

```
<div>
</div>
</div>

<section id="services">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 text-center">
        <h2 class="section-heading">All Your Services</h2>
        <br class="primary">
      </div>
    </div>
    <div class="container">
      <div class="row">
        <div class="col-lg-3 col-md-6 text-center">
          <div class="service-box">
            <i class="fa fa-flask text-primary on-focus"></i>
            <h3>Sturdy Templates</h3>
            <p class="text-muted">Our templates are updated regularly so they don't break</p>
          </div>
        </div>
        <div class="col-lg-3 col-md-6 text-center">
          <div class="service-box">
            <i class="fa fa-leaf fa-paper-plane text-primary on-focus"></i>
            <h3>Ready to Ship</h3>
            <p class="text-muted">You can use this theme as is, or you can start designing</p>
          </div>
        </div>
        <div class="col-lg-3 col-md-6 text-center">
          <div class="service-box">
            <i class="fa fa-newspaper-o text-primary on-focus"></i>
            <h3>Up to Date</h3>
            <p class="text-muted">We update dependencies to keep things fresh</p>
          </div>
        </div>
        <div class="col-lg-3 col-md-6 text-center">
          <div class="service-box">
            <i class="fa fa-heart text-primary on-focus"></i>
            <h3>Made with Love</h3>
            <p class="text-muted">You have to make your website with love these themes</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

<section class="p-9" id="portfolio">
  <div class="container-fluid">
    <div class="row no-gutter portfolio-gallery">
      <div class="col-lg-4 col-sm-6">
        <a class="portfolio-box" href="img/portfolio/fullsize/1.jpg">
          
          <div class="portfolio-box-caption">
            <div class="portfolio-box-caption-content">
              <div>
```

DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS E WEB I

ILUMNO



DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS E WEB I

ILUMNO

Copyright © UVA 2019

Nenhuma parte desta publicação pode ser reproduzida por qualquer meio sem a prévia autorização desta instituição.

Texto de acordo com as normas do Novo Acordo Ortográfico da Língua Portuguesa.

AUTORIA DO CONTEÚDO

Carlos Augusto Sicsú Ayres
do Nascimento

PROJETO GRÁFICO

UVA

REVISÃO

Clarissa Penna
Theo Cavalcanti
Lydianna Lima

DIAGRAMAÇÃO

UVA

N244

Nascimento, Carlos Augusto Sicsú Ayres do

Desenvolvimento de aplicações distribuídas e Web I [livro eletrônico] / Carlos Augusto Sicsú Ayres do Nascimento. – Rio de Janeiro: UVA, 2019.

6,4 MB.

ISBN 978-85-5459-075-8

1. HTML (Linguagem de programação de computador). 2. Android (Recurso eletrônico). 3. Computação móvel. 4. Software de aplicação - Desenvolvimento. 5. JavaScript (Linguagem de programação de computador). 6. Folhas de estilo. I. Universidade Veiga de Almeida. II. Título.

CDD - 005.268

Bibliotecária Katia Cavalheiro CRB 7 - 4826.
Ficha Catalográfica elaborada pela Biblioteca Central da UVA.

SUMÁRIO

Apresentação	6
Autor	7
UNIDADE 1	
Introdução ao HTML	8
• Introdução à arquitetura cliente <i>versus</i> servidor; browser (navegadores) e protocolos Web	
• Estrutura de uma página HTML e suas principais tags	
• Criação de sites de conteúdo	
UNIDADE 2	
HTML avançado com desenvolvimento de formulários	36
• Introdução ao desenvolvimento de formulários para a Web	
• As principais tags HTML para a criação de formulários para a Web	
• <i>Front-ends</i> para aplicações Web	

SUMÁRIO

UNIDADE 3

Padronização de sites com uso de CSS

76

- Conceitos fundamentais para a aplicação de folhas de estilos (CSS), definição de classes, seletores, unidades de medidas e cores
- Padronização da estrutura da página (caixas, margens, bordas, espaçamento, posição e cursores)
- Padronização de textos (com definição de fontes, cores e fundos, padronização de imagens e tabelas) e criação de sites padronizados

UNIDADE 4

Introdução ao desenvolvimento de aplicações Web com JavaScript

118

- Estrutura de aplicações com programação com JavaScript e tipos de dados e expressões
- Estruturas de controle de fluxo
- Desenvolvimento de aplicações com o uso de programação com JavaScript

APRESENTAÇÃO

O desenvolvimento de aplicações distribuídas na Web se divide em dois momentos distintos baseados na arquitetura cliente versus servidor. O primeiro momento compreende o entendimento do usuário (front-end), utilizando o layout e suas interações. O segundo momento abarca o desenvolvimento da parte servidor, ou back-end, que abrange a estrutura de serviços e servidores que dão suporte às aplicações para a Web.

Nesta disciplina, abordaremos o desenvolvimento da parte cliente, ou seja, a interface com o usuário, conhecida como *front-end*. Nosso alvo é o desenvolvimento da interface com o usuário para a criação de aplicações para a Web, envolvendo a criação de sites padronizados com o uso de HTML e CSS, além da programação JavaScript, para execução local do *front-end*.

AUTOR

PROF. CARLOS AUGUSTO SICSÚ AYRES DO NASCIMENTO, DSC.

Possui curso técnico em Edificações pelo Centro Federal de Educação Tecnológica Celso Suckow da Fonseca – Cefet-RJ, formado em Matemática Aplicada à Informática e em Engenharia da Computação. Possui três pós-graduações, sendo elas: Especialista em Engenharia de Computação, pela Universidade do Estado do Rio de Janeiro – Uerj, mestre em Engenharia de Computação, também pela Uerj, doutor em Computação de Alto Desempenho pelo Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia – Coppe-UFRJ.

Tem experiência de mais de 20 anos no magistério, em cursos presenciais e EAD; trabalhou na área de TI da indústria de embalagens durante dois anos e atualmente dedica-se ao desenvolvimento de projetos de pesquisa na área de aplicativos para dispositivos móveis.

REVISÃO E ATUALIZAÇÃO

IGOR GONZALEZ PIMENTA

Possui graduação em Bacharelado em Sistema de Informação – Faculdade Ruy Barbosa-BA. Possui uma pós-graduações em Engenharia de Software, mestre em Ciência e Sistema da Computação, pela UNIFACS-BA.

Tem experiência de mais de 8 anos no magistério, em cursos presenciais e EAD; trabalha na área de TI a mais de 15 anos e atualmente dedica-se ao desenvolvimento de projetos de TI área de supermercados e varejos.

UNIDADE 1

Introdução ao HTML

INTRODUÇÃO

A arquitetura cliente *versus* servidor é altamente empregada no desenvolvimento de aplicações, sejam elas para o desenvolvimento de aplicações desktop, para a Web ou ainda para o desenvolvimento de aplicações para dispositivos móveis. A sua compreensão visa entender como ocorrem as trocas de informações entre as chamadas aplicações clientes e aplicações servidoras.

Os sistemas para a Web são amplamente baseados em aplicações executadas diretamente em navegadores (browsers), e o HTML continua sendo a linguagem essencial para a criação de sites e aplicações. Compreender sua estrutura é fundamental para o desenvolvimento de soluções front-end modernas, que não apenas oferecem experiências de usuário mais ricas e interativas, mas também garantem a integração fluida com arquiteturas cliente-servidor. Essas arquiteturas permitem que o HTML atue como a base para a comunicação entre o front-end e os serviços de back-end, atendendo às crescentes demandas de organizações e da sociedade por aplicações Web de alta qualidade e escalabilidade.



OBJETIVO

Nesta unidade você será capaz de:

- Desenvolver sites de conteúdo, a partir do conhecimento da estrutura básica de páginas HTML e da aplicação das principais tags HTML.

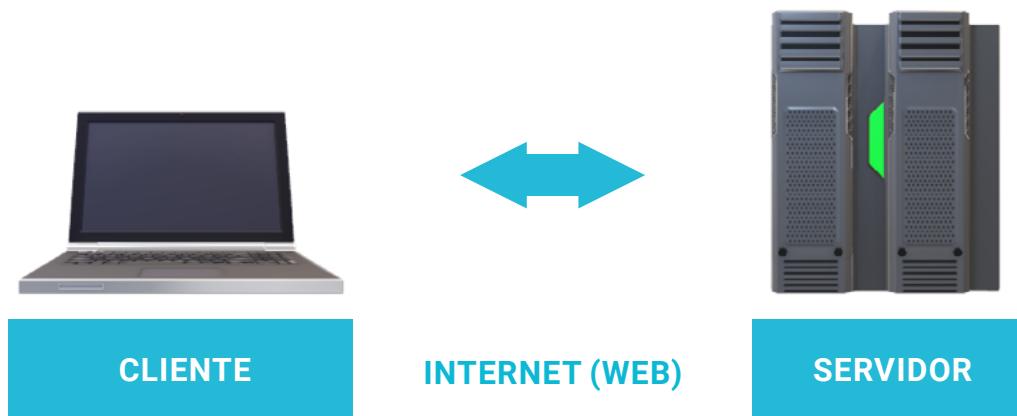
Introdução à arquitetura cliente versus servidor; browser(navegadores) e protocolos Web

Arquitetura cliente versus servidor

A internet é hoje o principal meio de comunicação utilizado no mundo. A sua arquitetura se divide em duas partes:

- Uma parte chamada cliente, em que os usuários interagem por meio de navegadores ou aplicações Web, incluindo as chamadas aplicações móveis ou desktops.
- Outra parte chamada de servidor, que dá suporte às aplicações clientes e é responsável pela troca de mensagens com elas, além de acesso de bancos de dados e regra de negócio dos sistemas.

Basicamente, a arquitetura cliente versus servidor pode ser facilmente entendida conforme a figura a seguir:



Na estrutura cliente, temos a interface com o usuário, chamada *front-end*; normalmente são aplicações construídas com a aplicação de componentes de uso local, tais como as linguagens de marcação HTML e CSS, além de programação por meio de linguagens de script, tais como JavaScript ou VBScript, sendo mais comum o uso de JavaScript. Essas aplicações são responsáveis pela interação do usuário com o sistema servidor e visam à entrada e crítica dos dados incluídos pelos usuários, sendo executadas normalmente por meio dos chamados navegadores da internet.

A internet é uma estrutura de suporte às comunicações entre os diferentes dispositivos, permitindo que os dados e informações fluam em ambos os sentidos entre as aplicações clientes e as aplicações servidores. Possui uma série de equipamentos de rede, tais como roteadores, switches, entre outros. Possui, ainda, infraestrutura para suporte à comunicação, tais como cabos de rede, fibras óticas, entre outros. Para a perfeita comunicação entre diferentes tipos de equipamentos, ainda é necessária uma série de protocolos de comunicação ou de rede, responsáveis por controlar e confirmar o envio e recebimento das mensagens trafegadas pela Web por meio das aplicações, tanto clientes como servidores.

A estrutura básica é fácil de entender, mas a Web é muito mais abrangente e possui uma quantidade interminável de equipamentos interligados entre si, incluindo os dispositivos móveis, tais como smartphones e tablets.

Na estrutura servidor, temos os chamados servidores de aplicações, que são máquinas com grande poder computacional, responsáveis por responder a requisições dos usuários emanadas das aplicações clientes por meio da execução de códigos e acesso a gerenciadores de bancos de dados. Esses códigos formam aplicações, e essas aplicações são chamadas de aplicações servidoras ou de serviços. Essas aplicações são comumente criadas utilizando linguagens de programação mais abrangentes do que as linguagens de script empregadas nas aplicações clientes. As linguagens de programação mais utilizadas para a criação de aplicações servidoras são: PHP, Java, Python, Ruby, C#, entre outras.

Na figura a seguir temos um pequeno exemplo, com o emprego de alguns poucos equipamentos clientes e servidores, mas a internet possui uma quantidade imensa de equipamentos interligados simultaneamente.



A internet, por meio de satélites e cabos de fibra ótica, é responsável por interligar o mundo, permitindo que as pessoas possam interagir a partir de qualquer lugar.



MEDIATECA

Acesse a midiateca da Unidade 1 e veja o conteúdo complementar indicado pelo professor sobre a arquitetura cliente versus servidor.



Saiba mais

Para saber mais sobre definição de sistema, leia o Capítulo 1 (p. 3-8) do seguinte livro:

MILETTO, E.; BERTAGNOLLI, S. **Desenvolvimento de software II**: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. Minha Biblioteca.

Navegadores para Web (browsers)

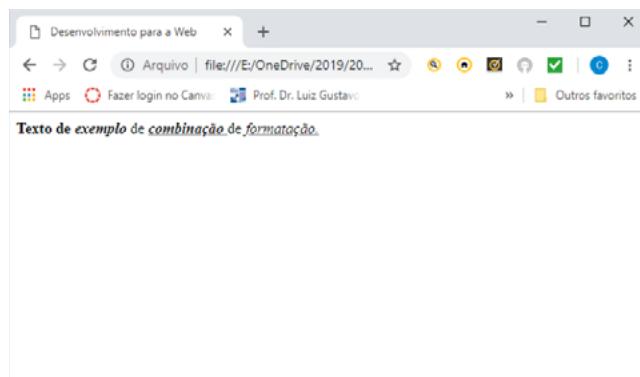
A internet é utilizada por bilhões de pessoas todos os dias, e uma das formas mais comuns é a utilização de navegadores. O navegador é um tipo de aplicativo capaz de carregar arquivos de conteúdo contendo textos a serem exibidos ao usuário, mas com uma série de marcas incorporadas ao texto contendo marcas que são interpretadas pelo navegador e que são responsáveis por configurar a forma como o conteúdo será exibido.

Essas marcas são chamadas de tags e são delimitadas pelos símbolos: < ... >. Cada marca ou tag determina uma ação específica para a formatação do conteúdo a ser exibido pelo navegador. A função do navegador, então, é carregar o conteúdo do arquivo, contendo a página a ser exibida, extrair as tags e formatar o texto a ser exibido de acordo com a formatação determinada. São exemplos de tags:

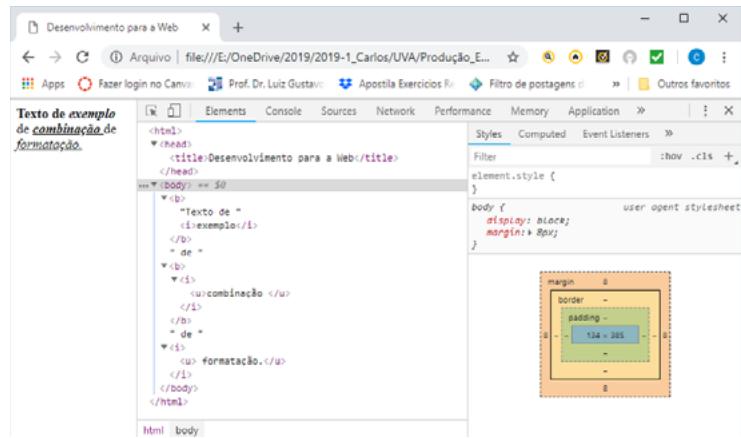
- <p> Parágrafo </p>, que indica um parágrafo.
-
, que pula uma linha.
- Itálico , que formata um texto em itálico.

Essas tags são definidas a partir de uma linguagem de marcação chamada HTML (*hyper-text markup language*, ou seja, “linguagem de marcação de hipertexto”). Atualmente os navegadores estão evoluindo para atender aos requisitos da versão 5.0 do HTML, que é uma evolução da versão 4.0, além de incorporar melhorias apresentadas pelo XHTML e o HTML DOM, mas é esperado que, em breve, os navegadores passem a dar suporte ao HTML 5.1. Essa evolução foi necessária quando saímos da chamada Web 1.0, que era capaz de apresentar conteúdos estáticos, para a chamada Web 2.0, que passou a incorporar conteúdo dinâmico, permitindo, dessa forma, maior capacidade de interação dos usuários com os websites.

A seguir, podemos analisar uma página simples em HTML sendo formatada e exibida pelo Google Chrome®.



A maioria dos principais navegadores para a web nos permite observar o conteúdo completo do código da página, bastando clicar sobre a tecla F12. Sendo assim, o conteúdo original que deu origem à página acima pode ser visto na próxima figura. Esse é um dos principais fatores para a utilização da arquitetura cliente versus servidor, pois a codificação na parte cliente é de fácil acesso, o que acaba por tornar o conteúdo inseguro para tratar senhas e acessos a gerenciadores de bancos de dados.



Alguns dos navegadores mais conhecidos e utilizados atualmente são: Chrome, Internet Explorer, Firefox (antigo Netscape), Edge, entre outros.



Saiba mais

Para saber mais sobre compatibilidade do HTML 5, leia o Capítulo 4 do seguinte livro:

FLATSCHART, F. **HTML 5:** embarque imediato. Rio de Janeiro: Brasport, 2018.
Biblioteca Virtual.

Protocolos Web

Como vimos anteriormente, a Web necessita de um padrão de utilização para a comunicação e troca de dados e informações entre as aplicações clientes e as aplicações servidores. As regras que determinam como essa comunicação irá ocorrer, garantindo a qualidade dessa comunicação, são determinadas por um conjunto de protocolos. Para a Web, o protocolo mais comum é o HTTP (hypertext transfer protocol, isto é, “protocolo de transferência de hipertexto”), que é o mais importante quando realizamos transferências de conteúdo, dados e informações pela internet. Além do HTTP, a Web ainda se utiliza de vários outros protocolos para atender às suas necessidades, dando ressalva para o protocolo HTTPS, na qual é uma evolução do HTTP, para garantir a segurança ao utilizar alguns Sites, exemplo disso são os sites de bancos. Segue outros protocolos a serem utilizados:

Protocolo	Finalidade
FTP (<i>file transfer protocol</i>)	Transferência de arquivos.
SSH (<i>secure shell</i>)	Acesso remoto seguro a um sistema.
Telnet (<i>remote login service</i>)	Acesso remoto.
SMTP (<i>simple mail transfer protocol</i>)	Padrão para envio de e-mails.
DNS (<i>domain name system</i>) service	Registro de máquinas (<i>hosts</i>) em um determinado domínio.
POP3 (<i>post office protocol</i>)	Acesso à caixa de correio eletrônico.
NNTP (<i>network news transfer protocol</i>)	Usado por grupos de discussão.
IMAP (<i>internet message access protocol</i>)	Gerenciamento de correio eletrônico.
SNNP (<i>simple network management protocol</i>)	Gerenciamento de dispositivos web baseados em endereço internet IP.
HTTPS (HTTP secure)	Protocolo HTTP seguro.
TCP (<i>transmission control protocol</i>)	Protocolo básico utilizado pela estrutura de suporte da Web para localização e intercomunicação dos dispositivos, também conhecido como TCP/IP.
UDP (<i>user datagram protocol</i>)	Permite o envio de um conjunto de dados a um destino.

Estrutura de uma página HTML e suas principais tags

Uma página HTML se divide basicamente em duas áreas distintas. Uma área de cabeçalho, onde são descritas informações de configurações para o navegador, além de dados contendo informações sobre a construção da página. Essa área é delimitada pelas tags de início de cabeçalho: <head> e de término de cabeçalho: </head>, onde todas as informações declaradas nessa área são de configurações para o navegador e de informações sobre a página. A outra área é de codificação (área de conteúdo ou corpo da página), iniciada pela tag: <body> e encerrada pela tag </body>.

Toda página HTML deve possuir inicialmente a tag de início de HTML (<html>) e de término (</html>). Dessa forma, uma página deve começar com uma estrutura mínima, tal como:

```
<html>
  <head>
    <!-- Cabeçalho da página -->
  </head>
  <body>
    <!-- Corpo da página -->
  </body>
</html>
```

<html> </html> – Início e término de codificação HTML.

<head> </head> – Início e término do cabeçalho da página.

<body> </body> – Início e término do corpo da página.

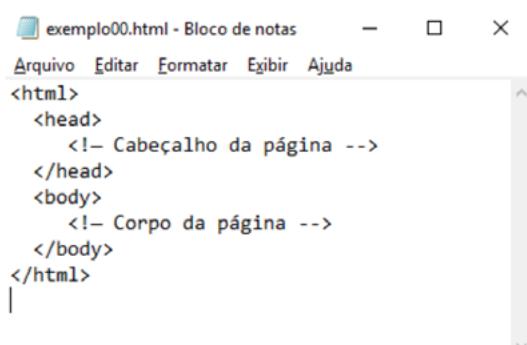
<!-- Este é um comentário -->

Obs.: Conteúdos dentro de comentários não são exibidos pelo navegador.

