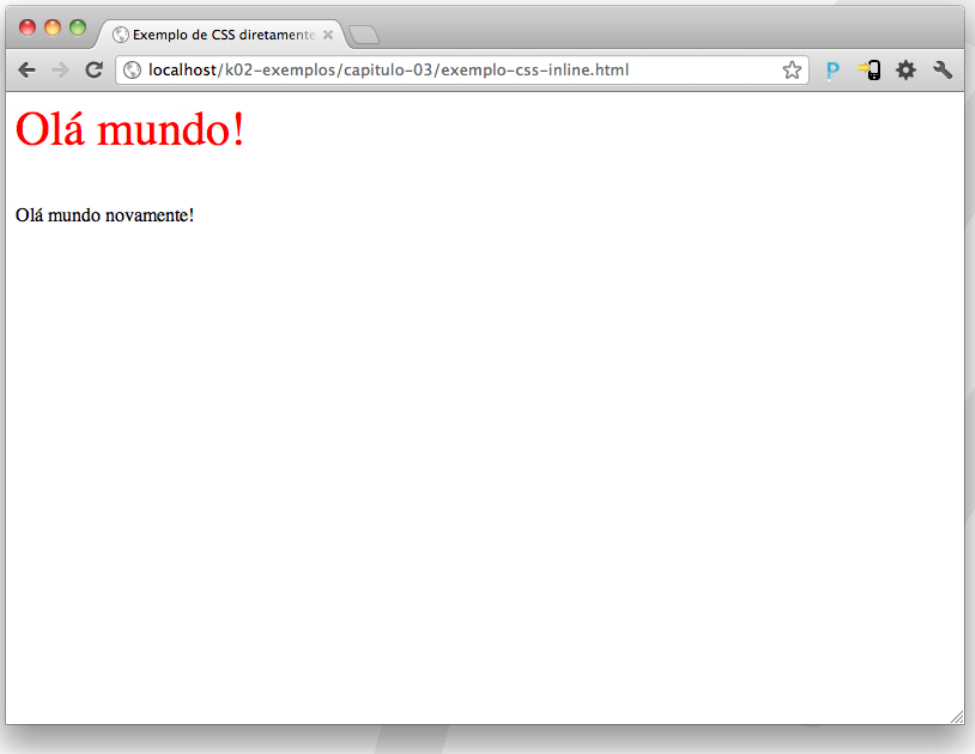

Figura 3.1: Exemplo de aplicação das propriedades CSS inline

Quando utilizamos as propriedades CSS diretamente em um elemento, dizemos que aplicando o *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`

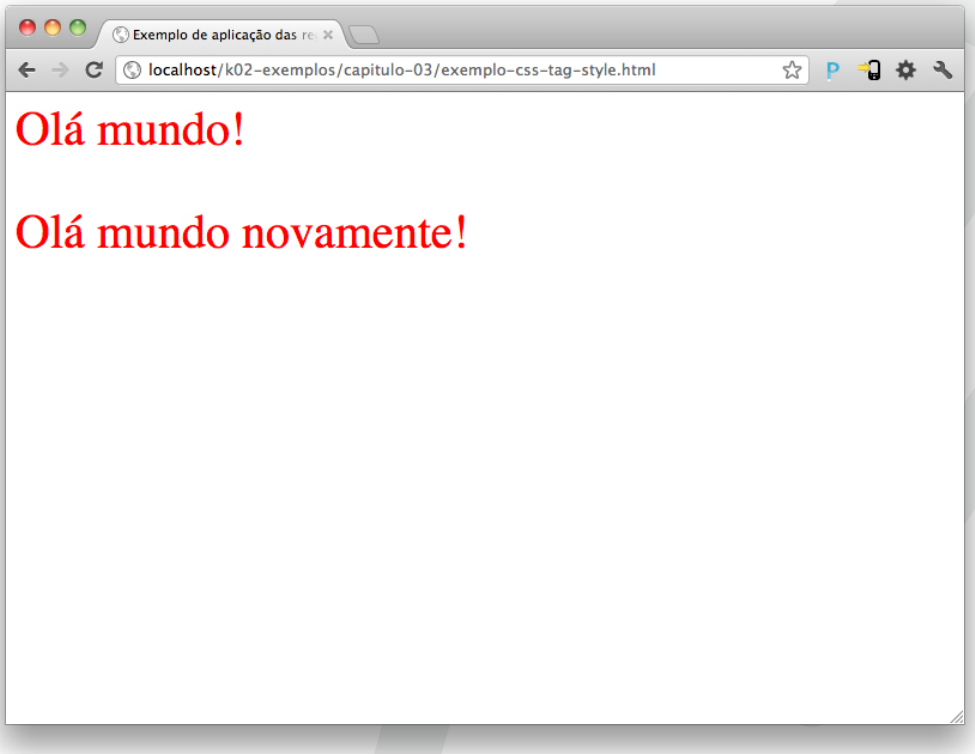

Figura 3.2: Exemplo de aplicação das regras CSS através da tag `style`

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:

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

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

```
p {
  font-size: 40px;
  color: #ff0000;
}
```

Código CSS 3.1: estilo.css

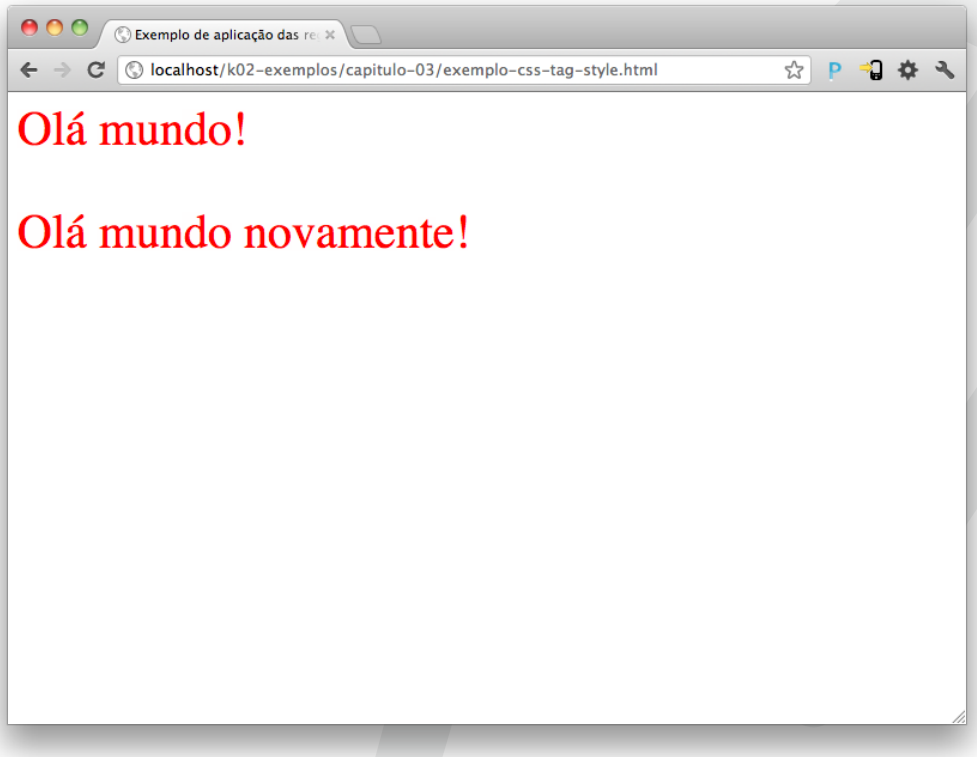

Figura 3.3: Exemplo de aplicação das regras CSS através de um arquivo

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.

Estrutura de uma regra CSS

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

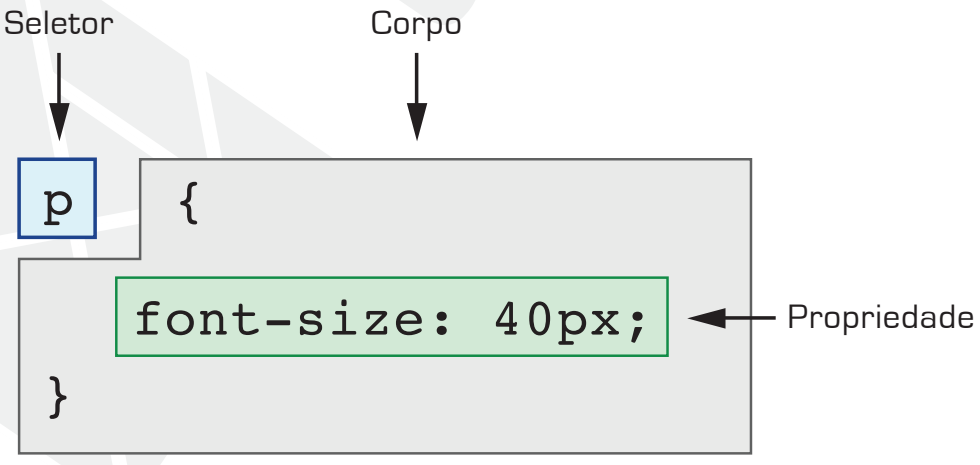

Figura 3.4: Estrutura de uma regra CSS

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`.

Como no exemplo acima utilizamos aquilo que chamamos de *seletor de tipo*, todos os elementos com a tag `p` receberão as propriedades definidas no corpo da regra.

Tipos de seletores

No CSS temos definidos sete tipos de seletores que podem ser utilizados sozinhos ou em conjunto:

- 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 iremos fazer com que todos os elementos tenham margem igual a `0px`.

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

Código CSS 3.2: Usando o seletor universal

Seletor de tipo

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

No exemplo a seguir iremos selecionar todos os elementos que possuem a tag `textarea`.

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

Código CSS 3.3: 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 3.4: 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 3.5: 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 3.6: 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 3.7: 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 3.8: 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 3.9: 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 `titulos`. Repare que o seletor de classe começa com o caractere `.` (ponto).

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

Código CSS 3.10: Usando o seletor de classe

Exercícios de Fixação

1 Crie uma nova página com diversos elementos HTML. Altere a formatação dos elementos utilizando os seletores que você acabou de aprender. Tente usar todos os seletores.

Principais propiedades 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 3.11: 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 3.12: 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 3.13: 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 prorpiedades 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;
}

ol {
```

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

Código CSS 3.14: Propriedades de lista

Propriedades de tabela

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

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

Código CSS 3.15: 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 3.16: Propriedades de dimensão

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`).

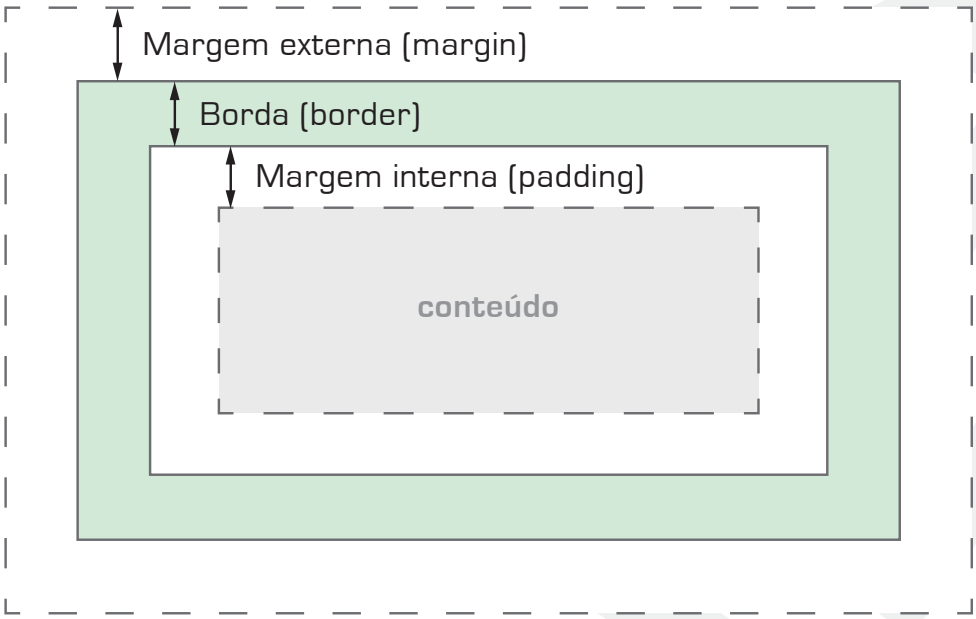

Figura 3.5: Box model

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 3.17: 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:

```
1 div {
2   width: 258px;
3   padding: 10px;
4   margin: 10px;
5   border: 1px solid green;
6 }
```

Código CSS 3.18: Exemplo de uso correto das dimensões de um elemento

Posicionando elementos

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

- 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 idéia 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.

Cores em CSS

Em uma propriedade CSS na qual que devemos atribuir uma cor, podemos utilizar três formas diferentes de se escrever esse valor: nome da cor, valor hexadecimal ou RGB.

Nem todas as cores possuem um nome e é por esse motivo que normalmente utilizamos a forma hexadecimal ou RGB na grande maioria das vezes.

Uma cor heaxadecimal é definida da seguinte forma: `#RRGGBB`, na qual `RR`, `GG` e `BB` devem variar de `00` a `FF` e representam os componentes vermelho, verde e azul de uma cor.

Para definirmos uma cor utilizando a notação RGB devemos utilizar a função `rgb(R, G, B)` substituindo as letras R, G e B por valores de `0` a `255` ou por uma porcentagem de `0%` a `100%`.

Em CSS3 ainda possuimos mais três formas: RGBA, HSL e HSLA.

O HSL define as cores através da matiz, saturação e luminosidade (hue, saturation e lightness). Devemos utilizar a função `hsl(H, S, L)`, na qual H pode variar de `0` a `360` e S e L de `0%` a `100%`.

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

Unidades de medida

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

- in - polegada.
- cm - centrímentro.
- 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`.

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 3.6: 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  int valor = 30;
2  boolean 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 com o nome **javascript**. Crie um documento HTML com um código JavaScript que imprima na página o seu nome 100 vezes. Salve o documento na pasta **javascript** e em seguida abra o arquivo no navegador.

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

Código Javascript 4.11: imprime-nome.js

2 Crie um documento HTML com um código JavaScript que imprima na página os números de 1 até 100. Salve o documento na pasta **javascript** e em seguida abra o arquivo no navegador.

```
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 com um código JavaScript que percorra todos os número 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 o documento na pasta **javascript** e em seguida abra o arquivo no navegador.

```
for(var contador = 1; contador <= 100; contador++) {
  var resto = contador % 2;
  if(resto == 1) {
    document.writeln('*');
  } else {
    document.writeln('**');
  }

  document.writeln('<br />');
}
```

Código Javascript 4.13: imprime-padrao-1.js

4 Crie um documento HTML com um código 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 o documento na pasta **javascript** e em seguida abra o arquivo no navegador.

```
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 o documento na pasta **javascript** e em seguida abra o arquivo no navegador.

Exercícios Complementares

1 Crie um documento HTML com um código JavaScript que imprima na página um triângulo de "*". Veja o exemplo abaixo:

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

2 Crie um documento HTML com um código JavaScript que imprima na tela vários triângulos de "*". Observe o padrão abaixo.

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

3 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 com um código 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 4.18: Definindo uma função

Definindo uma função que recebe parâmetros:

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

Código Javascript 4.19: Definindo uma função que recebe parâmetros

Definindo uma função que retorna um valor:

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

Código Javascript 4.20: 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 4.21: Chamando uma função

Chamando uma função que recebe parâmetros:

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

Código Javascript 4.22: Chamando uma função que recebe parâmetros

Chamando uma função que retorna um valor:

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

Código Javascript 4.23: 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.

```
1  <html>
2      <head>
3        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4          <title>Chamando uma função JavaScript pelo HTML</title>
5          <script type="text/javascript">
6            function exibeOlaMundo() {
7          alert('Olá Mundo!');
8        }
9          </script>
10     </head>
11     <body>
12         <input type="button" value="Exibe saudação" onclick="exibeOlaMundo()" />
13     </body>
14 </html>
```

Código HTML 4.2: 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:

```
1  var contaBancaria = new Object();
2
3  contaBancaria.numero = 1234;
4  contaBancaria.saldo = 1000;
5
6  contaBancaria.deposita = function(valor) {
7    if(valor > 0) {
8      this.saldo += valor;
9    }
10   else {
11     alert('Valor inválido!');
12   }
13 };
```

Código Javascript 4.24: 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) {
      this.saldo += valor;
    }
    else {
      alert('Valor inválido!');
    }
  }
}
```

Código Javascript 4.25: 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.

```
var numeros = new Array();
numeros[0] = 34;
numeros[1] = 52;
numeros[2] = 83;

var nomes = new Array('Jonas', 'Rafael', 'Marcelo');

var cidades = ['São Paulo', 'Rio de Janeiro', 'Curitiba'];
```

Código Javascript 4.26: 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 com um código 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.

6 Crie um documento HTML com um código 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.

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 idéia 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 prótotipo 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 prótotipo 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".

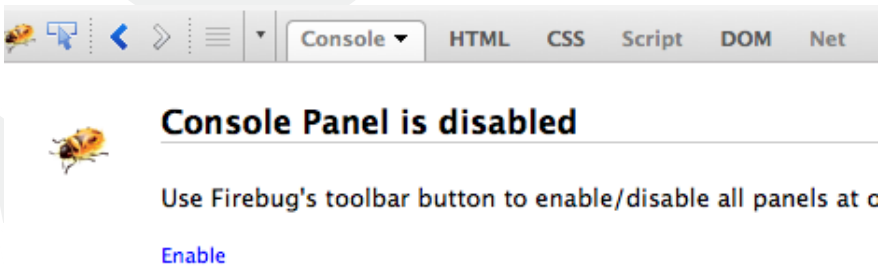

Figura A.1: Habilitando o console

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.

```
var curso = {sigla: "K11", nome: "Orientação a Objetos em Java"};
console.log(curso.sigla);
console.log(curso.nome);

var curso2 = {sigla: "K12", nome: "Desenvolvimento Web com JSF2 e JPA2"};
console.log(curso2.sigla);
console.log(curso2.nome);
```

Código Javascript A.15: Criando dois objetos

3 Verifique o funcionamento das referências em JavaScript.

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

// imprime K11
console.log(curso.sigla);