Essa página não apresentará nada, mas possui a estrutura mínima básica de uma página HTML.

Para a criação de uma página podemos utilizar um aplicativo muito simples e prático, o Bloco de Notas ou um editor de textos simples que não incorpore outras formas de formatação, tal como o Word e outros editores. Muitos desenvolvedores utilizam o Notepad++, Sublime ou Visual Code, como ferramentas de programação,

por apresentarem suporte à programação com uma formatação do código baseada em cores. Aqui utilizaremos apenas o Bloco de Notas para a edição e criação de nossas páginas, e o navegador utilizado nos testes será o Google Chrome®.



A screenshot of the Microsoft Notepad application window titled "exemplo00.html - Bloco de notas". The menu bar includes Arquivo, Editar, Formatar, Exibir, and Ajuda. The main content area displays the following HTML code:

```
<html>
  <head>
    <!-- Cabeçalho da página -->
  </head>
  <body>
    <!-- Corpo da página -->
  </body>
</html>
```

Para facilitar o desenvolvimento, podemos salvar o arquivo com o nome entre aspas, isso evita a inclusão de uma extensão indesejada:

Arquivo > Salvar Como: > "exemplo00.html"

Depois, basta abrir o arquivo com o navegador ou arrastar o arquivo e soltar sobre o navegador.

Não se esqueça de, sempre que você realizar alguma alteração, salvar o arquivo novamente e atualizar a página no navegador.

Vamos conhecer mais algumas tags para podermos, então, construir a página utilizada como exemplo anteriormente:

Para a formatação de cabeçalho (head)

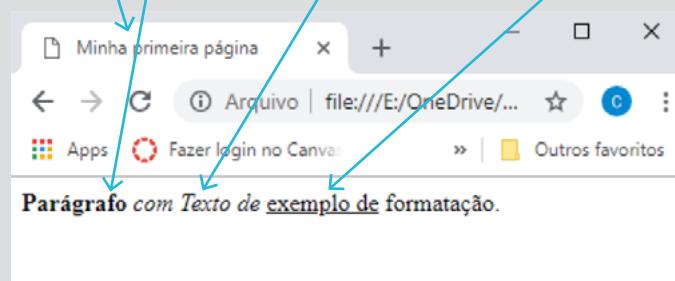
`<title>...</title>` – É utilizada para determinar o nome da página a ser exibido na aba ou área de título do navegador.

Para formatação de conteúdo (body):

- `<p> ... </p>` – É utilizado para delimitar um bloco de texto de conteúdo como um parágrafo. Exemplo: `<p>Esta linha é um parágrafo.</p>`
- ` ... ` – É utilizado para delimitar um bloco de texto que será exibido em negrito. Exemplo: `Texto negrito.`. Antigo (HTML4): ` ... ` – Apesar de estar em desuso, ainda é compatível.
- `<u> ... </u>` – É utilizado para delimitar um bloco de texto que será exibido sublinhado. Exemplo: `<u>Texto sublinhado.</u>`
- ` ... ` – É utilizado para delimitar um bloco de texto que será exibido em itálico. Exemplo: `Texto itálico.`. Antigo (HTML4): `<i> ... </i>` – Apesar de estar em desuso, ainda é compatível.

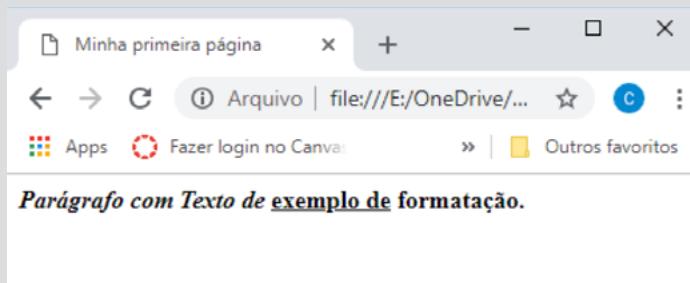
O exemplo a seguir apresenta uma página com o uso dessas tags:

```
<html>
  <head>
    <title>Minha primeira página</title>
  </head>
  <body>
    <p><strong>Parágrafo</strong><em>com Texto de </em><u>exemplo de
    </u> formatação.</p>
  </body>
</html>
```



Muitas tags podem ser combinadas (aninhadas), como podemos ver no exemplo a seguir:

```
<html>
  <head>
    <title>Minha primeira página</title>
  </head>
  <body>
    <p> <strong><em>Parágrafo com Texto de </em><u> exemplo de</u> formatação.</
    strong></p>
  </body>
</html>
```



Podemos observar que todo o parágrafo ficou em negrito; o texto [Parágrafo com Texto de], além de estar em negrito, também está em itálico (combinado); já o texto [exemplo de] está em negrito e sublinhado, mas não está em itálico; e, por fim, o texto [formatação.] encontra-se apenas em negrito. Dessa forma, podemos utilizar uma série de tags em conjunto para conseguirmos realizar a formatação ideal.

Para utilizarmos o HTML 5, nossa página deve incorporar como primeira tag: **<!DOCTYPE HTML>**.

Outro ponto importante é a definição da tabela de caracteres, ou CharSet, que será utilizada para definir o idioma da página criada. Podemos realizar essa definição por meio de um comando meta (responsável por disponibilizar dados e informações da página) pela tag: **<meta content="text/html; charset="utf-8">**, que define a página de caracteres que será utilizada para exibição no navegador. Ainda devemos definir a página de caracteres do idioma principal na tag inicial do HTML, usando **<html lang="pt-br">**, que define o idioma para português brasileiro.

Dessa forma, nossa última página criada ficará assim:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta content="text/html; charset="utf-8">
    <title>Minha primeira página</title>
  </head>
  <body>
    <p> <strong><em>Parágrafo com Texto de </em><u> exemplo de</u> for-
      matação.</strong></p>
  </body>
</html>
```



MEDIATECA

Acesse a midiateca da Unidade 1 e veja o conteúdo complementar indicado pelo professor sobre a criação da sua primeira página HTML utilizando o Notepad++.



Saiba mais

Para saber mais sobre compatibilidade do HTML 5, leia o Capítulo 5 do seguinte livro:

FLATSCHART, F. **HTML 5**: embarque imediato. Rio de Janeiro: Brasport, 2018.
Biblioteca Virtual.

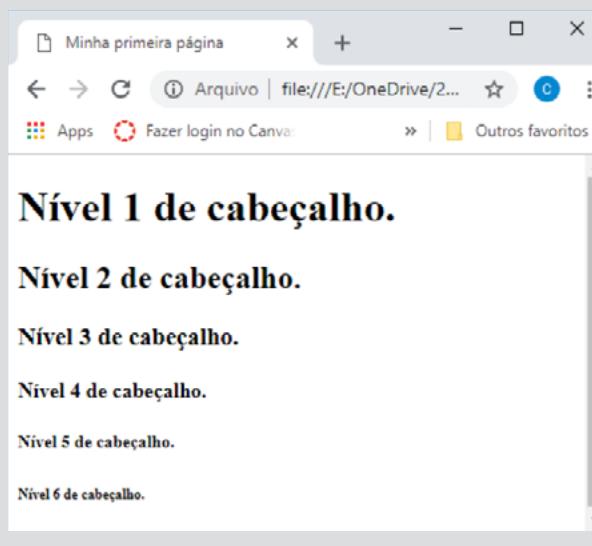
Criação de sites de conteúdo

Vamos conhecer mais algumas tags que serão muito úteis na criação de sites de conteúdo.

<h1> ... <h6> - Pré-formatação de cabeçalhos do conteúdo

Já existe um conjunto de tags definidas para determinar diferentes níveis de cabeçalhos, de acordo com a importância no texto. Essas tags de cabeçalhos predefinidos são numeradas de 1 a 6, conforme o exemplo a seguir (h1, h2, h3, h4, h5 e h6):

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Minha primeira página</title>
</head>
<body>
    <h1> Nível 1 de cabeçalho.</h1>
    <h2> Nível 2 de cabeçalho.</h2>
    <h3> Nível 3 de cabeçalho.</h3>
    <h4> Nível 4 de cabeçalho.</h4>
    <h5> Nível 5 de cabeçalho.</h5>
    <h6> Nível 6 de cabeçalho.</h6>
</body>
</html>
```

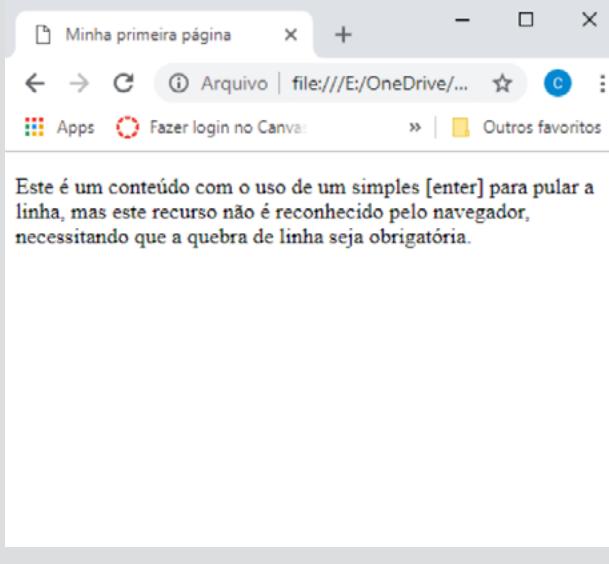


 - Quebra de linha

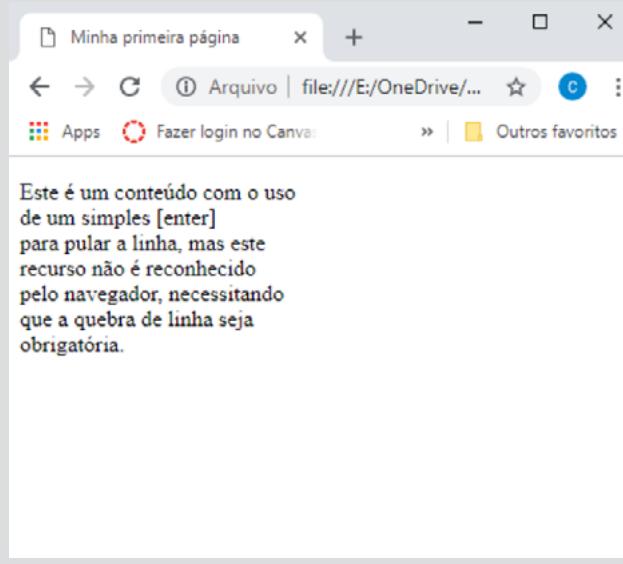
O uso de um simples [enter] não é capaz de encerrar uma linha e começar outra. Sendo assim, é necessário forçar que a linha seja encerrada para o início de outra.

Vamos comparar os exemplos a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta content="text/html; charset=utf-8">
    <title>Minha primeira página</title>
  </head>
  <body>
    <p>Este é um conteúdo com o
    uso de um simples [enter]
      para pular a linha, mas este recurso não é reconhecido
      pelo navegador, necessitando que a quebra de linha seja
      obrigatória.</p>
  </body>
</html>
```



```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Minha primeira página</title>
</head>
<body>
    <p>Este é um conteúdo com o<br>
    uso de um simples [enter]<br>
        para pular a linha, mas este<br>
    recurso não é reconhecido<br>
    pelo navegador, necessitando<br>
    que a quebra de linha seja<br>
    obrigatória.</p>
</body>
</html>
```



Observação

Repare que, ao alterar o tamanho lateral da janela no primeiro exemplo, as linhas mudam e, se o tamanho for suficiente, apenas uma linha será exibida. Já no segundo exemplo isso não ocorre, pois são obedecidos os comandos de quebra de linha.

<pre> ... </pre> - Texto pré-formatado

Como vimos anteriormente, o navegador não pula linha apenas por encontrar um [enter], assim como ele também não se importa com a quantidade de caracteres em branco digitados, apresentando apenas um. Dessa forma, os conteúdos a seguir são apresentados da mesma forma no navegador:

<p>Esta linha é um parágrafo</p> ou escrever: <pre> Esta linha é um parágrafo</pre>

Os espaços em branco no início do parágrafo não serão exibidos. Dessa forma, podemos utilizar a tag <pre>, que levará em conta os espaços, mas esse não é um recurso usado normalmente, uma vez que, caso o usuário altere a largura de visualização do navegador, esse texto pré-formatado não será ajustado ao novo tamanho. Isso também poderá fazer com que apareça uma barra de rolagem horizontal no caso de o texto não ser exibido em sua totalidade.

_{...} - Formata o texto com o estilo subscrito

Permite que uma parte do texto seja exibida em formato subscrito.

Exemplo: <p>H₂O.</p> Resultado: H₂O.

^{...} - Formata o texto com o estilo sobrescrito

Permite que uma parte do texto seja exibida em formato sobrescrito.

Exemplo: <p>100^oC.</p> Resultado: 100°C.

<hr> - Inclui uma linha horizontal

Permite criar uma separação entre conteúdos diferentes em uma mesma página.

Exemplo: <p>H₂O.</p>

<hr>

<p>100^oC.</p>

Resultado: $\frac{\text{H}_2\text{O}}{100^\circ\text{C}}$

Listas:

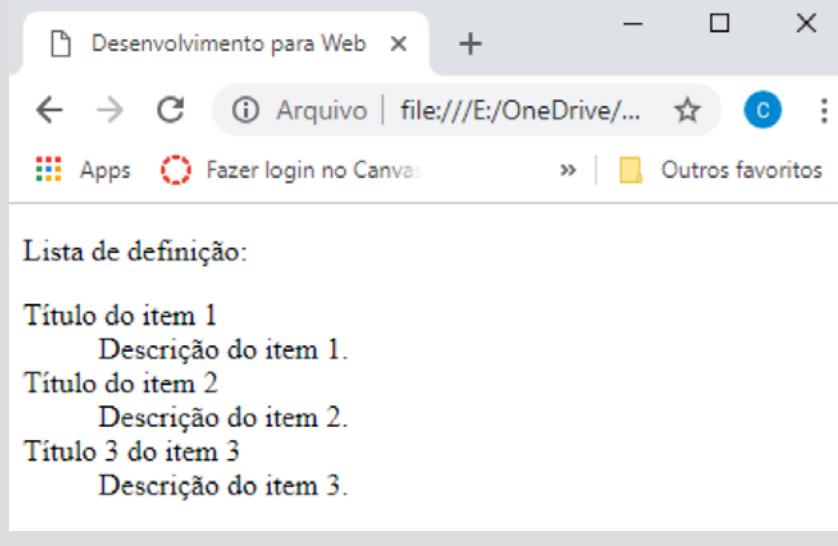
- Definição – dl
- Classificadas ou numeradas – ol
- Não classificadas – ul

<dl> ... </dl> - Define uma lista de definição

Permite a criação de uma lista com apresentação de itens e suas descrições.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Desenvolvimento para Web</title>
</head>
<body>
    <p>Lista de definição:</p>
<dl>
    <dt>Título do item 1</dt>
    <dd>Descrição do item 1.</dd>
    <dt>Título do item 2</dt>
    <dd>Descrição do item 2.</dd>
    <dt>Título 3 do item 3</dt>
    <dd>Descrição do item 3.</dd>
</dl>
</body>
</html>
```



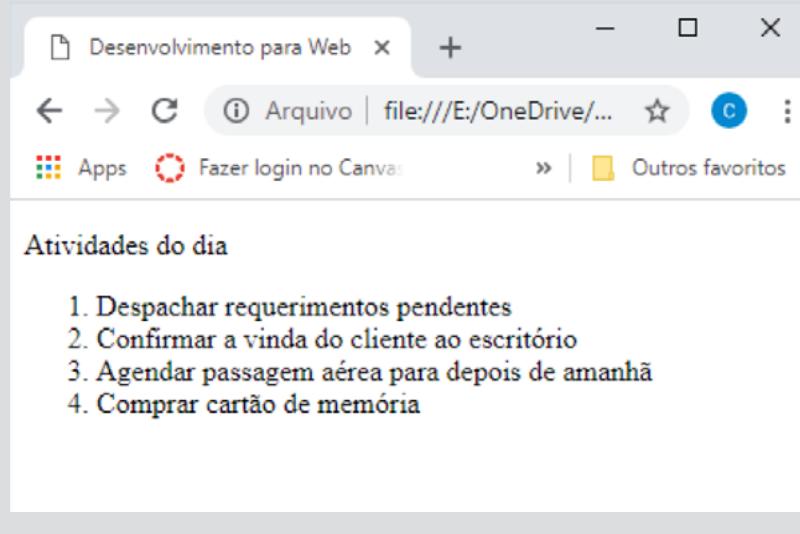
<dl> ... </dl> – Serve para definir uma lista.
<dt> ... </dt> – Serve para definir um termo da lista.
<dd> ...</dd> – Serve para definir uma descrição para o respectivo termo da lista.

 ... - Define uma lista ordenada (numerados)

Permite a criação de uma lista com itens ordenados.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta content="text/html; charset=utf-8">
<title>Desenvolvimento para Web</title>
</head>
<body>
<p> Atividades do dia </p>
<ol>
<li>Despachar requerimentos pendentes</li>
<li>Confirmar a vinda do cliente ao escritório</li>
<li>Agendar passagem aérea para depois de amanhã</li>
<li>Comprar cartão de memória</li>
</ol>
</body>
</html>
```



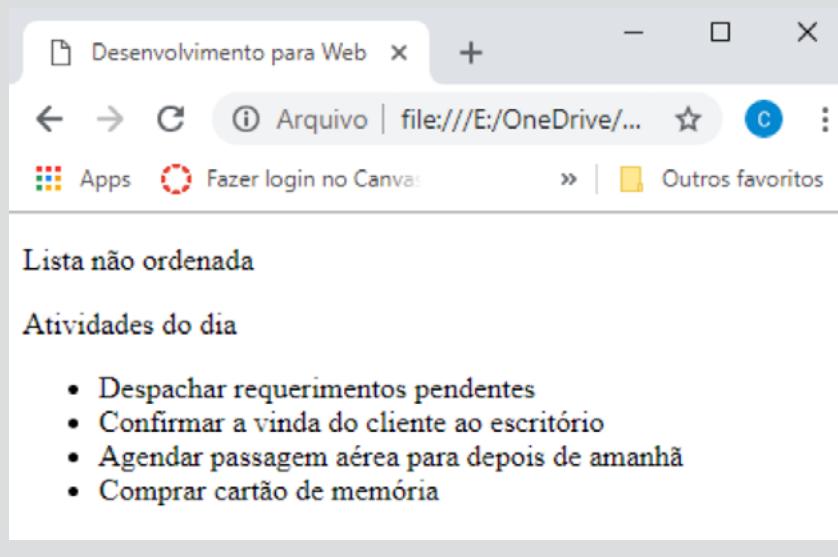
Note que foi incluída uma numeração nos itens que não foi especificada na lista. Cada tag ... indica que esse é um item diferente da lista.

 ... - Define uma lista ordenada (não numerados)

Permite a criação de uma lista com itens não ordenados, semelhante à lista ordenada, em que cada item é definido por um ponto tal qual um item (sem numeração).

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Desenvolvimento para Web</title>
</head>
<body>
    <p>Lista não ordenada</p>
    <p>Atividades do dia</p>
    <ul>
        <li>Despachar requerimentos pendentes</li>
        <li>Confirmar a vinda do cliente ao escritório</li>
        <li>Agendar passagem aérea para depois de amanhã</li>
        <li>Comprar cartão de memória</li>
    </ul>
</body>
</html>
```



Assim como na lista ordenada, cada tag ... indica que esse é um item diferente da lista.

 **Saiba mais**

Para saber mais sobre tabelas e listas, leia o Capítulo 1 (p. 23-29) do seguinte livro:

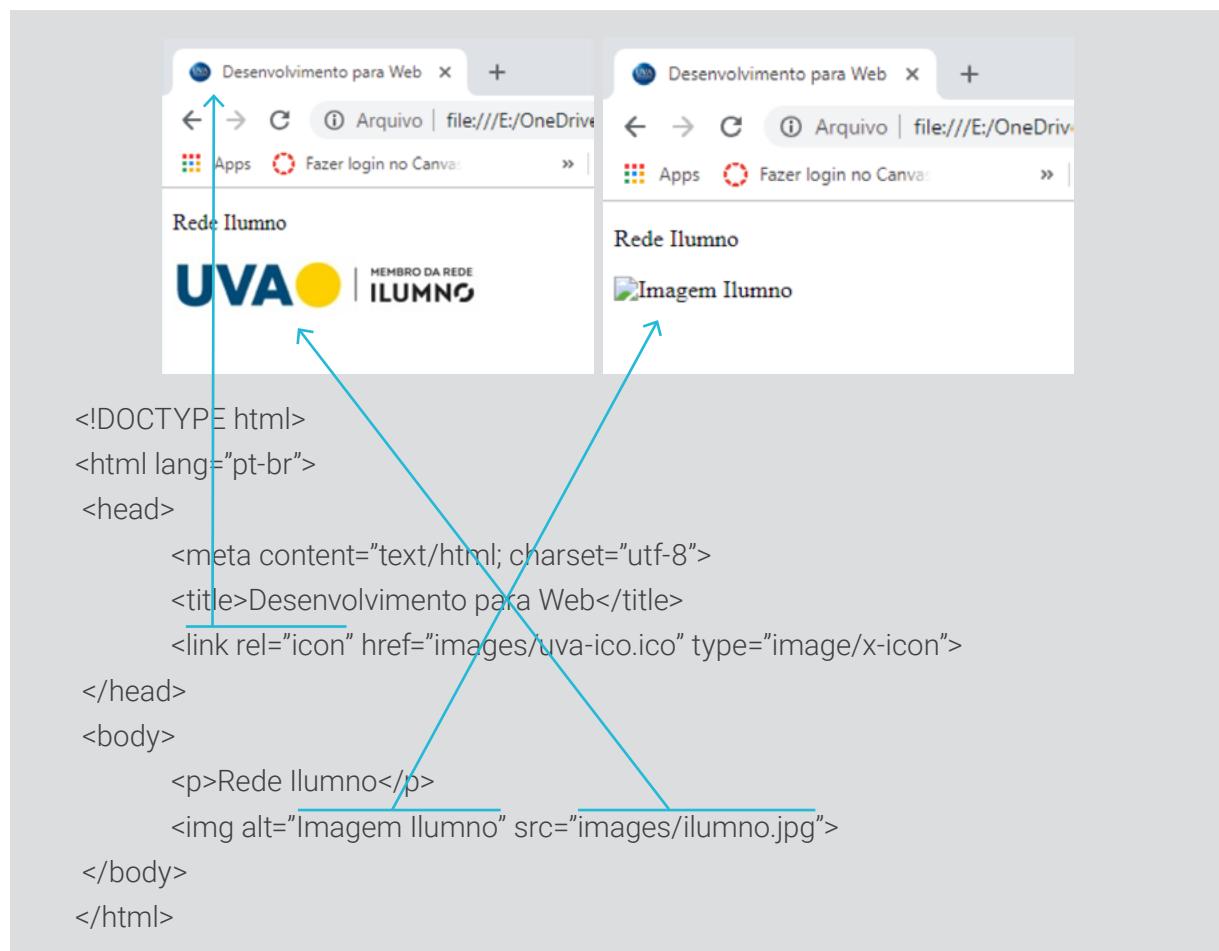
SEGURADO, V. S. **Projeto de interface com o usuário**. São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

 - Exibe uma imagem

Permite que seja exibida uma imagem na página.

Exemplo:

Resultado:



```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Desenvolvimento para Web</title>
    <link rel="icon" href="images/uva-ico.ico" type="image/x-icon">
</head>
<body>
    <p>Rede Ilumno</p>
    
</body>
</html>
```

A opção alt é opcional e permite determinar um texto alternativo para ser exibido se a imagem não puder ser carregada pelo navegador. O texto, no caso “Imagen Ilumno”, é exibido no segundo exemplo.

A opção src é fundamental, pois é ela que determina a localização e especificação do nome do arquivo de imagem. A configuração: images/ilumno.jpg indica que o arquivo de imagem “ilumno.jpg” se encontra no subdiretório images do servidor de arquivos. É comum a criação de um subdiretório para armazenar as imagens de um site.

Observação: Nesse exemplo foi incluído um link relacionado para identificar o ícone da página, que é apresentado junto com seu título.

```
<link rel="icon" href="images/uva-ico.ico" type="image/x-icon">
```

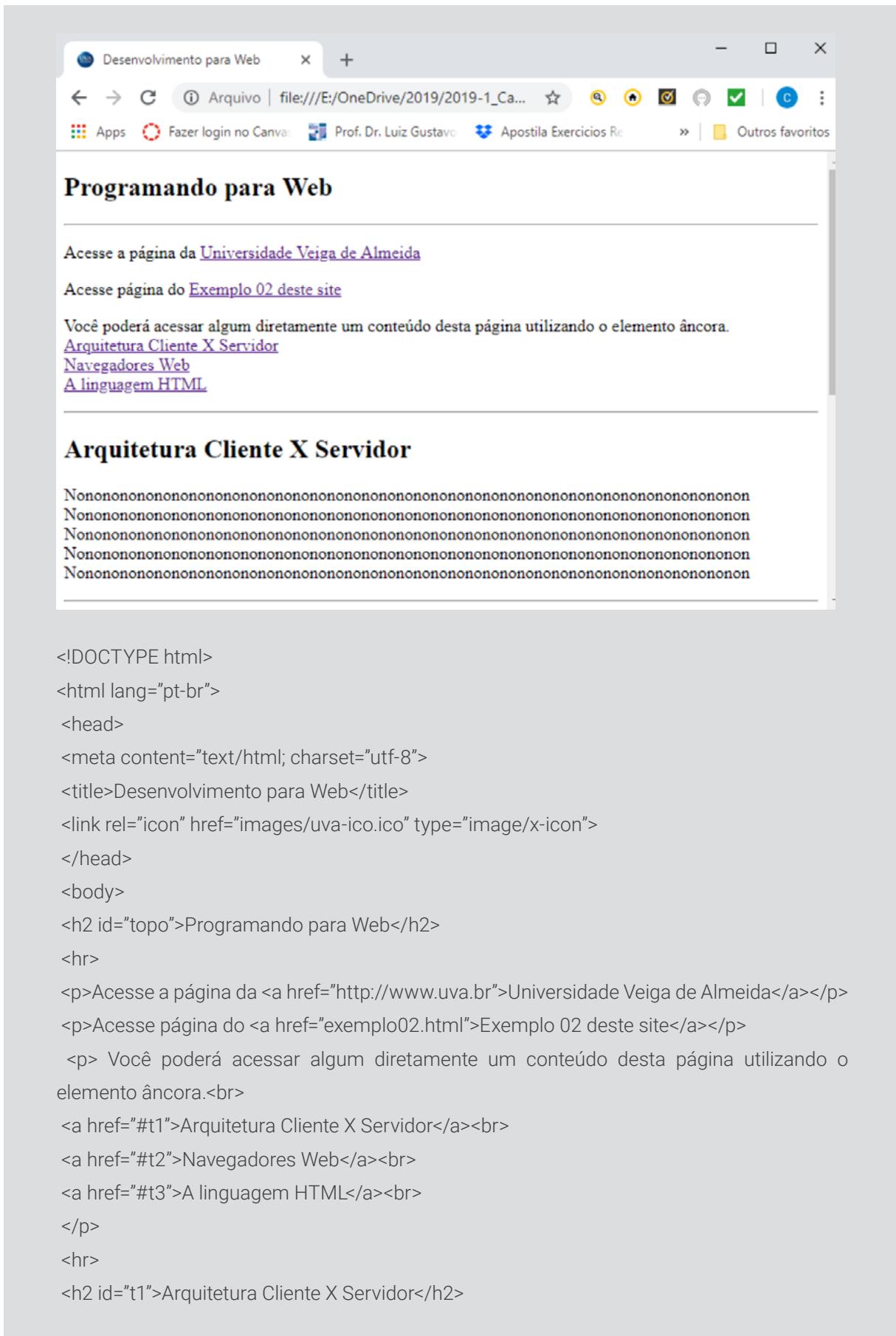
<a> - Define uma âncora ou link

Permite a criação de links para dentro da própria página (internos ou âncora) ou para outras páginas e sites (externos ou link).

Exemplos:

- Universidade Veiga de Almeida
 - Cria um link para o site da Universidade Veiga de Almeida.
- Página do exemplo 1
 - Cria um link para uma página armazenada no mesmo diretório do servidor. Usado para acesso às demais páginas de um mesmo site.
- Site 2
 - Cria um link para a página inicial de outro site ou parte de um site que se encontra no mesmo servidor, mas em diretório de mesma hierarquia que o site original.
- Tema 1
 - Cria um link interno (âncora para a mesma página, mas para um local específico dela). É necessário identificar o local incluindo uma configuração de id (identificação), conforme o exemplo: <h2 id="local1">Título do assunto.</h2>

A página inicial de site deve se chamar “index.html”. Isso não é obrigatório, mas depende das configurações do servidor. A maioria dos servidores carregam a página inicial quando acessamos um site por meio do seu endereço Web: www.site.com.



Essa página possui links para um site externo (no caso, da UVA), um link para outra página do mesmo site, que se encontra no mesmo diretório da página inicial do site (index.html), além de links internos ou âncoras para acesso rápido, como em um atalho para conteúdos que se encontram na mesma página.



MIDIATECA

Acesse a midiateca da Unidade 1 e veja o conteúdo complementar indicado pelo professor sobre como tornar uma imagem em um link.



Saiba mais

Para saber mais sobre elementos de texto, link, âncora e imagens, leia o Capítulo 1 (p. 11-18) do seguinte livro:

SEGURADO, V. S. **Projeto de interface com o usuário**. São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

Você verá mais detalhes sobre as tags apresentadas aqui, além de ter a oportunidade de aumentar o seu conhecimento com o aprendizado de novas tags, lendo o conteúdo a seguir:

TERUEL, E. C. **HTML 5**: guia prático. 2. ed. São Paulo: Érica, 2014, p. 222-309. Minha Biblioteca.

Opinião do autor

Existem muitas aplicações para a criação de páginas para a Web, o próprio Word é um exemplo, mas o desenvolvimento de aplicações profissionais necessita do conhecimento estrutural do HTML. Páginas *front-ends* criadas por editores exigem muitos ajustes que só podem ser feitos com edição direta do código, sem contar que o desenvolvimento de *back-ends* obriga que a codificação HTML seja realizada sem o uso de aplicativos de edição, uma vez que o código HTML é inserido diretamente junto com a linguagem de programação utilizada no servidor.



NA PRÁTICA

Você pode não ter notado, mas toda vez que você realiza login em um site, você está usando uma estrutura baseada na arquitetura cliente *versus* servidor. Ao acessar uma conta do Gmail, por exemplo, por meio do endereço <http://www.gmail.com>, você realiza o login em um *front-end* (cliente) em sua máquina, mas a verificação do usuário e senha, assim como o conteúdo retornado ao seu navegador para que você possa ver os e-mails, é o resultado de uma aplicação *back-end* (servidor).

Resumo da Unidade 1

Nesta unidade você compreendeu a arquitetura cliente versus servidor e entendeu a diferença entre a criação de front-end (cliente) e back-end (servidor) de uma aplicação para a Web, tendo como foco a criação de front-ends. Compreendeu ainda a importância dos navegadores e dos protocolos de rede para o desenvolvimento de aplicações para a Web. Por fim, teve o entendimento do que seria um HTML com a aplicação de tags básicas para a criação de páginas e desenvolvimento de sites de conteúdo para aplicações clientes na Web.



CONCEITO

Nesta unidade destacaram-se os seguintes conteúdos: a diferença entre a construção de *front-ends* (cliente) e de *back-ends* (servidor) e o processo de criação de uma página HTML e de um site de conteúdo a partir de um conjunto de páginas inter-relacionadas.

Referências

FLATSCHART, F. **HTML 5:** embarque imediato. Rio de Janeiro: Brasport, 2018. Biblioteca Virtual.

SEGURADO, V. S. **Projeto de interface com o usuário.** São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

TERUEL, E. C. **HTML 5:** guia prático. 2. ed. São Paulo: Érica, 2014. Minha Biblioteca.

UNIDADE 2

HTML avançado com
desenvolvimento de formulários

INTRODUÇÃO

A criação de sites de conteúdo é bem simples, uma vez que basta apresentar ao usuário um conteúdo que será lido por ele. A esse tipo de conteúdo, chamamos de estático, isso porque o conteúdo será sempre o mesmo para todos os usuários, e o servidor precisa apenas armazenar os arquivos dessas páginas.

Já quando temos uma aplicação Web, temos uma interação entre um usuário específico e o servidor. Nesse caso, é normal que o usuário seja identificado, pois o conteúdo a ser exibido será específico, necessitando assim da identificação do usuário. Dessa forma, é muito comum em aplicações Web a necessidade de um acesso ao sistema, geralmente relacionado a um login e uma senha. Como nesses casos não serão utilizados apenas links para outras páginas, é necessário que tenhamos uma forma de interação do usuário com o sistema, e essa interação é realizada por meio de formulários.

Formulários Web nada mais são do que um conjunto de componentes de tela específicos para realizar a interação do usuário com o sistema Web. Nesta unidade, trataremos da criação e formatação de formulários HTML para a interação entre o cliente (*front-end*) e o servidor (*back-end*).



OBJETIVO

Nesta unidade, você será capaz de:

- Construir páginas de formulários de *front-end* para aplicações Web.

Introdução ao desenvolvimento de formulários para a Web

A montagem de uma página HTML pode exigir uma organização maior quando usamos em uma mesma página um conteúdo e um formulário. Organizar adequadamente uma página que contém um formulário é fundamental para a criação de uma interface com o usuário que seja atraente, além de facilitar o entendimento do usuário. Por essa razão, é importante acomodar diferentes partes de uma mesma página de forma adequada, muitas vezes para podermos alinhar apropriadamente todo o conteúdo necessário.

Vamos conhecer primeiramente como criar uma tabela, que é uma forma eficiente não apenas de organizar os diferentes conteúdos de uma mesma página, mas também para alinhar e facilitar a interação do usuário com um formulário para o preenchimento de dados.

Uma tabela é um componente HTML capaz de estruturar diferentes conteúdos por meio da organização do conteúdo em linhas e colunas. As linhas e colunas podem ser mescladas de forma a adequar com maior qualidade visual esses diferentes conteúdos ou componentes de formulários.

Inicialmente, vamos conhecer as principais tags referentes à criação de uma tabela, sem a aplicação de estilos.

As principais tags para a criação de tabelas são:

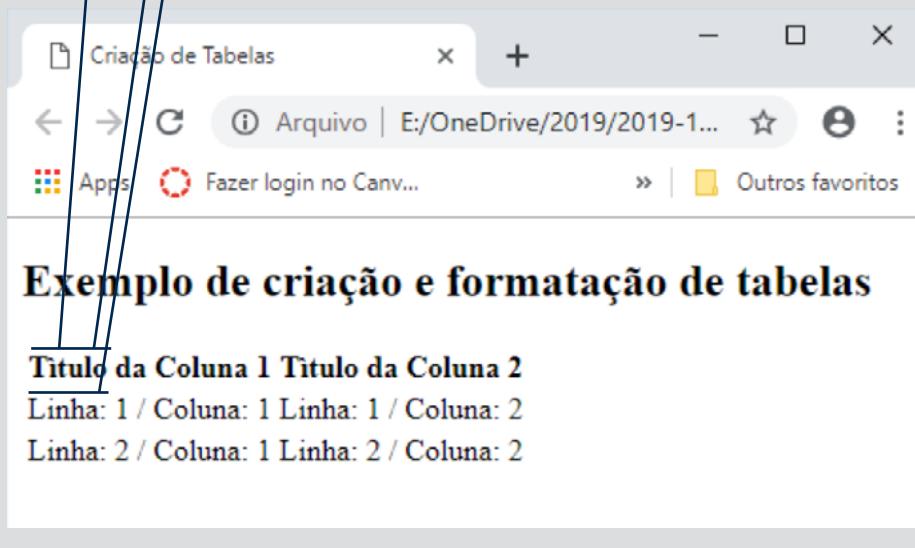
TAGS	Descrição
<table> ... </table>	São responsáveis por delimitar o início e término da tabela.
<tr>...</tr>	São responsáveis por delimitar cada nova linha da tabela.
<th>...</th>	São responsáveis por delimitar e formatar uma coluna como um título.
<td>...</td>	São responsáveis por delimitar cada coluna de uma linha da tabela.

Podemos, então, analisar a criação de uma tabela simples, por meio da página: tabela1.html:

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Criação de Tabelas</title>
</head>
<body>
    <h2> Exemplo de criação e formatação de tabelas</h2>
    <table>
        <tr>
            <th>Título da Coluna 1</th>
            <th>Título da Coluna 2</th>
        </tr>
        <tr>
            <td>Linha: 1 / Coluna 1</td>
            <td>Linha: 1 / Coluna: 2</td>
        </tr>
        <tr>
            <td>Linha: 2 / Coluna: 1</td>
            <td>Linha: 2 / Coluna: 2</td>
        </tr>
    </table>
</body>
</html>

```



Observações:

- A primeira linha da tabela tem o seu conteúdo em negrito, isso se deve ao fato de essas colunas serem do tipo TH, ou seja, formatado como título.

- Cada linha está separada por TR.
- As colunas de cada linha estão alinhadas.

Ponto de Atenção: Sempre que for colocar uma tag TD, deve estar dentro da tag TR e sempre seguindo a seguinte estrutura:

```
<tr>
    <th>Título da Coluna 1</th>
    <th>Título da Coluna 2</th>
</tr>
```

Você pode utilizar linhas de borda para cada linha e coluna, incluindo um estilo a sua tabela, com a inclusão do descritor border, conforme pode ser observado no exemplo tabela2.html:

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta content="text/html; charset=utf-8">
        <title>Criação de Tabelas</title>
    </head>
    <body>
        <h2> Exemplo de criação e formatação de tabelas</h2>
        <table border="1">
            <tr>
                <th>Título da Coluna 1</th>
                <th>Título da Coluna 2</th>
            </tr>
            <tr>
                <td>Linha: 1 / Coluna: 1</td>
                <td>Linha: 1 / Coluna: 2</td>
            </tr>
            <tr>
                <td>Linha: 2 / Coluna: 1</td>
                <td>Linha: 2 / Coluna: 2</td>
            </tr>
            <tr>
                <td>Linha: 3 / Coluna: 1</td>
                <td>Linha: 3 / Coluna: 2</td>
            </tr>
        </table>
    </body>
</html>
```

The screenshot shows a web browser window titled "Criação de Tabelas". The address bar indicates the file is located at E:/OneDrive/2019/2019-1_Car... . The page content is titled "Exemplo de criação e formatação de tabelas". It displays a table with two columns and three rows. The first column is labeled "Titulo da Coluna 1" and the second column is labeled "Titulo da Coluna 2". The rows contain the text "Linha: 1 / Coluna: 1", "Linha: 2 / Coluna: 1", and "Linha: 3 / Coluna: 1" respectively, followed by "Linha: 1 / Coluna: 2", "Linha: 2 / Coluna: 2", and "Linha: 3 / Coluna: 2".

Titulo da Coluna 1	Titulo da Coluna 2
Linha: 1 / Coluna: 1	Linha: 1 / Coluna: 2
Linha: 2 / Coluna: 1	Linha: 2 / Coluna: 2
Linha: 3 / Coluna: 1	Linha: 3 / Coluna: 2

Observações:

- Ao determinar o estilo de borda com o valor 1, a tabela passou a apresentar uma linha externa a toda a tabela, além de linhas internas delimitando externamente cada coluna.
- Se você aumentar o valor do descritor border, as linhas externas da tabela ficarão mais grossas, mas não ocorrerão mudanças nas linhas internas, como pode ser observado a seguir com o valor do descritor border = "5".

The screenshot shows a web browser window titled "Criação de Tab...". The address bar indicates the file is located at E:/OneDrive/2019/2019-1_Car... . The page content is titled "Exemplo de criação e formatação de tabelas". It displays a table with two columns and three rows, enclosed in a thick black border. The first column is labeled "Titulo da Coluna 1" and the second column is labeled "Titulo da Coluna 2". The rows contain the text "Linha: 1 / Coluna: 1", "Linha: 2 / Coluna: 1", and "Linha: 3 / Coluna: 1" respectively, followed by "Linha: 1 / Coluna: 2", "Linha: 2 / Coluna: 2", and "Linha: 3 / Coluna: 2".