// imprime Orientação a Objetos em Java
console.log(curso.nome);

var x = curso;

x.sigla = "K12";
x.nome = "Desenvolvimento Web com JSF2 e JPA2";

// imprime K12
console.log(curso.sigla);

// imprime Desenvolvimento Web com JSF2 e JPA2
console.log(curso.nome);
```

Código Javascript A.16: Referências

4 Crie um objeto a partir de outro objeto existente.

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

var novo_curso = Object.create(curso);

// imprime K11
console.log(novo_curso.sigla);

// imprime Orientação a Objetos em Java
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.

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

var novo_curso = Object.create(curso);

curso.carga_horaria = 36;

// imprime K11
console.log(novo_curso.sigla);

// imprime Orientação a Objetos em Java
console.log(novo_curso.nome);

// imprime 36
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 afetato.

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

var novo_curso = Object.create(curso);

novo_curso.carga_horaria = 36;

// imprime K11
console.log(curso.sigla);

// imprime Orientação a Objetos em Java
console.log(curso.nome);

// imprime undefined
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.

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

var novo_curso = Object.create(curso);

// imprime K11
console.log(novo_curso.sigla);

// imprime Orientação a Objetos em Java
console.log(novo_curso.nome);

curso.sigla = "K12";
curso.nome = "Desenvolvimento Web com JSF2 e JPA2";

// imprime K12
console.log(novo_curso.sigla);

// imprime Desenvolvimento Web com JSF2 e JPA2
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.

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

var novo_curso = Object.create(curso);

novo_curso.sigla = "K12";
novo_curso.nome = "Desenvolvimento Web com JSF2 e JPA2";

// imprime K12
console.log(novo_curso.sigla);

// imprime Desenvolvimento Web com JSF2 e JPA2
console.log(novo_curso.nome);

curso.sigla = "K21";
curso.nome = "Persistência com JPA2 e Hibernate";

// imprime K12
console.log(novo_curso.sigla);

// imprime Desenvolvimento Web com JSF2 e JPA2
console.log(novo_curso.nome);
```

Código Javascript A.21: Protótipo

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.

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

Código Javascript A.22: Criando uma função

Utilizando uma Função

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

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

Código Javascript A.23: 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.

```
1  var conta = {
2    saldo: 0,
3    deposita: function(valor) {
4      this.saldo += valor;
5    }
6  }
7
8  conta.deposita(100);
9  console.log(conta.saldo);
```

Código Javascript A.24: Método

Apply

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

```
1  var deposita = function(valor) {
2    this.saldo += valor;
3  }
4
5  var conta = {
6    saldo: 0
7  }
8
9  deposita.apply(conta, [200]);
10 console.log(conta.saldo);
```

Código Javascript A.25: 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.

```
1  var soma = function() {
2    var soma = 0;
3
4    for(var i = 0; i < arguments.length; i ++) {
5      soma += arguments[i];
6    }
7
8    return soma;
9  }
10
11 var resultado = soma(2,4,5,6,1);
12
13 console.log(resultado);
```

Código Javascript A.26: 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.27: 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.28: 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.29: multiplicacao()

10 Execute a função `multiplicacao()`

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

Código Javascript A.30: 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.31: deposita()

12 Execute o método `deposita()`.

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

Código Javascript A.32: 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.33: soma()

14

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

console.log(resultado);
```

Código Javascript A.34: 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.35: Exceptions

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

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

Código Javascript A.36: Executando o método deposita()

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

```
try {
```

```
2   conta.deposita(0);
3 } catch(e) {
4   console.log(e.name);
5   console.log(e.message);
6 }
```

Código Javascript A.37: Capturanto exceções com `try-catch`

Arrays

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

```
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.38: Criando um array

Percorrendo um Array

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

```
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.39: for

Adicionando Elementos

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

```
1 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
2 cursos[cursos.length] = "K01";
3 for(var i = 0; i < cursos.length; i++) {
4   console.log(cursos[i]);
5 }
```

Código Javascript A.40: Adicionando elementos ao final do array com length

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

```
1 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
2 cursos.push("K01");
3 for(var i = 0; i < cursos.length; i++) {
4   console.log(cursos[i]);
5 }
```

Código Javascript A.41: 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.42: 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.43: 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.44: Concatenando

Gerando uma String com os Elementos de um Array

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

```
var formacao_java = ["K11","K12"];

var resultado = formacao_java.join(",");
```

```
4 console.log(resultado);
```

Código Javascript A.45: 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.46: 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.47: 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.48: 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.49: 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.50: 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.51: 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.52: Adicionando um elemento na primeira posição

Métodos das Strings

Acessando os Caracteres de uma String por Posição

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

```
1 var curso = "K12";
2
3 console.log(curso.charAt(0));
```

Código Javascript A.53: 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.54: 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.55: 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.56: Criando dois arrays e imprimindo o tamanho

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.57: for

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

```
1 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
2
3 cursos[cursos.length] = "K01";
4
5 for(var i = 0; i < cursos.length; i++) {
6   console.log(cursos[i]);
7 }
```

Código Javascript A.58: Adicionando elementos ao final do array com length

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

```
1 var cursos = ["K11","K12","K21","K22","K23", "K31", "K32"];
2 cursos.push("K01");
3 for(var i = 0; i < cursos.length; i++) {
4   console.log(cursos[i]);
5 }
```

Código Javascript A.59: Adicionando elementos através do método push

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

```
1 var formacao_java = ["K11","K12"];
2 var formacao_java_avancado = ["K21","K22","K23"];
3
4 var formacao_completa = formacao_java.concat(formacao_java_avancado);
5
6 for(var i = 0; i < formacao_completa.length; i++) {
7   console.log(formacao_completa[i]);
8 }
```

Código Javascript A.60: Concatenando

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

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

Código Javascript A.61: Removendo o último elemento

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

```
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.62: Adicionando um elemento na última posição

22

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

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

cursos.reverse();

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

Código Javascript A.63: Invertendo os elementos de um array

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

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

var curso = cursos.shift();

console.log("Elemento removido: " + curso);

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

Código Javascript A.64: Removendo o primeiro elemento

Faça uma cópia de um determinado trecho de um array.