Titulo da Coluna 1	Titulo da Coluna 2
Linha: 1 / Coluna: 1	Linha: 1 / Coluna: 2
Linha: 2 / Coluna: 1	Linha: 2 / Coluna: 2
Linha: 3 / Coluna: 1	Linha: 3 / Coluna: 2

Exemplo de criação e formatação de tabelas

Titulo da Coluna 1	Titulo da Coluna 2
Linha: 1 / Coluna: 1	Linha: 1 / Coluna: 2
Linha: 2 / Coluna: 1	Linha: 2 / Coluna: 2
Linha: 3 / Coluna: 1	Linha: 3 / Coluna: 2

É comum que tenhamos conteúdos com imagens ou componentes de formulários que podem exigir tamanhos diferentes de linhas ou colunas. Para esses casos, podemos mesclar tanto linhas como colunas da tabela para melhor organizar o conteúdo ou componentes do formulário.

No exemplo a seguir (tabela3.html), você pode analisar o código para entender como é realizada a mesclagem de uma ou mais linhas e colunas de uma tabela:

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta content="text/html; charset=utf-8">
        <title>Criação de Tabelas</title>
    </head>
    <body>
        <h2> Exemplo de criação e formatação de tabelas</h2>
        <table border="1">
            <tr>
                <th>Título da coluna 1</th>
                <th>Título da coluna 2</th>
                <th>Título da coluna 3</th>
            </tr>
            <tr>
                <td rowspan="2">Linhas: 1 e 2 / Coluna: 1</td>
                <td>Linha: 1 / Coluna: 2</td>
                <td>Linha: 2 / Coluna: 3</td>
            </tr>
            <tr>
                <!-- <td>Linha: 2 / Coluna: 1</td>
                Esta linha foi comentada, pois será substituída
                pelo conteúdo da linha anterior-->
                <td>Linha: 2 / Coluna: 2</td>
                <td>Linha: 2 / Coluna: 3</td>
            </tr>
            <tr>
                <td colspan="3"><center>Linha: 3 / Colunas: 1, 2 e 3</center></td>
                <!-- <td>Linha: 3 / Coluna: 2</td>
                <td>Linha: 3 / Coluna: 3</td>
                Estas linhas foram comentadas, pois serão
                substituídas pelo conteúdo da coluna anterior-->
            </tr>
        </table>
    </body>
</html>

```

Exemplo de criação e formatação de tabelas

Título da coluna 1	Título da coluna 2	Título da coluna 3
Linha: 1 e 2 / Coluna: 1	Linha: 1 / Coluna: 2, 3	Linha: 2 / Coluna: 3
Linha: 2 / Coluna: 2	Linha: 2 / Coluna: 3	
Linha: 3 / Colunas: 1, 2 e 3		

Observações:

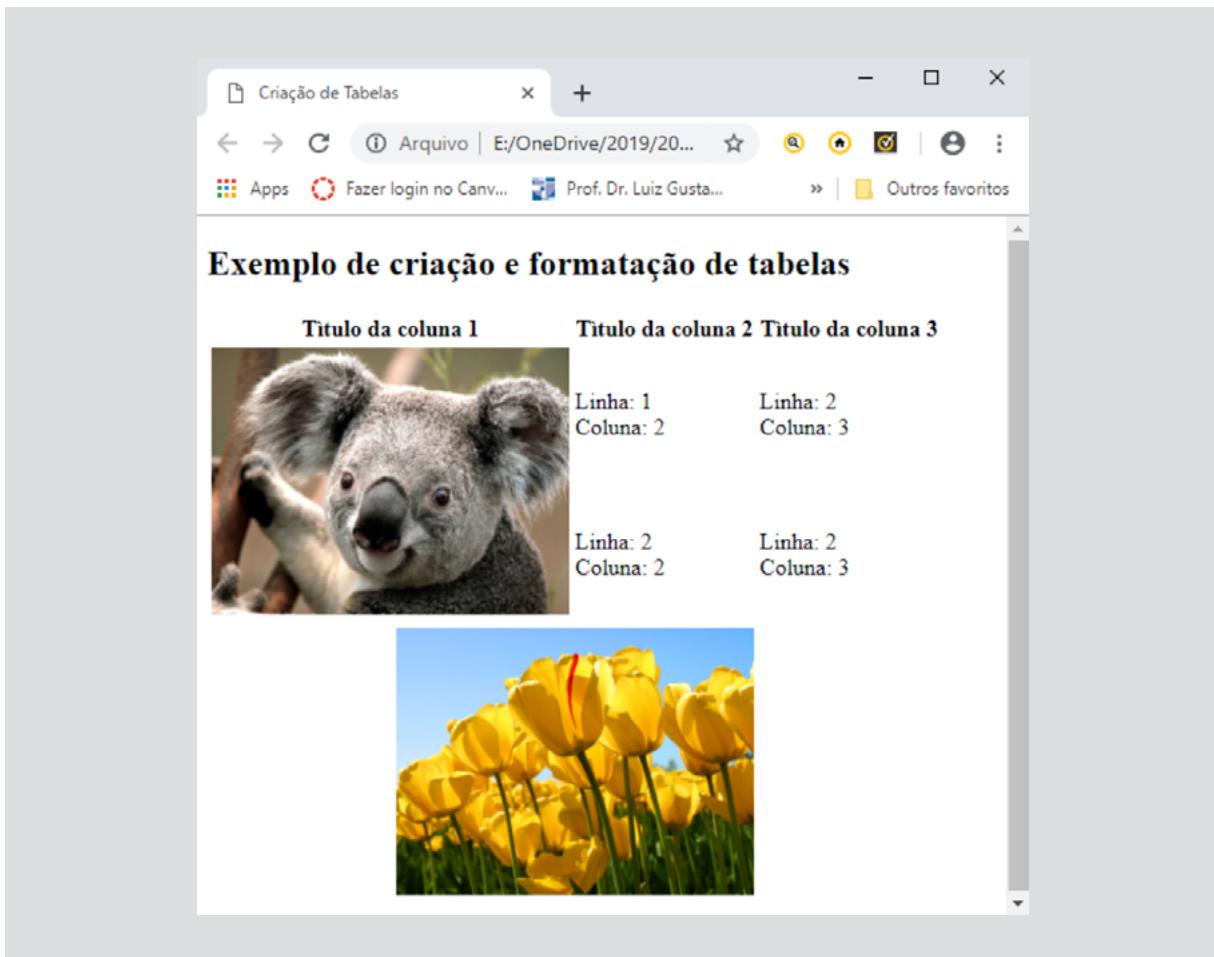
- Ao se incluir o descritor **rowspan="2"**, você determinou que a coluna fosse mesclada com a primeira coluna da próxima linha, por isso você deve notar que a primeira coluna da próxima linha foi comentada, uma vez que o seu conteúdo será o mesmo da primeira coluna da linha anterior. Caso você mantenha essa coluna, a linha passará a conter uma coluna a mais do que as demais linhas da tabela. O uso de um número maior para esse descritor fará com que mais linhas sejam mescladas.
- O mesmo ocorre similarmente com as colunas ao se utilizar **colspan="3"**, à diferença que serão mescladas colunas e não linhas. Veja que, na última linha, todas as três colunas foram mescladas.

Em uma tabela podemos utilizar qualquer tipo de componente HTML. Vamos analisar o próximo exemplo, tabela4.html, em que foram retiradas as linhas separadoras externas e incluídas imagens no lugar de textos. Essa é uma forma de apresentar imagens e textos em sites de comércio eletrônico:

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta content="text/html; charset=utf-8">
        <title>Criação de Tabelas</title>
    </head>
    <body>
        <h2> Exemplo de criação e formatação de tabelas</h2>
        <table>
            <tr>
                <th>Título da coluna 1</th>
                <th>Título da coluna 2</th>
                <th>Título da coluna 3</th>
            </tr>
            <tr>
                <td rowspan="2"></td>
                <td>Linha: 1 <br>Coluna: 2</td>
                <td>Linha: 2 <br>Coluna: 3</td>
            </tr>
            <tr>
                <!-- <td>Linha: 2 / Coluna: 1</td>
                Esta linha foi comentada, pois será substituída
                pelo conteúdo da linha anterior-->
                <td>Linha: 2 <br>Coluna: 2</td>
                <td>Linha: 2 <br>Coluna: 3</td>
            </tr>
            <tr>
                <td colspan="3"><center></center></td>
                <!-- <td>Linha: 3 / Coluna: 2</td>
                <td>Linha: 3 / Coluna: 3</td>
                Estas linhas foram comentadas, pois serão
                substituídas pelo conteúdo da coluna anterior-->
            </tr>
        </table>
    </body>
</html>

```



Observações:

- Não se esqueça de colocar, no mesmo diretório da página, os dois arquivos de imagens e de definir o descriptor SRC das tags IMG com os nomes dos arquivos que você usou.
- Repare que as colunas permanecem alinhadas dentro de cada linha.
- Você também pode incluir links às imagens ou aos textos internos à tabela.

Você também pode organizar sua página utilizando a tag DIV, que é capaz de dividir uma mesma página em partes menores, em que você pode definir diferentes formatações a cada uma dessas partes, esta tag é muito utilizada no HTML5 para colocar sua página compatível para diversas telas de visualização. O DIV também pode dividir o seu código em trechos menores, que podem ser mais bem compreendidos e organizados.

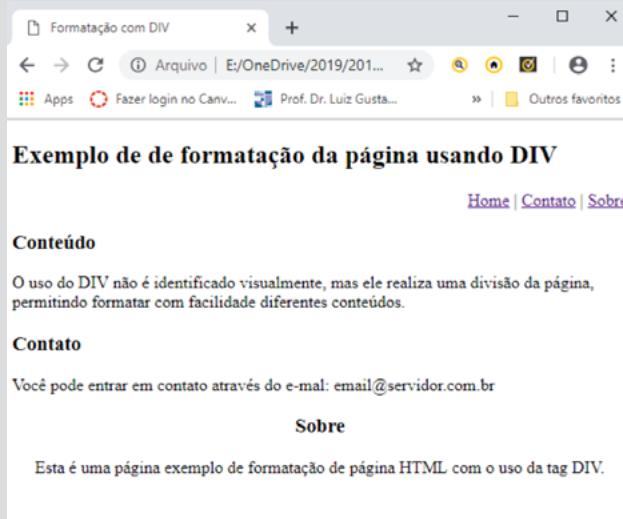
No exemplo a seguir, o DIV é usado para separar o cabeçalho da página de um menu e dos demais conteúdos. Cada conteúdo foi separado então em uma DIV diferente, que pode ser formatada individualmente.

O exemplo a seguir div1.html apresenta uma forma prática de separar o menu da página dos demais conteúdos:

```

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta content="text/html; charset=utf-8">
    <title>Formatação com DIV</title>
  </head>
  <body>
    <h2> Exemplo de de formatação da página usando DIV</h2>
    <div id="menu" align="right">
      <a href="#t1">Home</a> | <a href="#t2">Contato</a> | <a href="#t3">Sobre</a>
    </div>
    <div id="t1" align="left">
      <h3>Conteúdo</h3>
      <p>O uso do DIV não é identificado visualmente, mas ele realiza uma divisão da página, permitindo formatar com facilidade diferentes conteúdos.</p>
    </div>
    <div id="t2" align="left">
      <h3>Contato</h3>
      <p>Você pode entrar em contato através do e-mail: email@servidor.com.br</p>
    </div>
    <div id="t3" align="center">
      <h3>Sobre</h3>
      <p>Esta é uma página exemplo de formatação de página HTML com o uso da tag DIV.</p>
    </div>
  </body>
</html>

```



Observações:

- Nesse exemplo, foram criadas quatro diferentes regiões, sendo elas: menu, conteúdo, contato e sobre.
- Foram utilizadas âncoras (links internos para locais específicos da própria página), mas poderiam ter sido usados links externos também.
- Cada região foi identificada por meio do descritor **id**.
- Cada região foi alinhada à direita, à esquerda ou ao centro, de acordo com o descritor **align**, permitindo que elas possuam sua própria formatação.



MIDIA TECA

Acesse a midiateca da Unidade 2 e veja o conteúdo complementar indicado pelo professor sobre a criação de tabelas em páginas HTML para organizar o conteúdo de uma página.



Saiba mais

Elementos HTML para criação de tabelas e divisões da página:

TERUEL, E. C. **HTML 5**: guia prático. 2. ed. São Paulo: Érica, 2014. Cap. 6. p. 264-266; 297-304.

As principais tags HTML para a criação de formulários para a Web

Formulários em HTML têm como objetivo criar uma interface entre o usuário e a aplicação Web. Por meio dos formulários, o usuário interage com o sistema preenchendo dados que serão enviados ao servidor. Esses dados podem ser previamente criticados para se averiguar se eles são válidos, por intermédio da programação JavaScript, que será objeto de nossos estudos em nossa última unidade, para que esses dados sejam tratados pelo servidor (*back-end*).

Existem alguns tipos de componentes específicos para o desenvolvimento de telas de aplicativos e que são suportados pelo HTML. Uma mesma página pode conter um ou mais formulários, mas o mais comum é que tenhamos um formulário por página, por isso o formulário também deve conter um identificador.

Um formulário HTML inicia por uma tag **<form>** e se encerra pela tag **</form>**.

Existem dois métodos de envio de dados por meio de um formulário:

- Get – O método “get” envia os dados juntamente com a URL especificada no atributo “action”, utilizando o caractere ponto de interrogação como separador, por exemplo: <https://www.loja.com/carrinho.php?categoria=12&codproduto=123>

No exemplo, os dados de categoria e o código do produto serão enviados ao script carrinho.php, que receberá a categoria=12 e codproduto=123.

Esse método é considerado inseguro para envio de senhas, uma vez que o seu conteúdo é exibido durante o envio; além disso, esse método está limitado a até 2.048 caracteres na URL de envio e todos os dados são enviados como textos (*strings*).

- Post – O método “post” envia os dados de forma oculta do usuário, sendo mais indicado para envio de senhas. Esse método não possui limitação de tamanho, mas suporta diferentes tipos de dados no envio (texto, numérico etc). Um exemplo de criação de formulário em HTML é: `<form id="formcadastro" action="https://www.loja.com.br/carrinho.php" method="post">`



Importante

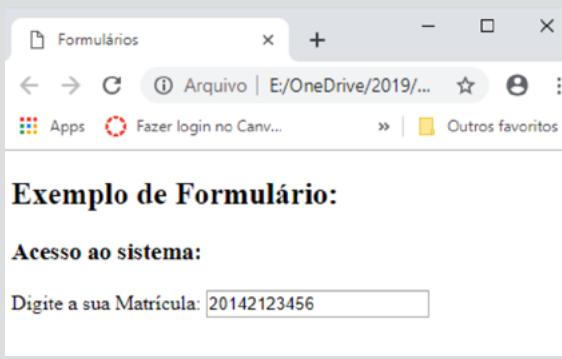
O método get é considerado o padrão de envio de dados pelo HTML.

Os componentes de entrada de dados são chamadas através da tag input e os tipos mais comuns são:

- text – Utilizado para a entrada de dados que serão digitados pelo usuário. Seu conteúdo será exibido para que o usuário possa se certificar de ter inserido dados corretos.

Exemplo: <p>Digite a sua Matrícula: <input type="text" name="matricula"></p> [formulario1.html]

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta content="text/html; charset=utf-8">
<title>Formulários</title>
</head>
<body>
<h2> Exemplo de Formulário:</h2>
<div id="t1" align="left">
<h3>Acesso ao sistema:</h3>
<form id="acesso" action="https://www.site.com.br/acesso.php" method="post"
>
    <p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
</form>
</div>
</body>
</html>
```



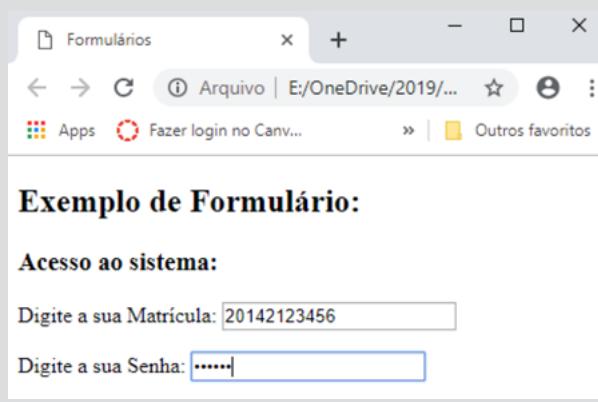
Observação: o descritor name identifica o nome do campo, isso é importante para o envio dos dados, uma vez que o envio deve ser feito por meio do par: identificador / valor e poderemos ter vários pares de dados com o nome do campo e o valor associado em uma mesma página.

- password – Utilizado para entrada de dados que não devem ser exibidos, tais como senhas; o seu conteúdo é mascarado para não ser lido, mas são indicados que dados foram digitados.

Exemplo: <p>Digite a sua Senha: <input type="password" name="senha" ></p> [formula-rio2.html]

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Formulários</title>
</head>
<body>
    <h2> Exemplo de Formulário:</h2>
    <div id="t1" align="left">
        <h3>Acesso ao sistema:</h3>
        <form id="acesso" action="https://www.site.com.br/acesso.php" method="post" >

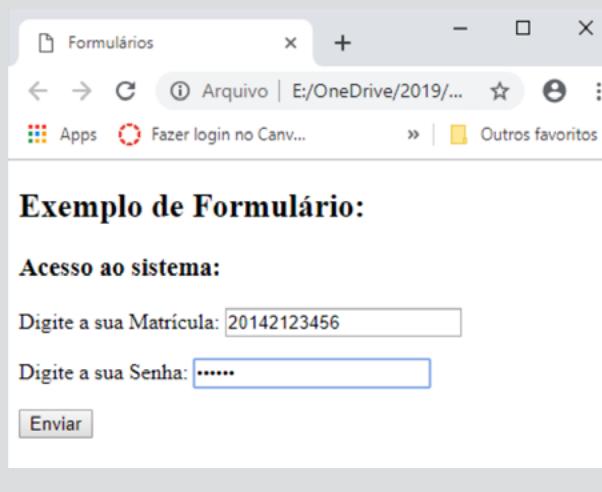
            <p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
            <p>Digite a sua Senha: <input type="password" name="senha"></p>
        </form>
    </div>
</body>
</html>
```



- submit – Utilizado para criar um botão responsável por enviar os dados do formulário ao servidor.

Exemplo: <p><input type="submit" value="Enviar" ></p>
 [formulario3.html]

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta content="text/html; charset=utf-8">
  <title>Formulários</title>
</head>
<body>
  <h2> Exemplo de Formulário:</h2>
  <div id="t1" align="left">
    <h3>Acesso ao sistema:</h3>
    <form id="acesso" action="https://www.site.com.br/acesso.php" method="post">
      <p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
      <p>Digite a sua Senha: <input type="password" name="senha"></p>
      <p><input type="submit" value="Enviar" ></p>
    </form>
  </div>
</body>
</html>
```



Observação: se desejado, você pode utilizar uma imagem como botão “submit”:

```
<p> <input type="image" src="enviar.gif" alt="Enviar dados."> </p>
```

Nesse caso, o tipo (type) é “image”, “src” define onde encontrar o arquivo de imagem e “alt” é um texto alternativo; aparece no lugar da imagem, caso ela não possa ser carregada ou não exista. Ao clicar na imagem, as coordenadas são passadas ao destinatário da mensagem.

- reset – Utilizado para criar um botão capaz de limpar todos os dados do formulário.

Exemplo: <p><input type="reset" value="Limpar" ></p>
[formulario4.html]

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta content="text/html; charset=utf-8">
    <title>Formulários</title>
  </head>
  <body>
    <h2> Exemplo de Formulário:</h2>
    <div id="t1" align="left">
      <h3>Acesso ao sistema:</h3>
      <form id="acesso" action="https://www.site.com.br/acesso.php" method="post"
      >
        <p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
        <p>Digite a sua Senha: <input type="password" name="senha"></p>
        <p><input type="submit" value="Enviar" ></p>
        <p><input type="reset" value="Limpar" ></p>
      </form>
    </div>
  </body>
</html>
```

Exemplo de Formulário:

Acesso ao sistema:

Digite a sua Matrícula:

Digite a sua Senha:

Observação: você pode manter os dois botões na mesma linha, caso você mantenha no mesmo parágrafo:

```
<p><input type="submit" value="Enviar" >
<input type="reset" value="Limpar" ></p> [formulario4b.html]
```

Exemplo de Formulário:

Acesso ao sistema:

Digite a sua Matrícula:

Digite a sua Senha:

- radio – Utilizado para apresentar ao usuário um conjunto de opções em que apenas uma pode ser selecionada.