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

var formacao_java = cursos.slice(0,2);

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

Código Javascript A.65: Copiando um trecho de um array

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

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

cursos.unshift("K11");

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

Código Javascript A.66: Adicionando um elemento na primeira posição

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

```
var curso = "K12-Desenvolvimento Web com JSF2 e JPA2";
var aux = curso.split("-");

console.log(aux[0]);
console.log(aux[1]);
```

Código Javascript A.67: Dividindo uma string

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 provêem 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>
</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".

```
<!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"/>
  <script>$('.curso').val('K19');</script>
</body>
</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 (-)`.

```
<!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"/>
  <script>$('input[name|="formacao"]').val('K19');</script>
</body>
</html>
```

Código HTML B.6: curso.html

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

```
<!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"/>
  <script>$('input').val('K19');</script>
</body>
</html>
```

Código HTML B.7: curso.html

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.8: 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
`$(seletor).dblclick(função de callback...)`	A função de callback será chamda quando o usuário clicar 2X (duas) vezes nos elementos selecionados
`$(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.9: 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.10: 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>
```

```
10        <li>K31 - C# e Orientação a Objetos</li>
11        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12        <li>K11 - Java e Orientação a Objetos</li>
13        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14      </ul>
15    </div>
16   <script>
17    $('li').dblclick(function(){ alert("Elemento clicado 2X:"+$(this).text());});
18   </script>
19  </body>
20  </html>
```

Código HTML B.11: eventos.html

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

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <script src="http://code.jquery.com/jquery-latest.js"></script>
5     <script>
6     $(document).ready(function () {
7       $('li').text('DOM carregado por completo.');
8     });
9     </script>
10  </head>
11  <body>
12    <h1>Cursos K19</h1>
13    <div>
14      <ul>
15        <li>K31 - C# e Orientação a Objetos</li>
16        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
17        <li>K11 - Java e Orientação a Objetos</li>
18        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
19      </ul>
20    </div>
21  </body>
22  </html>
```

Código HTML B.12: 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.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <script src="http://code.jquery.com/jquery-latest.js"></script>
5   </head>
6   <body>
7     <h1>Cursos K19</h1>
8     <div>
9       <label for="cursos">Selecione um curso:</label>
10      <select name="cursos" id="cursos">
11              <option>----</option>
12        <option value="K31">K31 - C# e Orientação a Objetos</option>
13        <option value="K32">K32 - Desenvolvimento Web com ASP .NET MVC</option>
14        <option value="K11">K11 - Java e Orientação a Objetos</option>
15        <option value="K12">K12 - Desenvolvimento Web com JSF2 e JPA2</option>
16      </select>
17    </div>
18    <script>
19      $('#cursos').change(function(){
20        alert("Curso selecionado: "+$("#cursos option:selected").text());
21      });
```

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

Código HTML B.13: eventos.html

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`.

```
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="mostrar">Mostrar</button>
11    <button id="ocultar">Ocultar</button>
12
13    <p style="display: none">Cursos K19</p>
14  <script>
15      $("#mostrar").click(function () {
16      $("p").show("slow");
17      });
18      $("#ocultar").click(function () {
19      $("p").hide("slow");
20      });
21      </script>
22  </body>
23  </html>
```

Código HTML B.14: 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.

```
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>Mostrar/Ocultar</button>
11
12    <p style="display: none">Cursos K19</p>
13  <script>
14      $("button").click(function () {
15      $("p").toggle("slow");
16      });
17      </script>
18  </body>
19  </html>
```

Código HTML B.15: 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 milisegundos.

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.

```
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="fadein">Fade in</button>
11    <button id="fadeout">Fade out</button>
12    <button id="fadeto">Fade to</button>
13    <p>Cursos K19</p>
14  <script>
15      $("#fadein").click(function () {
16      $("p").fadeIn();
17      });
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.16: 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 milisegundos.

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

```
<!DOCTYPE html>
<html>
<head>
  <style>
      p { background:green; }
      </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
  <button id="slidedown">Slide Down</button>
  <button id="slideup">Slide Up</button>
  <button id="slidetoggle">Slide Toggle</button>
  <p>Cursos K19</p>
<script>
    $("#slidedown").click(function () {
    $("p").slideDown();
    });
    $("#slideup").click(function () {
    $("p").slideUp();
    });
    $("#slidetoggle").click(function () {
    $("p").slideToggle("slow");
    });
</script>
</body>
</html>
```

Código HTML B.17: 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>
```

```
7   <h1>Cursos K19</h1>
8   <div>
9     <ul>
10      <li>K31 - C# e Orientação a Objetos</li>
11      <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12      <li>K11 - Java e Orientação a Objetos</li>
13      <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14    </ul>
15  </div>
16 </body>
17 </html>
```

Código HTML B.18: efeitos.html

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

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <script src="http://code.jquery.com/jquery-latest.js"></script>
5  </head>
6  <body>
7    <h1>Cursos K19</h1>
8    <div>
9      <ul>
10       <li>K31 - C# e Orientação a Objetos</li>
11       <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12       <li>K11 - Java e Orientação a Objetos</li>
13       <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14     </ul>
15   </div>
16   <div>
17     <button id="mostrar">Mostrar</button>
18     <button id="ocultar">Ocultar</button>
19   </div>
20   <script>
21     $("#mostrar").click(function(){ $("li").show("slow");});
22     $("#ocultar").click(function(){ $("li").hide("slow");});
23   </script>
24 </body>
25 </html>
```

Código HTML B.19: efeitos.html

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

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <script src="http://code.jquery.com/jquery-latest.js"></script>
5  </head>
6  <body>
7    <h1>Cursos K19</h1>
8    <div>
9      <ul>
10       <li>K31 - C# e Orientação a Objetos</li>
11       <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12       <li>K11 - Java e Orientação a Objetos</li>
13       <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14     </ul>
15   </div>
16   <div>
17     <button id="mostrar-ocultar">Mostrar/Ocultar</button>
18   </div>
```

```
19    <script>
20      $("#mostrar-ocultar").click(function(){ $("li").toggle("slow"); });
21    </script>
22  </body>
23  </html>
```

Código HTML B.20: efeitos.html

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

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <script src="http://code.jquery.com/jquery-latest.js"></script>
5   </head>
6   <body>
7     <h1>Cursos K19</h1>
8     <div>
9       <ul>
10        <li>K31 - C# e Orientação a Objetos</li>
11        <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12        <li>K11 - Java e Orientação a Objetos</li>
13        <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14      </ul>
15    </div>
16    <div>
17      <button id="mostrar">Mostrar</button>
18      <button id="ocultar">Ocultar</button>
19    </div>
20    <script>
21    $("#mostrar").click(function(){ $("li").fadeIn("slow"); });
22    $("#ocultar").click(function(){ $("li").fadeOut("slow"); });
23    </script>
24  </body>
25  </html>
```

Código HTML B.21: efeitos.html

Exercícios Complementares

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

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

3 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.

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()`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
      p { background:green; }
      </style>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
  <button id="prepend">Prepend</button>
  <button id="append">Append</button>
  <p>Cursos K19</p>
<script>
    $("#prepend").click(function () {
      $("p").prepend("K19 Treinamentos - ");
    });
    $("#append").click(function () {
      $("p").append(" - K31");
    });
</script>
</body>
</html>
```

Código HTML B.25: 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.26: 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.27: 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.28: 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");});
```

```
18    </script>
19  </body>
20  </html>
```

Código HTML B.29: html.html

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

```
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    <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.30: html.html

RESPOSTAS

APÊNDICE C

Resposta do Exercício 2.5

```
<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>

    <p>Utilize um recipiente que permita o macarrão ficar totalmente submerso
    na água.</p>
    <p>Quando ouvir o bip não saia correndo. O microondas não irá explodir,
    pois o bip significa que o macarrão está pronto.</p>

    <h3>No fogão</h3>

    <p>Ferva a água em uma panela.</p>
    <p>Insira o macarrão e cozinhe-o por 3 minutos</p>

    <h4>Ponto importante</h4>

    <p>Utilize uma panela que permita o macarrão ficar totalmente submerso
    na água.</p>
    <p>Não se distraia com a televisão ou qualquer outra coisa. Você poderá
    queimar a sua refeição</p>
  </body>
</html>
```

Código HTML 2.9: Resposta do exercício

Resposta do Exercício 2.6

```
<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 2.12: Resposta do exercício - 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 2.13: Resposta do exercício - pagina2.html

Resposta do Exercício 2.7

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Exercício sobre âncoras</title>
    </head>
    <body>
        <p><a href="#sobre">Sobre o texto dessa página</a></p>

        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
            massa, sodales sit amet eleifend a, elementum eu nibh. Donec egestas dolor
            quis turpis dictum tincidunt. Donec blandit tempus velit, sit amet
            adipiscing velit consequat placerat. Curabitur id mauris facilisis dui
            iaculis auctor quis vel lacus. Vestibulum ante ipsum primis in faucibus
            orci luctus et ultrices posuere cubilia Curae; Ut auctor diam in magna
            feugiat in varius ligula faucibus. Suspendisse tempor mi nec sem fermentum
            malesuada. In sit amet enim vel leo bibendum cursus. Curabitur nec velit
            at nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra. Duis vel
            tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
            accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>

        <p>Donec dictum sem ut orci ornare ultrices. Cras blandit nibh sed eros
            suscipit in feugiat nunc tincidunt. Praesent semper lorem sed ipsum ↩
                placerat
            porta. In arcu massa, dignissim ut elementum nec, luctus nec sem. Nam ut
            purus urna. Class aptent taciti sociosqu ad litora torquent per conubia
            nostra, per inceptos himenaeos. Curabitur fermentum dapibus ullamcorper.
            Vivamus ante tellus, facilisis vitae interdum eget, mollis eget ipsum.</p>

        <p>Praesent dapibus risus eu quam egestas ultricies. Nunc sed arcu purus.
            Integer vehicula, nisl sit amet tincidunt accumsan, nisi felis venenatis ↩
                ante,
            non auctor felis tellus tempor quam. Pellentesque laoreet feugiat lacus ac
            convallis. Vestibulum quis elementum eros. Quisque convallis, justo nec
            congue consequat, nisl felis dapibus mi, vel facilisis risus enim ut ↩
                turpis.
            Nam rhoncus turpis at nibh vulputate nec placerat nibh aliquam.</p>
```

```
34
35          <p>Aenean vestibulum purus eget nunc varius tempor. Nulla placerat suscipit mi
36              fermentum mattis. Aenean id feugiat eros. Donec tristique libero nec ↩
                    sapien
37              eleifend vel  vulputate nibh facilisis. Pellentesque dolor massa, ↩
                    convallis eu
38              lobortis sed, fermentum eu lectus. Maecenas eget metus tellus, non dictum
39              erat. Sed aliquam lobortis nunc, in consectetur risus aliquet et. Fusce ↩
                    quam
40              arcu, tincidunt quis pellentesque sit amet, tempus eu nulla. Pellentesque
41              ante diam, t ristique a dictum nec, rutrum sed mauris. Cum sociis natoque
42              penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer
43              nunc  sapien, bibendum et mattis sit amet, pulvinar ut mi. Pellentesque
44              habitant morbi tristique senectus et netus et malesuada fames ac turpis
45              egestas. Nunc aliquet libero sed dui euismod sodales.</p>
46
47          <a name="sobre">Sobre o texto dessa página</a>
48
49          <p>
50              O texto dessa página foi gerado através do site:
51              <a href="http://www.lipsum.com/">http://www.lipsum.com/</a>
52          </p>
53      </body>
54  </html>
```