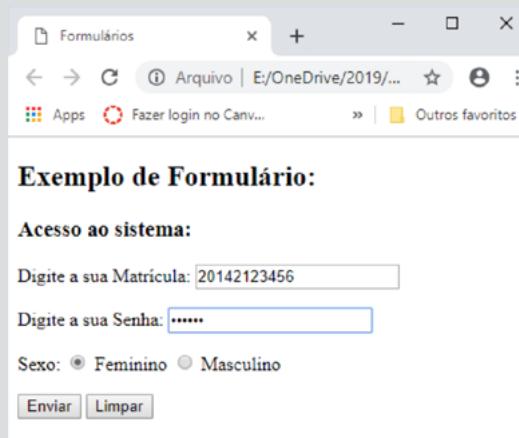
Exemplo:

```
<p>Sexo: <input type="radio" name="sexo" value="f" checked > Feminino
<input type="radio" name="sexo" value="m"> Masculino</p> [formulario5.html]
```

```

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta content="text/html; charset=utf-8">
        <title>Formulários</title>
    </head>
    <body>
        <h2> Exemplo de Formulário:</h2>
        <div id="t1" align="left">
            <h3>Acesso ao sistema:</h3>
            <form id="acesso" action="https://www.site.com.br/acesso.php" method="post">
                <p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
                <p>Digite a sua Senha: <input type="password" name="senha"></p>
                <p>Sexo: <input type="radio" name="sexo" value="f" checked> Feminino
                    <input type="radio" name="sexo" value="m"> Masculino</p>
                <p><input type="submit" value="Enviar" ></p>
                <p><input type="reset" value="Limpar" ></p>
            </form>
        </div>
    </body>
</html>

```



Observações:

- Como os botões de rádio estão relacionados, eles devem conter o mesmo nome de campo. Isso é necessário para enviar apenas o valor da opção selecionada, além de permitir que você possa criar diferentes tipos de escolhas em um mesmo formulário.

- Se você precisar ainda cadastrar o estado civil do usuário, basta criar um novo conjunto de botões de rádio com o mesmo descritor name.
- O descritor checked determina que o botão com esse descritor será exibido ao usuário previamente marcado. Só devemos incluir o descritor checked em uma opção em cada conjunto de botões de rádio.
- O descritor value determina o valor que será associado ao nome do campo (name) quando do envio dos dados. Isso permite que em um mesmo conjunto de rádios apenas o valor da opção selecionada seja enviada ao servidor. No exemplo, se a opção Feminino estiver selecionada, o valor do campo sexo será "f", se a opção Masculino estiver selecionada, o valor enviado no campo sexo será "m".
- checkbox – Utilizado para apresentar ao usuário opções do tipo Sim ou Não, por meio da marcação de confirmação ou não. Cada checkbox deve ser individual, uma vez que nos interessa saber separadamente quais estão selecionados e quais não estão para que possamos enviar a confirmação ou não de cada individualmente, por isso os descritores name de cada um devem ser diferentes, ao contrário dos conjuntos de rádio, em que cada conjunto deve possuir o mesmo nome no descritor name.

Exemplo:

```
<p>Línguas Estrangeiras:<br>
<input type="checkbox" name="ingles" value="i"> Inglês<br>
<input type="checkbox" name="frances" value="f"> Francês<br>
<input type="checkbox" name="espanhol" value="e"> Espanhol</p>[formulario6.html]
```

```
<html lang="pt-br">
<head>
  <meta content="text/html; charset=utf-8">
  <title>Formulários</title>
</head>
<body>
  <h2> Exemplo de Formulário:</h2>
  <div id="t1" align="left">
    <h3>Acesso ao sistema:</h3>
    <form id="acesso" action="https://www.site.com.br/acesso.php" method="post" >
```

```

<p>Digite a sua Matrícula: <input type="text" name="matricula"></p>
<p>Digite a sua Senha: <input type="password" name="senha"/><p>
<p>Sexo: <input type="radio" name="sexo" value="f" checked > Feminino
    <input type="radio" name="sexo" value="m"> Masculino</p>
<p>Línguas Estrangeiras:<br>
    <input type="checkbox" name="ingles" value="i"> Inglês<br>
    <input type="checkbox" name="frances" value="f"> Francês<br>
    <input type="checkbox" name="espanhol" value="e"> Espanhol</p>
<p><input type="submit" value="Enviar" >
    <input type="reset" value="Limpar" ></p>
</form>
</div>
</body>
</html>

```

Exemplo de Formulário:

Acesso ao sistema:

Digite a sua Matrícula:

Digite a sua Senha:

Sexo: Feminino Masculino

Línguas Estrangeiras:

Inglês
 Francês
 Espanhol

Observação: lembre-se de que cada **checkbox** é utilizado para identificar um campo do tipo Sim ou Não e, por isso, cada **checkbox** deve ser tratado separadamente e com descriptores **name** diferentes, uma vez que devemos identificar cada um individualmente para saber se estão ou não selecionados. Esses componentes não se relacionam entre si; sendo assim, haverá a associação independente de cada campo, sendo que, para o campo Inglês, será associado o valor “i” se ele estiver selecionado, e assim por diante para os demais. Pode-se usar o mesmo valor para todos, do tipo “s” para Sim de confirmação ou um valor numérico tal como “1”, para identificar o Sim. Na verdade, só nos interessa saber individualmente qual(is) está(ão) selecionado(s).

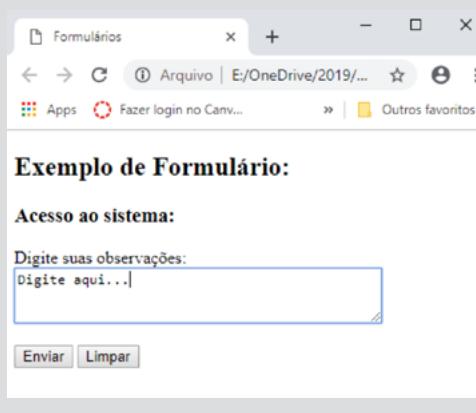
Outros componentes de formulário:

- textarea – Utilizado para a digitação de textos longos, com duas ou mais linhas de texto. A quantidade de linhas visíveis é definida no atributo “rows”, e a quantidade de colunas, no atributo “cols”.

Exemplo:

```
<p>Digite suas observações:<br>
<textarea name="obs" rows="3" cols="45">Digite aqui...</textarea></p>[formulario7.html]
```

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta content="text/html; charset=utf-8">
<title>Formulários</title>
</head>
<body>
<h2> Exemplo de Formulário:</h2>
<div id="t1" align="left">
<h3>Acesso ao sistema:</h3>
<form id="acesso" action="https://www.site.com.br/acesso.php" method="post">
<p>Digite suas observações:<br>
<textarea name="obs" rows="3" cols="45">Digite aqui...</textarea></p>
<p><input type="submit" value="Enviar" >
<input type="reset" value="Limpar" ></p>
</form>
</div>
</body>
</html>
```



Observação: todo o conteúdo de texto digitado pelo usuário será associado ao descritor **name** quando do envio dos dados do formulário ao servidor.

- select – Utilizado para apresentação de um conjunto grande de opções, em que o usuário deverá escolher apenas uma. Esse componente substitui o **rádio** quando temos um conjunto grande de opções que ocuparia um grande espaço da interface com o usuário. Ele é conhecido como **ComboBox** em algumas linguagens de programação.

Exemplo:

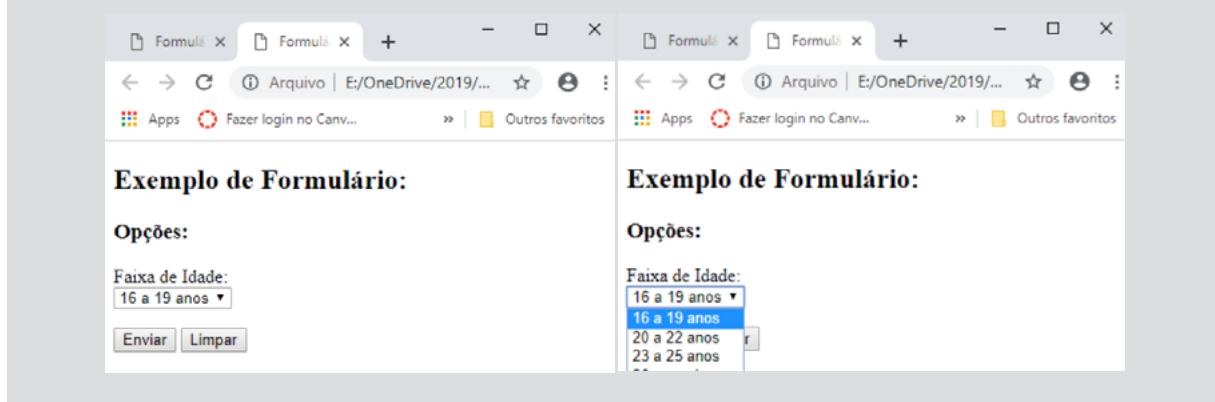
```
<p>Faixa de Idade:<br>
<select name="faixaldade">
<option value="1">16 a 19 anos</option>
<option value="2">20 a 22 anos</option>
<option value="3">23 a 25 anos</option>
<option value="4">26 ou mais</option>
</select></p> [formulario8.html]
```

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta content="text/html; charset=utf-8">
<title>Formulários</title>
</head>
<body>
<h2> Exemplo de Formulário:</h2>
<div id="t1" align="left">
<h3>Opções:</h3>
<form id="acesso" action="https://www.site.com.br/acesso.php" method="post">
<p>Faixa de Idade:<br>
<select name="faixaldade">
<option value="1">16 a 19 anos</option>
<option value="2">20 a 22 anos</option>
<option value="3">23 a 25 anos</option>
<option value="4">26 ou mais</option>
</select></p>
<p><input type="submit" value="Enviar" >
```

```

<input type="reset" value="Limpar" ></p>
</form>
</div>
</body>
</html>

```



Observação: o nome do campo (**name**) faixaidade é único e o valor relacionado quando do envio dos dados do formulário ao servidor será o equivalente ao conteúdo do descriptor **value** da opção escolhida. No exemplo, "1" para a primeira faixa, "2" para a segunda e assim por diante.

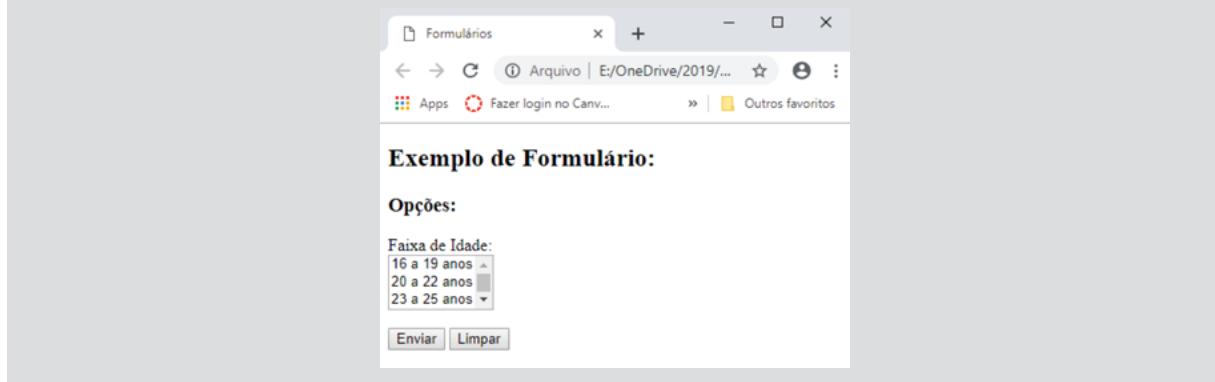
Um campo **select** pode se tornar uma lista, bastando acrescentar o descriptor **size**, que indica o número de linhas que serão visíveis da lista. Como no exemplo anterior esse descriptor não foi definido, o valor padrão nesse caso foi usado, que é "1". Ao acrescentar esse descriptor com valor "3", a lista apresentada em apenas uma linha irá utilizar três linhas:

```

<select name="faixaldade" size="3">
    .....
</select></p>

```

[formulario8b.html]





MIDIA TECA

Acesse a midiateca da Unidade 2 e veja o conteúdo complementar indicado pelo professor sobre a criação de formulários em páginas HTML para definir interfaces de interação com o usuário.



Saiba mais

Para saber mais sobre criação de formulários em HTML e como utilizar componentes específicos para formatar determinadas entradas de dados, leia o Capítulo 2 (p. 33-65) do seguinte livro:

TERUEL, E. C. **HTML 5**: guia prático. 2. ed. São Paulo: Érica, 2014. Minha Biblioteca.

Front-ends para aplicações Web

A criação de uma interface se inicia pela necessidade do sistema, ou seja, é necessário analisar os itens determinados para o sistema a fim de se identificar o componente adequado a cada necessidade. Sendo assim, vamos analisar o problema a seguir.

Uma empresa solicitou o desenvolvimento de uma página para ser incluída em seu site. Essa página é um *front-end* para uma aplicação Web de cadastro de pessoas interessadas em uma vaga de estágio. A empresa solicitou que o cadastro fosse realizado de acordo com as seguintes características:

1. Nome completo do interessado.
2. Sexo.
3. Nível escolar.
4. Áreas de interesse para o estágio, que foram definidas como Internet, Banco de Dados e Programação, sendo que o interessado pode escolher uma ou mais áreas de interesse.
5. Campo para que o candidato possa expressar, por meio de um texto, suas competências e habilidades para a vaga.

Analizando as necessidades, podemos, então, definir quais componentes devem ser usados em cada caso:

1. O nome completo é uma interação por meio da digitação de um texto curto, sendo aconselhável, nesse caso, o uso de um **input** do tipo **text**.
2. Para o sexo, temos duas opções, sendo que apenas uma pode ser escolhida pelo usuário. Essa característica excludente faz com que o botão de rádio ou uma lista definida com apenas uma opção sejam adequados. Como são apenas duas opções, o uso da lista passa a não ser aconselhável, uma vez que o usuário deve realizar dois cliques para mudar uma opção, um para rolar a lista e outro para realizar a escolha. Os botões de rádio (**radio**), portanto, são a escolha mais adequada. As opções então serão Feminino ou Masculino.
3. O nível escolar para estágio deve ser Ensino Fundamental, Ensino Médio, Ensino Superior ou Pós-Graduação, sendo que o usuário deve escolher apenas uma opção. Como no item anterior, temos uma única escolha, e como temos quatro opções e com textos grandes, o uso do botão de rádio pode não ser a melhor opção, por isso usaremos

uma lista com apenas uma linha (**select**). Será incluída ainda uma outra escolha para averiguar se o curso está completo ou incompleto, o que pode inviabilizar o estágio. A melhor opção, nesse caso, seria o uso de um **check** para determinar se está completo (Sim ou Não), mas nós usaremos um botão de rádio com as opções Completo ou Incompleto para que fique mais claro para o usuário.

4. As áreas de interesses são três, mas as opções não são excludentes, o que permite que o usuário possa realizar uma ou mais escolhas ao mesmo tempo. Dessa forma, é aconselhável utilizar um **check** para cada opção.

5. O texto de justificativa para que o interessado possa expressar suas habilidades e competências pode ser grande e ocupar muitas linhas. Sendo assim, é aconselhável o uso de um **textarea**.

Uma vez definidos os componentes adequados para cada uma das necessidades, você deve então preparar um layout que facilite para o usuário e seja o mais intuitivo possível. Como nós já definimos os componentes, devemos montar um formulário com esses componentes. Já vimos que o HTML não apresenta o conteúdo formatado, sendo então necessária uma tabela ou outro artifício que torne a formatação mais fácil. Como forma de facilitar a criação da tabela, sugerimos o uso do Excel, que se utiliza de uma tabela com linhas e colunas semelhante à nossa tabela HTML, inclusive com a opção de mesclar células, da mesma forma que iremos fazer na tabela do front-end. O exemplo a seguir vai nos ajudar como sendo o modelo a ser criado.

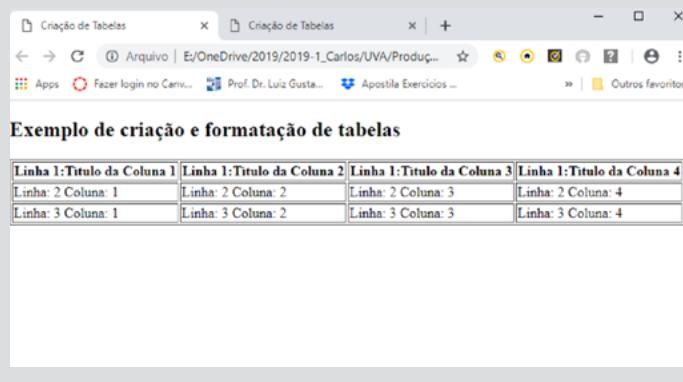
A	B	C	D
CADASTRO DE ESTÁGIO			
2	Nome Completo:		
3	Sexo:	Feminino	Masculino
4	Nível Escolar:	Ensino Fundamental /	
5		Completo Incompleto	
6	Áreas de interesse:	Internet	Banco de Dados
7		Programação	
8	Por quê você deve ser escolhido?		
9			
10			
11			

Já de antemão, podemos verificar que nossa tabela terá 11 linhas e quatro colunas no total, uma vez que usaremos a primeira linha para o título e cinco linhas para o **textarea**. Podemos então iniciar a codificação da nossa tabela, identificando as necessidades de mesclagem de células nela. Vamos aproveitar o exemplo da tabela4.html como base de criação da nossa tabela e, depois que ela estiver de acordo com o nosso layout, iremos realizar as inclusões dos componentes do formulário.

No primeiro exemplo, vamos montar uma tabela com as primeirastrês linhas e quatro colunas.

Exemplo: [formulario&tabela1.html]

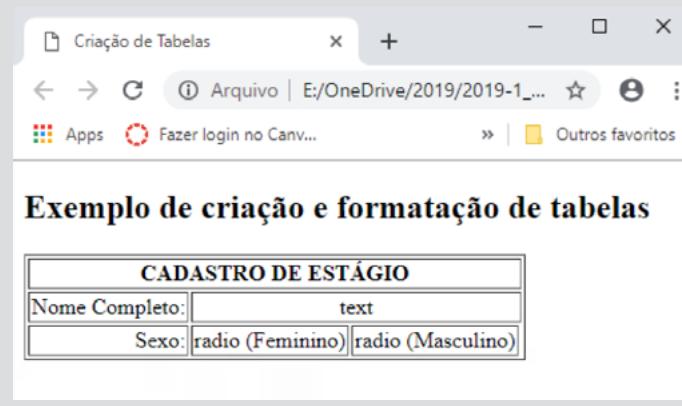
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset="utf-8">
    <title>Criação de Tabelas</title>
</head>
<body>
    <h2> Exemplo de criação e formatação de tabelas</h2>
    <table border="1">
        <tr>
            <th>Linha 1:Título da Coluna 1</th>
            <th>Linha 1:Título da Coluna 2</th>
            <th>Linha 1:Título da Coluna 3</th>
            <th>Linha 1:Título da Coluna 4</th>
        </tr>
        <tr>
            <td>Linha: 2 Coluna: 1</td>
            <td>Linha: 2 Coluna: 2</td>
            <td>Linha: 2 Coluna: 3</td>
            <td>Linha: 2 Coluna: 4</td>
        </tr>
        <tr>
            <td>Linha: 3 Coluna: 1</td>
            <td>Linha: 3 Coluna: 2</td>
            <td>Linha: 3 Coluna: 3</td>
            <td>Linha: 3 Coluna: 4</td>
        </tr>
    </table>
</body>
</html>
```



Agora vamos montar essas três linhas de acordo com o nosso layout, já ajustando e mesclando as células. O tipo de componente foi incluído para facilitar o entendimento e também como orientação para a criação do formulário.

Exemplo: [formulario&tabela2.html]

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Criação de Tabelas</title>
</head>
<body>
    <h2> Exemplo de criação e formatação de tabelas</h2>
    <table border="1">
        <tr>
            <th colspan="4" align="center">CADASTRO DE ESTÁGIO</th>
        </tr>
        <tr>
            <td align="right">Nome Completo:</td>
            <td align="center" colspan="3"> text</td>
        </tr>
        <tr>
            <td align="right">Sexo:</td>
            <td align="center"> radio (Feminino)</td>
            <td align="center"> radio (Masculino)</td>
        </tr>
    </table>
</body>
</html>
```



Aparentemente, a tabela apresenta apenas três colunas, mas, com a inclusão das novas linhas, isso logo irá mudar. Vamos incluir mais três linhas para acomodar o nível escolar e as áreas de interesse.