Código HTML 2.15: Resposta do exercicio

Resposta do Exercício 2.8

```
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="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 facilisis dui
14              iaculis auctor quis vel lacus. Vestibulum ante ipsum primis in faucibus
15              orci luctus et ultrices posuere cubilia Curae; Ut auctor diam in magna
16              feugiat in varius ligula faucibus. Suspendisse tempor mi nec sem fermentum
17              malesuada. In sit amet enim vel leo bibendum cursus. Curabitur nec velit
18              at nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra. Duis vel
19              tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
20              accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>
21
22          <p>Donec dictum sem ut orci ornare ultrices. Cras blandit nibh sed eros
23              suscipit in feugiat nunc tincidunt. Praesent semper lorem sed ipsum ↩
                    placerat
24              porta. In arcu massa, dignissim ut elementum nec, luctus nec sem. Nam ut
25              purus urna. Class aptent taciti sociosqu ad litora torquent per conubia
26              nostra, per inceptos himenaeos. Curabitur fermentum dapibus ullamcorper.
27              Vivamus ante tellus, facilisis vitae interdum eget, mollis eget ipsum.</p>
28
29          <p>Praesent dapibus risus eu quam egestas ultricies. Nunc sed arcu purus.
30              Integer vehicula, nisl sit amet tincidunt accumsan, nisi felis venenatis ↩
                    ante,
31              non auctor felis tellus tempor quam. Pellentesque laoreet feugiat lacus ac
32              convallis. Vestibulum quis elementum eros. Quisque convallis, justo nec
33              congue consequat, nisl felis dapibus mi, vel facilisis risus enim ut ↩
                    turpis.
```

```
            Nam rhoncus turpis at nibh vulputate nec placerat nibh aliquam.</p>

        <p>Aenean vestibulum purus eget nunc varius tempor. Nulla placerat suscipit mi
            fermentum mattis. Aenean id feugiat eros. Donec tristique libero nec ↩
                sapien
            eleifend vel  vulputate nibh facilisis. Pellentesque dolor massa, ↩
                convallis eu
            lobortis sed, fermentum eu lectus. Maecenas eget metus tellus, non dictum
            erat. Sed aliquam lobortis nunc, in consectetur risus aliquet et. Fusce ↩
                quam
            arcu, tincidunt quis pellentesque sit amet, tempus eu nulla. Pellentesque
            ante diam, t ristique a dictum nec, rutrum sed mauris. Cum sociis natoque
            penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer
            nunc  sapien, bibendum et mattis sit amet, pulvinar ut mi. Pellentesque
            habitant morbi tristique senectus et netus et malesuada fames ac turpis
            egestas. Nunc aliquet libero sed dui euismod sodales.</p>

        <a name="sobre">Sobre o texto dessa página</a>

        <p>
            O texto dessa página foi gerado através do site:
            <a href="http://www.lipsum.com/">http://www.lipsum.com/</a>
        </p>
    </body>
</html>
```

Código HTML 2.16: Resposta do exercicio

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Exercício sobre âncoras</title>
    </head>
    <body>
        <h1>Página 2</h1>

        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec justo
            massa, sodales sit amet eleifend a, elementum eu nibh. Donec egestas dolor
            quis turpis dictum tincidunt. Donec blandit tempus velit, sit amet
            adipiscing velit consequat placerat. Curabitur id mauris facilisis dui
            iaculis auctor quis vel lacus. Vestibulum ante ipsum primis in faucibus
            orci luctus et ultrices posuere cubilia Curae; Ut auctor diam in magna
            feugiat in varius ligula faucibus. Suspendisse tempor mi nec sem fermentum
            malesuada. In sit amet enim vel leo bibendum cursus. Curabitur nec velit
            at nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra. Duis vel
            tristique tellus. Maecenas ultrices placerat tortor. Pellentesque feugiat
            accumsan commodo. Proin non urna justo, id pulvinar lacus.</p>

        <p>Donec dictum sem ut orci ornare ultrices. Cras blandit nibh sed eros
            suscipit in feugiat nunc tincidunt. Praesent semper lorem sed ipsum ↩
                placerat
            porta. In arcu massa, dignissim ut elementum nec, luctus nec sem. Nam ut
            purus urna. Class aptent taciti sociosqu ad litora torquent per conubia
            nostra, per inceptos himenaeos. Curabitur fermentum dapibus ullamcorper.
            Vivamus ante tellus, facilisis vitae interdum eget, mollis eget ipsum.</p>

        <p>Praesent dapibus risus eu quam egestas ultricies. Nunc sed arcu purus.
            Integer vehicula, nisl sit amet tincidunt accumsan, nisi felis venenatis ↩
                ante,
            non auctor felis tellus tempor quam. Pellentesque laoreet feugiat lacus ac
            convallis. Vestibulum quis elementum eros. Quisque convallis, justo nec
            congue consequat, nisl felis dapibus mi, vel facilisis risus enim ut ↩
                turpis.
            Nam rhoncus turpis at nibh vulputate nec placerat nibh aliquam.</p>

        <p>Aenean vestibulum purus eget nunc varius tempor. Nulla placerat suscipit mi
            fermentum mattis. Aenean id feugiat eros. Donec tristique libero nec ↩
                sapien
            eleifend vel  vulputate nibh facilisis. Pellentesque dolor massa, ↩
```

```
                convallis eu
38              lobortis sed, fermentum eu lectus. Maecenas eget metus tellus, non dictum
39              erat. Sed aliquam lobortis nunc, in consectetur risus aliquet et. Fusce ↩
                    quam
40              arcu, tincidunt quis pellentesque sit amet, tempus eu nulla. Pellentesque
41              ante diam, t ristique a dictum nec, rutrum sed mauris. Cum sociis natoque
42              penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer
43              nunc  sapien, bibendum et mattis sit amet, pulvinar ut mi. Pellentesque
44              habitant morbi tristique senectus et netus et malesuada fames ac turpis
45              egestas. Nunc aliquet libero sed dui euismod sodales.</p>
46
47          <a name="outra_ancora">Mais uma âncora</a>
48
49          <p>Se você chegou aqui deu tudo certo! :)</p>
50      </body>
51  </html>
```

Código HTML 2.17: Resposta do exercicio - pagina2.html

Resposta do Exercício 2.9