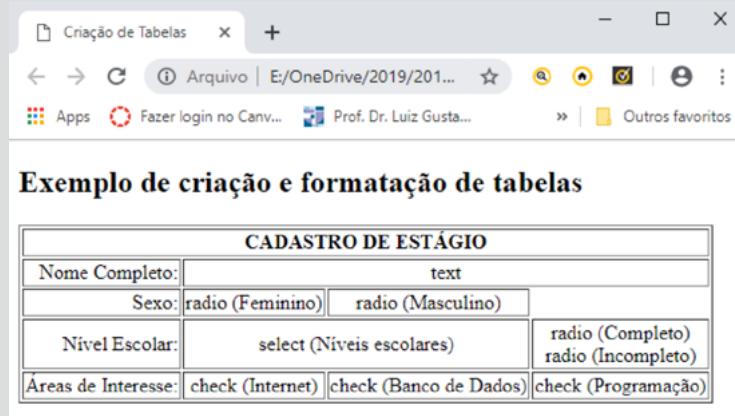
Exemplo: [formulario&tabela3.html]

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Criação de Tabelas</title>
</head>
<body>
    <h2> Exemplo de criação e formatação de tabelas</h2>
    <table border="1">
        <tr>
            <th colspan="4" align="center">CADASTRO DE ESTÁGIO</th>
        </tr>
        <tr>
            <td align="right">Nome Completo:</td>
            <td align="center" colspan="3"> text</td>
        </tr>
        <tr>
            <td align="right">Sexo:</td>
            <td align="center"> radio (Feminino)</td>
            <td align="center"> radio (Masculino)</td>
        </tr>
        <tr rowspan="2">
            <td align="right">Nível Escolar:</td>
            <td align="center" colspan="2"> select (Níveis escolares)</td>
            <td align="center">radio (Completo)<br>radio (Incompleto)</td>
        </tr>
        <tr>
            <td align="right">Áreas de Interesse:</td>
            <td align="center"> check (Internet)</td>
        </tr>
    </table>
</body>
```

```

<td align="center"> check (Banco de Dados)</td>
align="center"> check (Programação)</td>
</tr>
</body>
</html>

```



Agora nossa tabela já apresenta as quatro colunas. Vamos então incluir as cinco linhas referentes ao **textarea**, que, na verdade, será apenas mais uma linha na tabela, uma vez que o próprio **textarea** irá possuir as cinco linhas de texto.

Exemplo: [formulario&tabela4.html]

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta content="text/html; charset=utf-8">
    <title>Criação de Tabelas</title>
</head>
<body>
    <h2> Exemplo de criação e formatação de tabelas</h2>
    <table border="1">
        <tr>
            <th colspan="4" align="center">CADASTRO DE ESTÁGIO</th>
        </tr>
        <tr>
            <td align="right">Nome Completo:</td>
            <td align="center" colspan="3"> text</td>
        </tr>

```

```

<tr>
    <td align="right">Sexo:</td>
    <td align="center"> radio (Feminino)</td>
    <td align="center"> radio (Masculino)</td>
</tr>
<tr rowspan="2">
    <td align="right">Nível Escolar:</td>
    <td align="center" colspan="2"> select (Níveis escolares)</td>
    <td align="center">radio (Completo)<br>radio (Incompleto)</td>
</tr>
<tr>
    <td align="right">Áreas de Interesse:</td>
    <td align="center"> check (Internet)</td>
    <td align="center"> check (Banco de Dados)</td>
    <td align="center"> check (Programação)</td>
</tr>
<tr>
    <td align="right">Por quê você deve ser escolhido?:</td>
    <td align="center" colspan="3"> textarea</td>
</tr>
</body>
</html>

```

Criação de Tabelas

Exemplo de criação e formatação de tabelas

CADASTRO DE ESTÁGIO			
Nome Completo:	text		
Sexo:	radio (Feminino)	radio (Masculino)	
Nível Escolar:	select (Níveis escolares)		radio (Completo) radio (Incompleto)
Áreas de Interesse:	check (Internet)	check (Banco de Dados)	check (Programação)
Por quê você deve ser escolhido?:	textarea		

Nossa tabela está pronta, e já podemos iniciar a criação do formulário, bastando agora substituirmos as indicações dos componentes pelos componentes HTML identificados.

Nossos componentes serão:

1. Nome Completo:

```
<input type="text" size="50" name="nome"> -
```

Caixa de texto com 50 espaços visíveis (**size**). O descriptor **size** não limita o número de caracteres, servindo para determinar o tamanho do componente de texto.

2. Sexo: Coluna 1:

```
<input type="radio" name="sexo" value="f" checked> Feminino
```

Coluna 2:

```
<input type="radio" name="sexo" value="m"> Masculino
```

3. Nível Escolar:

```
<select name="nivelescolar">
<option value="1">Ensino Fundamental</option>
    <option value="2"> Ensino Médio</option>
    <option value="3">Ensino Superior</option>
    <option value="4">Pós-Graduação</option>
</select>
```

Botões de rádio:

```
<input type="radio" name="situacao" value="1" checked>Completo<br>
<input type="radio" name="situacao" value="2">Incompleto
```

4. Áreas de interesse:

```
<input type="checkbox" name="internet" value="i">Internet  
<input type="checkbox" name="bd" value="b">Banco de Dados  
<input type="checkbox" name="programacao" value="p">Programação
```

5. Justificativa:

```
<textarea name="justificativa" rows="5" cols="50">Digite aqui...</textarea>
```

Devemos também incluir os botões de Envio e de Limpeza do formulário:

```
<tr>  
<td colspan="2" align="center"><input type="submit" value="Enviar"></td>  
<td colspan="2" align="center"><input type="reset" value="Limpar"></td>  
</tr>
```

Após substituirmos cada identificador pelo componente HTML adequado, temos:

Exemplo: [formulario&tabela5.html]

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta content="text/html; charset=UTF-8" />
<title>Criação de Tabelas</title>
</head>
<body>
<h2>Exemplo de criação e formatação de tabelas</h2>
<form id="cadastro" action="https://www.empresacom.br/castratoestagio.php" method="post">
<table border="1">
<tr>
<th colspan="4" align="center">CADASTRO DE ESTÁGIO</th>
</tr>
<tr>
<td align="right">Nome Completo:</td>
<td align="left" colspan="3"><input type="text" size="50" name="nome"></td>
</tr>
<tr>
<td align="right">Sexo:</td>
<td align="center"><input type="radio" name="sexo" value="f" checked> Feminino</td>
<td align="center"><input type="radio" name="sexo" value="m"> Masculino</td>
</tr>
<tr rowspan="2">
<td align="right">Nível Escolar:</td>
<td align="center" colspan="2">
<select name="nivelescolar">
<option value="1">Ensino Fundamental</option>
<option value="2"> Ensino Médio</option>
<option value="3">Ensino Superior</option>
<option value="4">Pós-Graduação</option>
</select></td>
<td align="center">
<input type="radio" name="situacao" value="1" checked>Completo<br>
<input type="radio" name="situacao" value="2">Incompleto</td>
</tr>

```

```

<tr>
    <td align="right">Áreas de Interesse:</td>
    <td align="center">
        <input type="checkbox" name="internet" value="i">Internet</td>
    <td align="center">
        <input type="checkbox" name="bd" value="b">Banco de Dados</td>
    <td align="center">
        <input type="checkbox" name="programacao" value="p">Programação</td>
    </td>
</tr>
<tr>
    <td align="right">Por quê você deve ser escolhido?:</td>
    <td align="center" colspan="3">
        <textarea name="justificativa" rows="5" cols="50">Digite aqui...</textarea>
    </td>
</tr>
<tr>
    <td colspan="2" align="center"><input type="submit" value="Enviar"></td>
    <td colspan="2" align="center"><input type="reset" value="Limpar"></td>
</tr>
</form>
</body>
</html>

```

Criação de Tabelas

Exemplo de criação e formatação de tabelas

CADASTRO DE ESTÁGIO		
Nome Completo:		
Sexo:	<input checked="" type="radio"/> Feminino <input type="radio"/> Masculino	<input checked="" type="radio"/> Completo <input type="radio"/> Incompleto
Nível Escolar:	<input type="button" value="Ensino Fundamental ▾"/>	<input checked="" type="radio"/> Completo <input type="radio"/> Incompleto
Áreas de Interesse:	<input type="checkbox"/> Internet <input type="checkbox"/> Banco de Dados <input type="checkbox"/> Programação	Digite aqui...
Por quê você deve ser escolhido?:	Digite aqui...	
<input type="button" value="Enviar"/> <input type="button" value="Limpar"/>		

Para finalizarmos, vamos retirar as linhas de contorno, retirando o descritor **border** da tabela.

CADASTRO DE ESTÁGIO

Nome Completo:

Sexo: Feminino Masculino

Nível Escolar:

Áreas de Interesse: Internet Banco de Dados Programação

Completo
Incompleto

Por quê você deve ser escolhido?:

Até este momento, nós não nos preocupamos com a aplicação de estilos em nossas páginas, com a inclusão de diferentes fontes, tamanhos e cores. Aprender a realizar tais configurações e como criar estilos capazes de estruturar grandes sites, com centenas ou milhares de páginas, são assuntos que serão tratados em outro momento.



MEDIATECA

Acesse a midiateca da Unidade 2 e veja o conteúdo complementar indicado pelo professor sobre a criação de formulários em páginas HTML para a criação de interfaces de interação com o usuário com tabelas e formulários.



Saiba mais

Para saber mais sobre criação de tabelas e formulários em HTML com configuração de descritores, leia o Capítulo 1 (p. 23-45) do seguinte livro:

SEGURADO, V. S. **Projeto de interface com o usuário**. São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.



NA PRÁTICA

Você já deve ter notado que toda vez que você realiza um cadastro ou preenche qualquer tipo de dado em aplicações Web está preenchendo um formulário. Os dados introduzidos nesse formulário fazem parte da interação do sistema Web com o usuário. Esses dados podem então ser analisados pela aplicação cliente (*front-end*) por meio de funções incluídas no próprio código (JavaScript) e, posteriormente, enviados ao servidor (*back-end*). Após o recebimento dos dados pelo servidor, os dados são tratados e é enviada uma resposta ao usuário, que pode ser apenas uma página simples contendo alguma mensagem, tal como uma confirmação de um cadastro. Mas também pode ser retornada uma página exclusiva, com dados e informações específicas para cada usuário, tais como as suas informações em um determinado sistema. Um bom exemplo é o seu acesso ao sistema acadêmico, que, após o preenchimento de login (matrícula) e senha, retorna uma página exclusiva com as suas informações particulares armazenadas nesse sistema.

Resumo da Unidade 2

Nesta unidade, você pôde entender como criar tabelas e divisões para organizar a apresentação de uma página, além de poder utilizar esse conhecimento na criação de formulários HTML com os componentes de interações para a criação de interfaces com o cliente organizadas e com melhor apresentação em aplicações Web.



CONCEITO

Nesta unidade, destacaram-se os passos necessários para a criação e formatação de formulários HTML de interação com o usuário, que são fundamentais para o desenvolvimento de front-ends das aplicações Web.

Referências

FLATSCHART, F. **HTML 5:** embarque imediato. Rio de Janeiro: Brasport, 2018. Biblioteca Virtual.

SEGURADO, V. S. **Projeto de interface com o usuário.** São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

TERUEL, E. C. **HTML 5:** guia prático. 2. ed. São Paulo: Érica, 2014. Minha Biblioteca.

UNIDADE 3

Padronização de sites
com uso de CSS

INTRODUÇÃO

A construção de pequenos sites nos permite realizar mudanças visuais de estilo em pouco tempo, mas isso não ocorre com grandes sites, que podem possuir milhares de páginas. Sites grandes necessitariam, então, que cada página fosse alterada separadamente e que essa tarefa fosse realizada por uma equipe de programação, e poderia levar semanas ou mesmo meses para completar a tarefa de reestilizar o site. O uso de estilos permite, então, a realização da tarefa de reestilização em pouco tempo, alterando as tags de maneira apropriada ou a terão os arquivos de configuração de estilos do site.



OBJETIVO

Nesta unidade, você será capaz de:

- Desenvolver e criar sites padronizados com a utilização de folhas de estilos para aplicações Web.

Conceitos fundamentais para a aplicação de folhas de estilos (CSS), definição de classes, seletores, unidades de medidas e cores

CSS (cascading style sheets (“folha de estilo em cascata”))

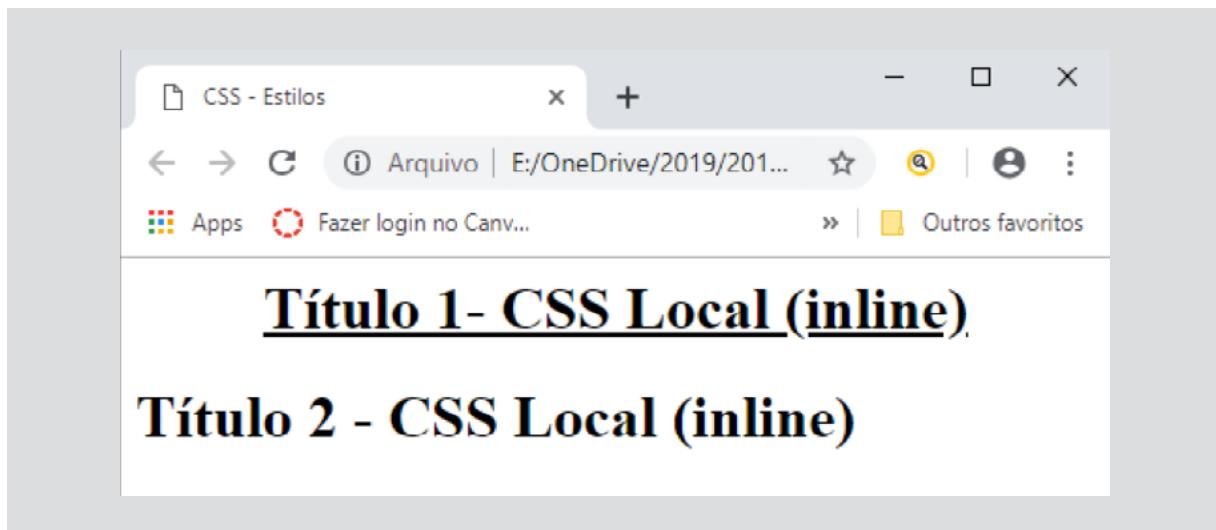
CSS é uma forma de padronização de sites que permite a personalização de estilos do site como um todo. Essa padronização é realizada de três formas:

1. Local (*inline*) – Permite a determinação do estilo de forma específica sobre um determinado componente da página. É utilizado para determinar estilos específicos de um componente que se diferencia da padronização global do site. É importante observar que os estilos para sublinhar e centralizar o texto do componente `<h1>` estão definidos diretamente na tag HTML do componente:

```
<h1 style="text-decoration:underline; text-align: center">Título 1 - CSS Local (inline)</h1>
```

Exemplo: css_01.html

```
<html>
  <head>
    <title>CSS - Estilos</title>
  </head>
  <body>
    <h1 style="text-decoration:underline; text-align: center">Título 1- CSS Local (inline)</h1>
    <h1>Título 2 - CSS Local (inline)</h1>
  </body>
</html>
```



Você pode observar que o primeiro componente, além de ser um cabeçalho do tipo <h1>, possui os estilos de sublinhado e centralizado, mas o segundo componente <h1> não é afetado pela padronização.

2. Incorporado (*embedded*) – Permite a determinação do estilo para um ou mais componentes em uma mesma página. É utilizado para determinar um conjunto de estilos diferentes do padrão do site, para componentes específicos em algumas páginas. Isso permite que essas páginas específicas possuam características próprias e diferentes do padrão do site. Note que a determinação dos padrões passa a ser realizada na área do cabeçalho da página (*head*), e não diretamente no componente como no padrão local. Perceba também que existe a identificação do componente, além de uma relação de padrões que serão aplicados a todos os componentes do mesmo tipo dentro dessa página. É necessário que o padrão seja definido por meio de uma tag própria <style>, que deve estar presente na área do cabeçalho. Nesse exemplo, ambos os componentes <h1> serão afetados, uma vez que a padronização incorporada se reflete sobre todos os componentes do mesmo tipo nessa página.

Observe ainda que agora temos apenas a determinação da padronização e o conteúdo a ser exibido será determinado na área do corpo (*body*):

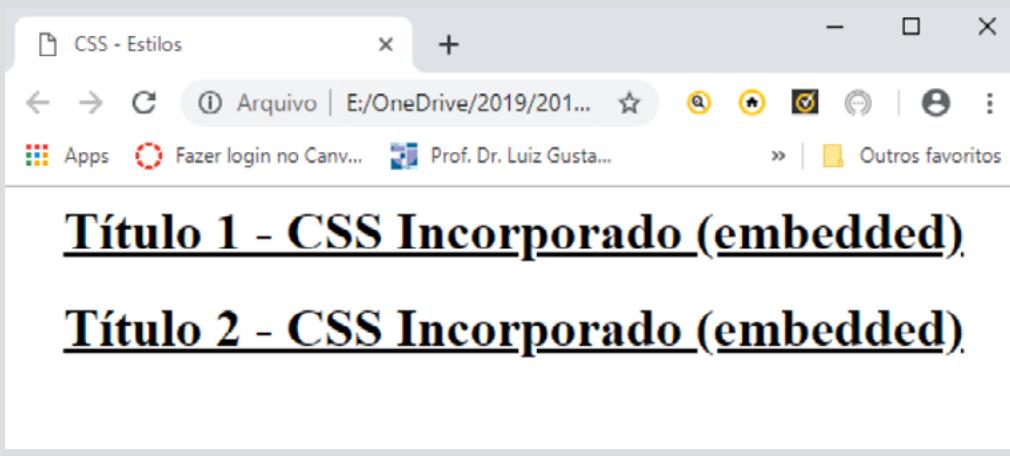
```
<style type="text/css">
  h1 { text-decoration : underline ; text-align : center; }
</style>
```

Exemplo: css_02.html

```

<html>
  <head>
    <title>CSS - Estilos</title>
    <style type="text/css">
      h1 { text-decoration : underline ; text-align : center; }
    </style>
  </head>
  <body>
    <h1>Título 1 - CSS Incorporado (embedded)</h1>
    <h1>Título 2 - CSS Incorporado (embedded)</h1>
  </body>
</html>

```



Nesse exemplo, ambos os componentes foram afetados pela padronização, uma vez que ela determina que o padrão seja aplicado a todos os componentes do tipo `<h1>` da página.

3. Externa (*linked*) – Permite a determinação do estilo para um ou mais componentes em diferentes páginas. É usado para determinar a padronização padrão do site, em que a maioria dos componentes (com exceção dos componentes com padrões específicos) seguirá a padronização global do site. É a forma mais comum e eficiente de uso em grandes sites, em que as configurações dos componentes são realizadas em um arquivo externo específico (arquivo.css), que contém as padronizações globais do site. Esse arquivo externo é lido e suas configurações são incorporadas a todas as páginas que o carregarem, bastando informar o link do arquivo com as configurações a ser carregado. Qualquer alteração em uma configuração desse arquivo impactará todas as páginas do site instantaneamente. Essa característica é responsável por realizar uma manutenção

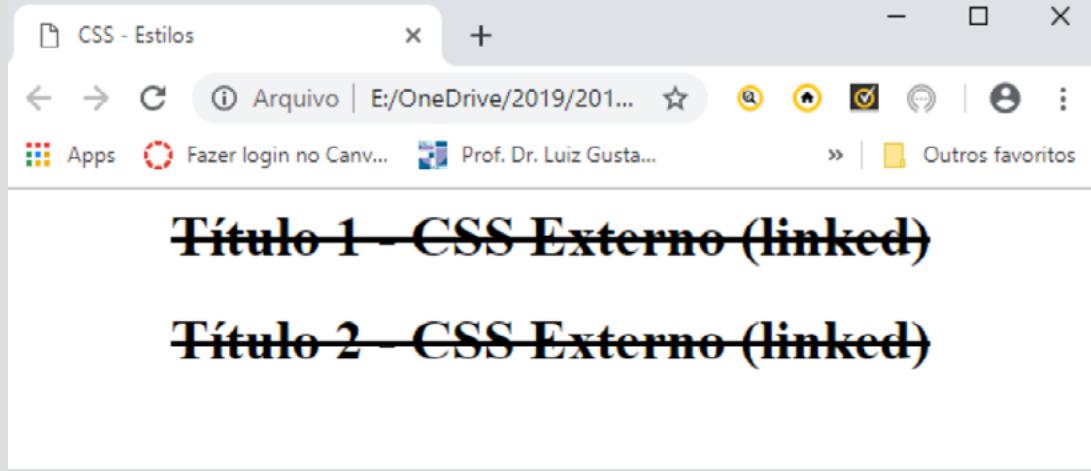
ou alteração em grandes sites de forma simples e rápida, sem a necessidade de uma grande equipe. Esse estilo de padronização deve ser definido junto com a tag link na definição da página na área de cabeçalho (*head*), conforme o exemplo:

```
<link rel="stylesheet" type="text/css" href="arq_estilo_1.css">
```

Porém, também é necessário criar o arquivo CSS (arq_estilo_1.css):

```
h1 {text-decoration : line-through; text-align : center; }
```

Exemplo: css_03.html



The screenshot shows a web browser window titled "CSS - Estilos". The address bar indicates the file is located at "E:/OneDrive/2019/201...". The page content displays two h1 headings: "Título 1 - CSS Externo (linked)" and "Título 2 - CSS Externo (linked)". Both titles have a horizontal line through them, indicating the CSS rule "text-decoration : line-through;" has been applied.