```
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          <p>
32            <img src="http://www.k19.com.br/css/img/k21-logo-large.png" />
33            K21 - Persistência com JPA2 e Hibernate
34          </p>
35          <p>
36            <img src="http://www.k19.com.br/css/img/k22-logo-large.png" />
37            K22 - Desenvolvimento Web Avançado com JSF2, EJB3.1 e CDI
38          </p>
39          <p>
40            <img src="http://www.k19.com.br/css/img/k23-logo-large.png" />
41            K23 - Integração de Sistemas com Webservices, JMS e EJB
42          </p>
43      </body>
44  </html>
```

Código HTML 2.19: Resposta do exercício

Resposta do Exercício 2.10

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Exercício para a tag table</title>
    </head>
    <body>
        <table>
            <thead>
                <tr>
                    <th>Marca</th>
                    <th>Modelo</th>
                    <th>Ano</th>
                </tr>
            </thead>
            <tfoot>
                <tr>
                    <td colspan="3">Última atualização: 06/2012</td>
                </tr>
            </tfoot>
            <tbody>
                <tr>
                    <td rowspan="2">Toyota</td>
                    <td>Corolla</td>
                    <td>2010</td>
                </tr>
                <tr>
                    <td>Camry</td>
                    <td>2011</td>
                </tr>

                <tr>
                    <td rowspan="3">Honda</td>
                    <td>Civic</td>
                    <td>2004</td>
                </tr>
                <tr>
                    <td>Fit</td>
                    <td>2012</td>
                </tr>
                <tr>
                    <td>City</td>
                    <td>2011</td>
                </tr>

                <tr>
                    <td>Mitsubishi</td>
                    <td>Lancer</td>
                    <td>2012</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

Código HTML 2.22: Resposta do exercício

Resposta do Exercício 2.11

```html
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Menu - K19 Pizzaria</title>
    </head>
    <body>
      <dl>
        <h1>K19 Pizzaria</h1>
        <h2>Menu</h2>

        <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>
      </dl>
    </body>
</html>
```

Código HTML 2.24: Resposta do exercício

Resposta do Exercício 2.12

```html
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Operação de saque - K19 Bank</title>
    </head>
    <body>
      <dl>
        <h1>K19 Bank</h1>
        <h2>Manual do caixa eletrônico</h2>
        <h3>Operação de saque</h3>

        <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>
      </dl>
    </body>
</html>
```

Código HTML 2.26: Resposta do exercício

Resposta do Exercício 2.13

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>K00 - Formação Básica - K19 Treinamentos</title>
    </head>
    <body>
      <dl>
        <h1>K19 Treinamentos</h1>
        <h2>K00 - Formação Básica</h2>

        <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 2.28: Resposta do exercício

Resposta do Exercício 2.14

```
<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 2.33: Resposta do exercício

Resposta do Exercício 2.15

```
<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:
```

```
18          <input type="radio" name="sexo" id="masculino" />
19          <label for="masculino">Masculino</label>
20          <input type="radio" name="sexo" id="feminino" />
21          <label for="feminino">Feminino</label>
22        </p>
23        <p>
24          <label for="mensagem">Mensagem:</label>
25          <textarea id="mensagem"></textarea>
26        </p>
27      </form>
28    </body>
29  </html>
```

Código HTML 2.35: Resposta do exercício

Resposta do Complementar 4.1

```
1  var linha = '*';
2  for(var contador = 1; contador <= 10; contador++) {
3    document.writeln(linha);
4    document.writeln('<br />');
5    linha += '*';
6  }
```

Código Javascript 4.15: imprime-padrao-3.js

Resposta do Complementar 4.2

```
1  var linha = '*';
2  for(var contador = 1; contador <= 10; 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 4.16: imprime-padrao-4.js

Resposta do Complementar 4.3

```
1  var penultimo = 0;
2  var ultimo = 1;
3
4  document.writeln(penultimo);
5  document.writeln(ultimo);
6
7  for(var contador = 0; contador < 28; contador++) {
8    var proximo = penultimo + ultimo;
9    document.writeln(proximo);
10   document.writeln('<br />');
11
```

```
12    penultimo = ultimo;
13    ultimo = proximo;
14  }
```

Código Javascript 4.17: imprime-padrao-5.js

Resposta do Exercício 4.5

```
1  var array = new Array();
2
3  for(var i = 0; i < array.length; i++){
4    array[i] = i;
5  }
6
7  for(var i = 0; i < array.length; i++){
8    document.writeln(array[i]);
9    document.writeln('<br />');
10 }
11
12 for(var i = 0; i < 10; i++){
13   var posicao1 = Math.floor(Math.random()*11);
14   var posicao2 = Math.floor(Math.random()*11);
15   var auxiliar = array[posicao1];
16
17   array[posicao1] = array[posicao2];
18   array[posicao2] = auxiliar;
19 }
20
21 document.writeln("--------------------");
22 document.writeln('<br />');
23
24 for(var i = 0; i < array.length; i++){
25   document.writeln(array[i]);
26   document.writeln('<br />');
27 }
```

Código Javascript 4.27: Resposta do exercício

Resposta do Exercício 4.6

```
1  var array = new Array();
2
3  for(var i = 0; i < array.length; i++){
4    array[i] = Math.floor(Math.random()*101);
5  }
6
7  for(var i = 0; i < array.length; i++){
8    document.writeln(array[i]);
9  }
10
11 array.sort();
12
13 document.writeln("--------------------");
14
15 for(var i = 0; i < array.length; i++){
16   document.writeln(array[i]);
17 }
```

Código Javascript 4.28: Resposta do exercício

Resposta do Complementar B.1

```
<!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>
    <button id="fadeto">FadeTo</button>
  </div>
  <script>
    $("#mostrar").click(function(){ $("li").fadeIn("slow"); });
    $("#ocultar").click(function(){ $("li").fadeOut("slow"); });
    $("#fadeto").click(function(){ $("li").fadeTo("slow", 0.2); });
  </script>
</body>
</html>
```

Código HTML B.22: efeitos.html

Resposta do Complementar B.2

```
<!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").slideDown("slow"); });
    $("#ocultar").click(function(){ $("li").slideUp("slow"); });
  </script>
</body>
</html>
```

Código HTML B.23: efeitos.html

Resposta do Complementar B.3

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <script src="http://code.jquery.com/jquery-latest.js"></script>
5  </head>
6  <body>
7    <h1>Cursos K19</h1>
8    <div>
9      <ul>
10       <li>K31 - C# e Orientação a Objetos</li>
11       <li>K32 - Desenvolvimento Web com ASP .NET MVC</li>
12       <li>K11 - Java e Orientação a Objetos</li>
13       <li>K12 - Desenvolvimento Web com JSF2 e JPA2</li>
14     </ul>
15   </div>
16   <div>
17     <button id="mostrar-ocultar">Mostrar/Ocultar</button>
18   </div>
19   <script>
20     $("#mostrar-ocultar").click(function(){ $("li").slideToggle("slow"); });
21   </script>
22 </body>
23 </html>
```

Código HTML B.24: efeitos.html