```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_1.css">
</head>
<body>
<h1>Título 1 - CSS Externo (linked)</h1>
<h1>Título 2 - CSS Externo (linked)</h1>
</body>
</html>
```

Agora ambos os componentes possuem a mesma padronização e todas as páginas que incluírem o arquivo CSS terão a mesma padronização.

Uma coisa que você pode estar pensando é: como se determina a prioridade sobre os diferentes tipos de estilo?

Isso é simples, como o padrão externo é o mais abrangente, ele deve ser o último na prioridade, caso contrário não poderíamos criar padrões específicos locais. Dessa forma, a prioridade é determinada de acordo com a seguinte ordem:



Quando temos configurações diferentes para um mesmo componente, a prioridade é sempre das configurações locais, depois as configurações incorporadas e por último as configurações externas; essa ordem garante que configurações específicas para um componente possam ser utilizadas.

É possível definir a padronização de um conjunto de componentes ao mesmo tempo, desde que possuam a mesma configuração, tal como no exemplo a seguir:

```
h1, h2, h3 { text-decoration : underline ; text-align : center ; }
```

Perceba que os componentes h1, h2 e h3 possuirão a mesma padronização.

Definição de classes

A definição de classes permite que sejam definidos padrões que podem ser utilizados em qualquer componente. A definição de uma classe deve ser precedida de um ponto (.), um nome de identificação e da configuração do padrão. O exemplo a seguir apresenta a definição de uma classe:

```
.texto_negrito { font-weight : bold ; }
```

Perceba o ponto antes da identificação da classe. O identificador texto_negrito determina que a fonte a ser utilizada no componente seja apresentada em negrito. A criação de classes pode ser usada em padrões incorporados ou externos, uma vez que o padrão local só se aplica ao componente onde o padrão foi definido, não permitindo a criação de classes.

Para a utilização, basta informar a classe (class) a ser utilizada no componente da página, dentro da tag, conforme exemplo:

```
<p class="texto_negrito">Parágrafo com o texto em negrito, de acordo com o padrão da classe.</p>
```

Seletores

São usados para padronizar âncoras e links. Você pode evitar que um link apareça sublinhado, por exemplo:

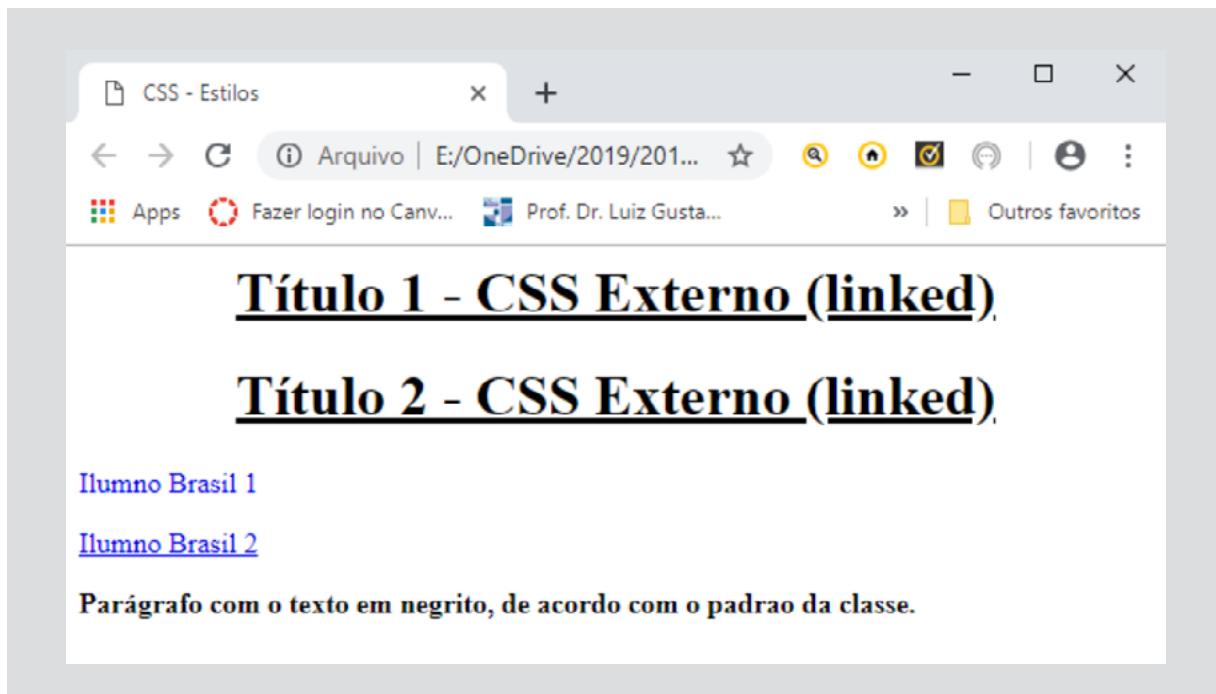
```
a.sem_sublinhado {text-decoration : none; }
```

No exemplo a seguir, podemos avaliar o uso de classes e seletores na prática:

Exemplos: css_04.html e arq_estilo_2.css:

```
<html>
  <head>
    <title>CSS - Estilos</title>
    <link rel="stylesheet" type= "text/css" href="arq_estilo_2.css">
  </head>
  <body>
    <h1>Título 1 - CSS Externo (linked)</h1>
    <h1>Título 2 - CSS Externo (linked)</h1>
    <p><a class="sem_sublinhado" href="http://ilumno.com/pt">Ilumno Brasil 1</a></p>
    <p><a href="http://ilumno.com/pt">Ilumno Brasil 2</a></p>
    <p class="texto_negrito">Parágrafo com o texto em negrito, de acordo com o padrão da classe.</p>
  </body>
</html>
```

```
h1 {text-decoration : underline; text-align : center; }
.texto_negrito { font-weight : bold; }
.a.sem_sublinhado {text-decoration:none; }
```



Note que, no primeiro link, foi aplicado o seletor `sem_sublinhado` e, dessa forma, o link não aparece sublinhado.

Você pode omitir o nome da classe no seletor de forma que todos os links passem a ter o mesmo padrão definido, omitindo a classe, como no exemplo a seguir:

Exemplos: `css_05.html` e `arq_estilo_3.css`:

```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_3.css">
</head>
<body>
<h1>Título 1 - CSS Externo (linked)</h1>
<h1>Título 2 - CSS Externo (linked)</h1>
<p><a href="http://ilumno.com/pt">Ilumno Brasil 1</a></p>
<p><a href="http://ilumno.com/pt">Ilumno Brasil 2</a></p>
<p class="texto_negrito">Parágrafo com o texto em negrito, de acordo com o padrão da classe.</p>
</body>
</html>
```

```
h1 {text-decoration : underline; text-align : center; }  
.texto_negrito { font-weight : bold; }  
a {text-decoration : none; }
```

The screenshot shows a browser window titled "CSS - Estilos". The address bar says "Arquivo | E:/OneDrive/2019/2019-1_Carlos/U...". Below the address bar are links for "Fazer login no Canva...", "Prof. Dr. Luiz Gusta...", and "Outros favoritos". The main content area displays two bold titles: "**Título 1 - CSS Externo (linked)**" and "**Título 2 - CSS Externo (linked)**". Underneath the titles are two blue links: "Ilumno Brasil 1" and "Ilumno Brasil 2". A bold text block follows: "Parágrafo com o texto em negrito, de acordo com o padrao da classe."

Note que agora todos os links não estão sublinhados, sendo que a configuração foi determinada para todos os links da página.

Unidades de medidas

Existe um grande conjunto de propriedades de configuração que utilizam unidades de medidas. Essas propriedades podem definir posicionamentos e configurações de espaços na página. Essas unidades podem ser absolutas ou relativas: as posições absolutas determinam o posicionamento determinado e inalterado, independentemente do tamanho da janela, já o posicionamento relativo depende do tamanho da janela do navegador.

Unidades absolutas:

cm	centímetros
mm	milímetros
in	polegada
pt	pontos (equivalente a 1/72 polegadas)
pc	paica (equivalente a 12 pontos)

Unidades relativas:

%	porcentagem
em	esta propriedade possui valor igual a n vezes o tamanho da propriedade (por exemplo font-size) do elemento.
px	pixel

Exemplo de uso de unidades de medida: css_06.html e arq_estilo_4.css:



Faça alterações nos valores atribuídos às medidas de posição e tamanho da imagem e analise os resultados ao atualizar o navegador. Não se esqueça de salvar as alterações no arquivo CSS.

Observação: se você definir uma posição padrão de uma imagem em dezenas ou centenas de páginas em um site, a imagem pode ser redimensionada uma única vez no arquivo CSS externo e as alterações serão automaticamente em todo o site.

Cores

As cores no HTML seguem o padrão RGB; dessa forma, os valores podem ser definidos em hexadecimal combinando os valores das três cores básicas (vermelha, verde/*lime* e azul) para compor uma determinada cor. A formatação padrão é #RRGGBB, em que:

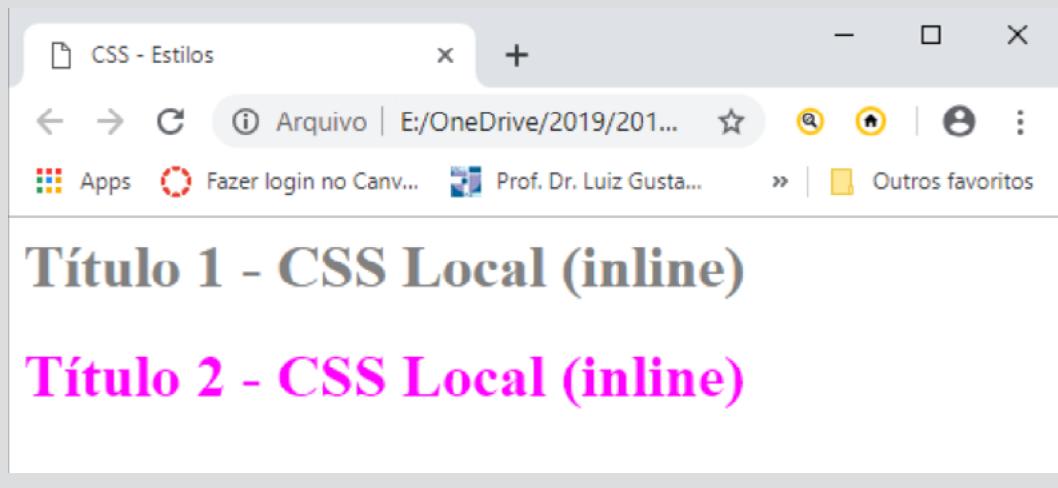
- #FF0000 – Define a cor vermelha total e nada de verde ou azul.
- #00FF00 – Define a cor verde (*lime*) total e nada de vermelho ou azul.
- #0000FF – Define a cor azul total e nada de vermelho ou verde.

Assim, podemos combinar as quantidades de cada cor básica para formar novas cores: #FF7722. Também é possível utilizar nomes para as cores mais comuns, tal como os exemplos apresentados na tabela a seguir:

White		#FFFFFF	Red		#FF0000
Black		#000000	Maroon		#800000
Yellow		#FFFF00	Silver		#C0C0C0
Aqua		#00FFFF	Gray		#808080
Lime		#00FF00	Fuchsia		#FF00FF
Green		#008000	Purple		#800080
Olive		#808000	Blue		#0000FF
Teal		#008080	Navy		#000080

Exemplo de padronização de cores: css_07.html

```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_4.css">
</head>
<body>
<h1 style="color: #808080;">Título 1 - CSS Local (inline)</h1>
<h1 style="color: fuchsia;">Título 2 - CSS Local (inline)</h1>
</body>
</html>
```



MEDIATECA

Acesse a mediateca da Unidade 3 e veja o conteúdo complementar indicado pelo professor sobre o uso adequado dos estilos em cascata.



Saiba mais

Para saber mais sobre a padronização de sites, leia o Capítulo 4 (p. 70-94) do seguinte livro:

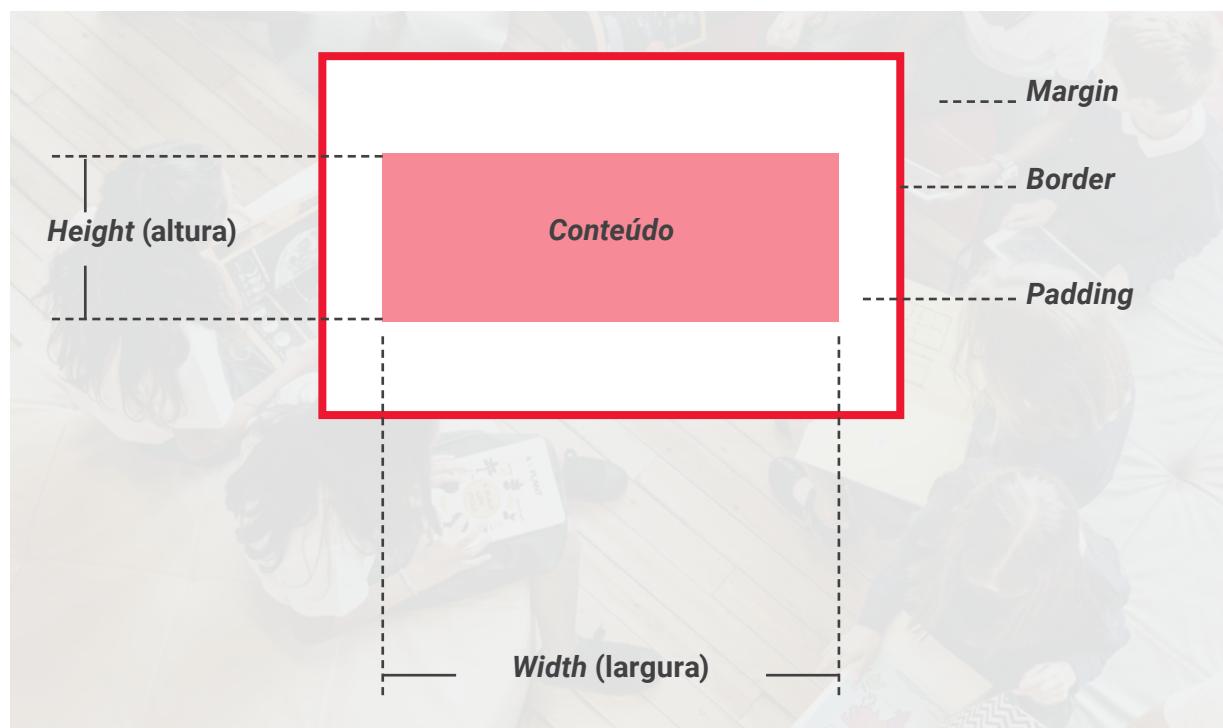
MILETTO, E.; BERTAGNOLLI, S. **Desenvolvimento de software II:** introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. Minha Biblioteca.

Padronização da estrutura da página (caixas, margens, bordas, espaçamento, posição e cursores)

Caixas

A apresentação de um conteúdo de uma página em um navegador é realizada mediante um conjunto de caixas, que podem ser configuradas de forma a melhorar a visualização do usuário. Essas caixas delimitam regiões da página tais como o espaçamento (*padding*), que é responsável por delimitar a distância entre o conteúdo central e a borda (*border*) e por uma margem (*margin*). Os elementos *padding* e *margin* de uma página são transparentes e não são visualizados pelo usuário, a não ser pela diferença de espaços entre o conteúdo e essas caixas.

Podemos observar a existência dessas caixas por meio da figura a seguir:



Margem

Margin (margem) – Essa propriedade pode ser configurada para especificar a distância da borda para os cantos do navegador. Essa configuração pode ser única para todas as margens ou específica para cada margem, sendo as margens divididas em:

- margin-top: margem superior.
- margin-right: margem direita.
- margin-bottom: margem inferior.
- margin-left: margem esquerda.

Os valores podem ser automáticos (nesse caso, o navegador utilizará um valor padrão) ou especificados em algum tipo de medida absoluta ou relativa (%; em; px; cm; mm; in; pt; pc). Você pode usar de diferentes formas, de acordo com os exemplos a seguir:

- <div style= "margin: 10px 20px 10px 20px"> é o mesmo que: <div style= "margin-top: 10px; margin-right: 20px; margin-bottom: 10px; margin-left: 20px">
- <div style= "margin: 10px;"> é o mesmo que: <div style= "margin: 10px 10px 10px 10px">

Observe que as margens são definidas de acordo com o sentido de um relógio, começando pela superior (*top*), direita (*right*), inferior (*bottom*) e esquerda (*left*).

Borda

Border (borda) – Essa propriedade pode ser configurada para especificar a borda da página a ser exibida pelo navegador. A configuração da borda pode especificar a cor, o estilo ou o tamanho de forma global (para todas as bordas) ou específica para cada uma das quatro bordas e pode ser usada em diversos elementos.

<i>border</i>	<i>border-color</i>	<i>border-style</i>	<i>border-width</i>
border-top	border-top-color	border-top-style	border-top-width
border-right	border-right-color	border-right-style	border-right-width
border-bottom	border-bottom-color	border-bottom-style	border-bottom-width
border-left	border-left-color	border-left-style	border-left-width

Cores (*color*) podem ser configuradas de acordo com as configurações já vistas.

Principais propriedades:

Estilo (*style*) pode ser configurado de acordo com os seguintes valores:

- *none*: nenhuma borda.
- *dotted*: pontilhada.
- *dashed*: tracejada.
- *solid*: linha sólida.
- *inset*: baixo-relevo.
- *outset*: alto-relevo.
- *double*: linha dupla.

Tamanho (*width*) pode ser configurado de acordo com os seguintes valores:

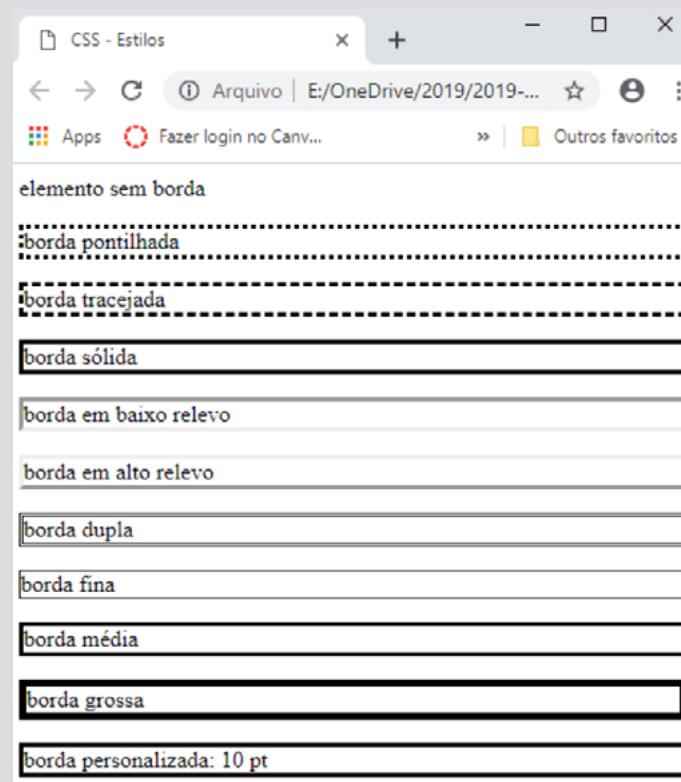
- *thin*: fina.
- *medium*: média.
- *thick*: grossa.
- Unidades de medida: %; em; px; cm; mm; in; pt; pc.

A seguir, css08.html e arq_estilo_5.css ilustram alguns exemplos:

```
<html>
<head>
    <title>CSS - Estilos</title>
    <link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
</head>
<body>
    <p class="sem_borda" />elemento sem borda</p>
    <p class="borda_pontilhada" />borda pontilhada</p>
    <p class="borda_tracejada" />borda tracejada</p>
    <p class="borda_sólida" />borda sólida</p>
    <p class="borda_baixorelevo" />borda em baixo relevo</p>
    <p class="borda_altorelevo" />borda em alto relevo</p>
    <p class="borda_dupla" />borda dupla</p>
    <p class="borda_fina" />borda fina</p>
    <p class="borda_media" />borda média</p>
```

```
<p class="borda_grossa" />borda grossa</p>
<p class="borda_personalizada" />borda personalizada: 10 pt</p>
</body>
</html>
```

```
h1 { text-decoration : underline ; text-align : center ; }
.texto_negrito { font-weight : bold ; }
a.sem_sublinhado {text-decoration : none; }
p.sem_borda {border-style : none; }
p.borda_pontilhada {border-style : dotted; }
p.borda_tracejada {border-style : dashed; }
p.borda_solidas {border-style : solid; }
p.borda_baixorelevo {border-style : inset; }
p.borda_altorelevo {border-style : outset; }
p.borda_dupla {border-style : double; }
p.borda_fina {border-style : solid; border-width : thin; }
p.borda_media {border-style : solid; border-width : medium; }
p.borda_grossa {border-style : solid; border-width : thick; }
p.borda_personalizada {
    border-style : solid;
    border-top-width : 10 pt ;
    border-right-width : 10 pt ;
    border-bottom-width : 10 pt ;
    border-left-width : 10 pt ;
}
```



Espaçamento

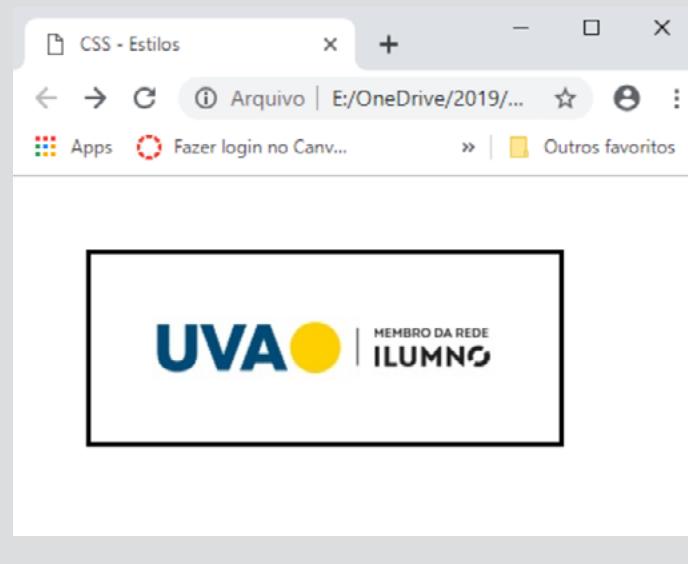
Padding (espaçamento) – Essa propriedade pode ser configurada para especificar a distância entre o conteúdo central e a borda. Essa configuração pode ser única para todos os espaçamentos ou específica para cada espaçoamento, sendo os espaçamentos divididos em:

- padding-top.
- padding-right.
- padding-bottom.
- padding-left.

Valores em unidades de medida: %; em; p; cm; mm; in; pt; pc.

Os exemplos a seguir, css09.html e css10.html, ilustram a forma de uso do espaçoamento com uma imagem:

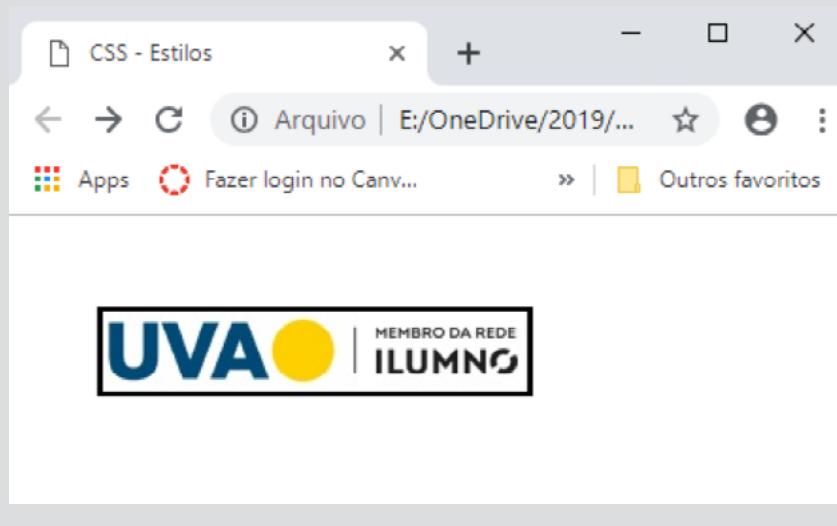
```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
</head>
<body>
<p></p>
</body>
</html>
```



```

<html>
  <head>
    <title>CSS - Estilos</title>
    <link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
  </head>
  <body>
    <p></p>
  </body>
</html>

```



Observe a diferença de espaçamento entre a borda e a imagem.

Posicionamento

Position (posicionamento) – Essa propriedade é importante para a configuração de imagens, por exemplo, e pode determinar não apenas o posicionamento da imagem, mas também a prioridade de exibição entre imagens que se sobreponem.

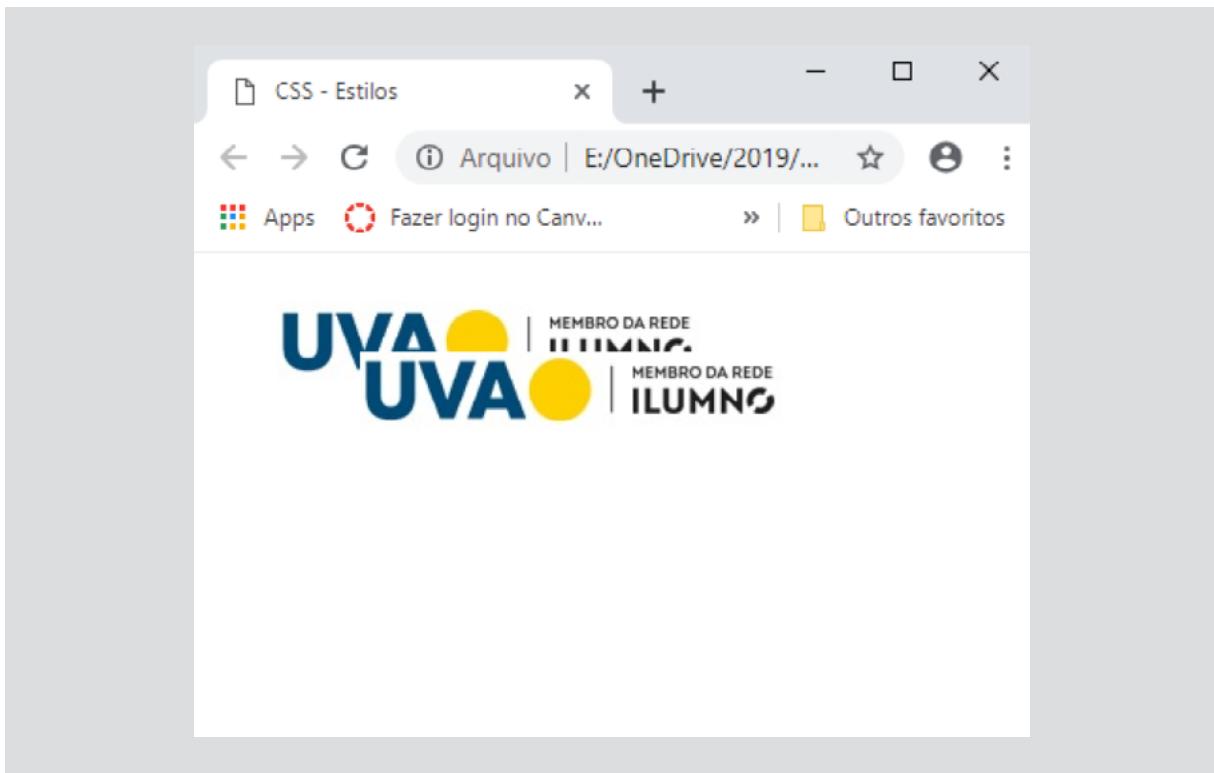
Principais propriedades:

- *static*: o elemento é exibido na ordem em que foi posicionado na página, não levando em conta outras propriedades de posicionamento.
- *relative*: a posição obtida é relativa ao seu posicionamento na página.
- *absolute*: o elemento é posicionado de acordo com as propriedades "*top*", "*bottom*", "*right*" e/ou "*left*".
- *fixed*: o elemento é posicionado de acordo com as propriedades "*top*", "*bottom*", "*right*" e/ou "*left*"; a imagem fica fixa nessa posição, mesmo que a janela sofra rolagem vertical.

- *top*:
 - auto: o navegador utiliza a distância padrão do elemento para o topo.
 - unidades de medida: %; em; px; cm; mm; in; pt; pc.
- *bottom* (não é recomendado o seu uso, uma vez que o posicionamento de elementos em janelas pode ser definido apenas pelas propriedades “*top*” e “*left*”).
auto: o navegador utiliza a distância padrão do elemento para “*bottom*”.
- unidades de medida: %; em; px; cm; mm; in; pt; pc.
- *left*:
 - auto: o navegador utiliza a distância padrão do elemento para “*left*”.
 - unidades de medida: %; em; px; cm; mm; in; pt; pc.
- *right* (não é recomendado o seu uso, uma vez que o posicionamento de elementos em janelas pode ser definido apenas pelas propriedades: “*top*” e “*left*”).
auto: o navegador utiliza a distância padrão do elemento para “*right*”.
- unidades de medida: %; em; px; cm; mm; in; pt; pc.
- *z-index*: um elemento com valor inteiro que determina a ordem de exibição de elementos sobrepostos. Esse recurso pode fornecer uma “sensação” de tridimensionalidade nos elementos sobrepostos. Valores maiores se sobrepõem a valores menores.

O exemplo css_11.html ilustra um exemplo de sobreposição de duas imagens em uma mesma página:

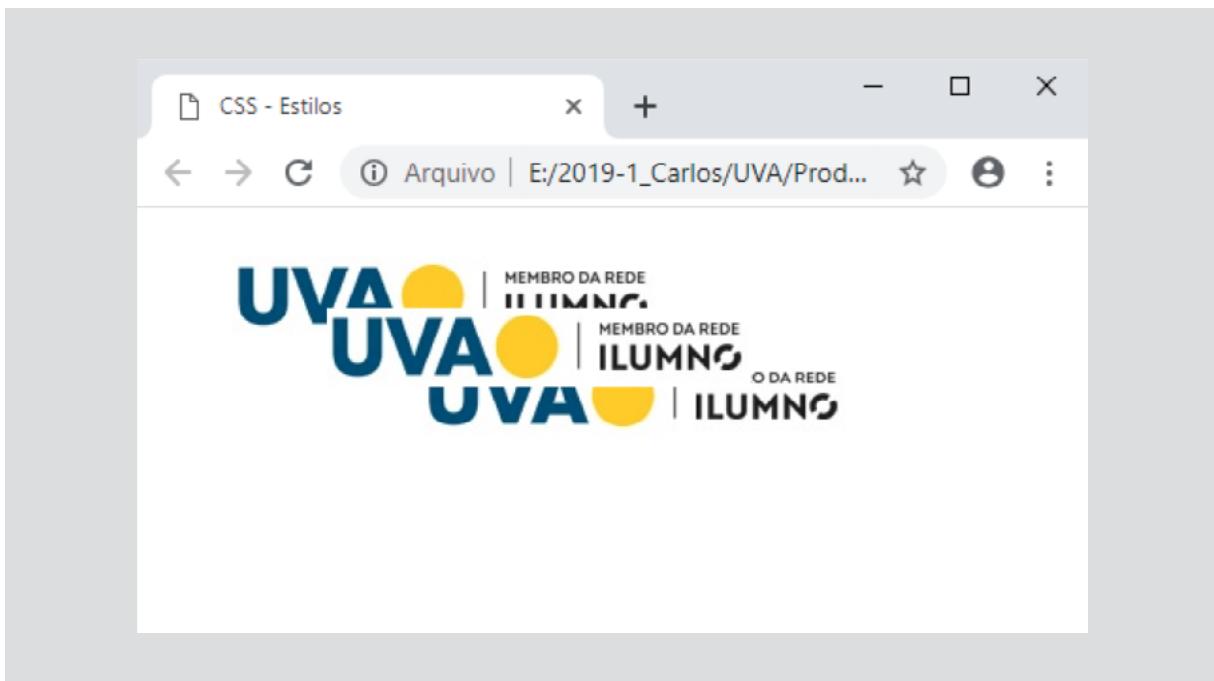
```
<html>
<head>
  <title>CSS - Estilos</title>
  <link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
</head>
<body>
  <p></p>
  <p></p>
</body>
</html>
```



Observe que as duas imagens estão sobrepostas, e a segunda imagem ficou sobre a primeira em função da ordem de inclusão do elemento na página.

Porém, essa ordem pode ser alterada pela propriedade *z-index*, passando a prioridade para uma determinada imagem, como pode ser visto no exemplo a seguir, css_12.html:

```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
</head>
<body>
<p></p>
    <p></p>
    <p></p>
</body>
</html>
```



Observe que o *z-index* determina a ordem de exibição na página; assim, a imagem do meio (segunda imagem), que tem número 3, será a terceira a ser exibida. Dessa forma, a primeira imagem exibida será a terceira (*z-order:1*), depois a primeira (*z-order:2*) e, por último, a segunda (*z-order:3*), que será a última a ser exibida e, por consequência, será a que irá se sobrepor às demais.



Importante

Se, ao visualizar a página no navegador, as imagens não ficarem sobrepostas, redimensione a janela do seu navegador, reduzindo os tamanhos vertical e horizontal dela.

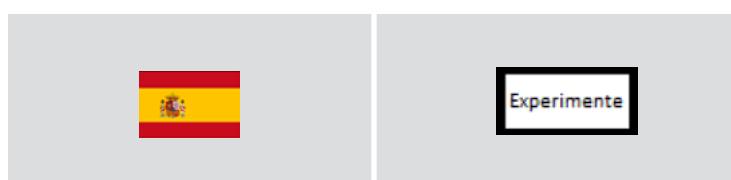
Cursor

Cursor (*cursor*) – Permite a mudança do cursor exibido pelo navegador durante a interação com a página. Um cursor é um ícone especial imagem, podendo ter as extensões: .cur, .gif ou .ani (cursor animado). Existe uma série de cursos padrão que podem ser utilizados. Na tabela a seguir você verá alguns exemplos, mas você poderá criar seus próprios cursos.

	default		progress
	crosshair		not-allowed
	hand		no-drop
	move		verticaltext
	text		all-scroll
	wait		col-resize
	help		row-resize
	n-resize		s-resize
	ne-resize		sw-resize
	e-resize		w-resize
	se-resize		nm-resize

Você também pode delimitar regiões em uma mesma página onde os cursores sejam diferentes. Outra forma de uso muito comum é na mudança do cursor de links, em que determinados links possuem cursores diferentes.

Copie aqui os arquivos de imagens necessários para o exemplo.



No exemplo a seguir, cada caixa com a imagem *experimente* irá possuir um cursor diferente. Nos primeiros exemplos, os cursos são padrão, bastando apenas identificá-los pelo nome, mas, nos dois últimos exemplos, os cursores são baseados em imagens específicas. No primeiro caso, a imagem está no mesmo diretório da página carregada, já no segundo exemplo, a imagem está armazenada em um outro servidor (AmazonWS).

O exemplo a seguir, `css_13.html`, demonstra o uso de diversos cursores padrão e de como usar cursores de arquivos de imagens locais ou na Web:

```

<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_5.css">
</head>
<body>
<p>
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    

```

```



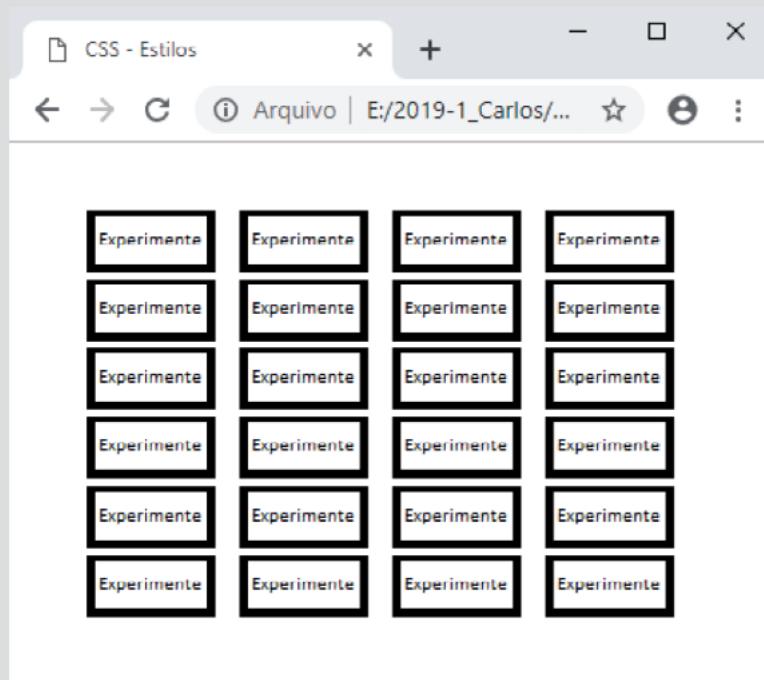






</p>
</body>
</html>

```



Se você passar o mouse sobre cada caixa de texto, verificará como o cursor muda quando está posicionado dentro de cada uma delas.



MIDIA TECA

Acesse a midiateca da Unidade 3 e veja o conteúdo complementar indicado pelo professor sobre o uso de imagens e suas sobreposições.



Saiba mais

Para saber mais sobre a padronização de sites, leia a Unidade 2 (p. 51-65) do seguinte livro:

SEGURADO, V. S. **Projeto de interface com o usuário**. São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

Padronização de textos (com definição de fontes, cores e fundos, padronização de imagens e tabelas) e criação de sites padronizados

Text (texto) – esta propriedade é responsável pela configuração do estilo do texto do componente.

Principais propriedades:

- *color* – cor do texto. Você pode definir a cor por meio de três formas:
 - pelo nome – color: blue;
 - pelo valor hexadecimal – color: #ff0000;
 - pelo valor RGB – color: rgb(255,0,255);
- *text-align* – alinhamento do texto em relação às margens.
 - *center*: posiciona o texto centralizado em relação às margens.
 - *left*: posiciona o texto junto à margem esquerda.
 - *right*: posiciona o texto junto a margem direita.
 - *justify*: posiciona o texto justificado entre as margens direita e esquerda.
- *text-decoration* – decoração do texto.
 - *none*: sem alteração.
 - *overline*: insere uma linha horizontal acima do texto.
 - *line-through*: insere uma linha horizontal no meio do texto.
 - *underline*: insere uma linha horizontal abaixo do texto.
- *text-transform* – realiza alterações transformando o texto entre caixa-alta (maiúsculas), caixa-baixa (minúsculas) e capitalização do texto, em que cada palavra começará por uma letra maiúscula (observe a diferença no código-fonte do exemplo a seguir do título 3, que no código HTML está começando por letra minúscula).
 - *capitalize*: as palavras começam sempre por letra maiúscula.
 - *uppercase*: todo texto é apresentado em letras maiúsculas.
 - *lowercase*: todo o texto é apresentado em letras minúsculas.

As propriedades a seguir podem ser definidas em: %; em; px; cm; mm; in; pt; pc.

Text-indent – identação do texto, de forma a linha inicial do texto comece a partir de um determinado ponto.

Exemplo – text-indent: 30px;

- *letter-spacing* – determina o espaçamento entre os caracteres de um texto.

Exemplo – letter-spacing: 5px;

- *line-height* – determina o espaçamento entre as linhas de um texto.

Exemplo – line-height: 1.8pc;

- *word-spacing* – determina o espaçamento entre as palavras de um texto.

Exemplo – word-spacing: 8px;

- *text-shadow* – determina um sombreamento sobre o texto, com a posição horizontal, vertical e a cor da sobra.

Exemplo – text-shadow: 4px 3px lime;

Os exemplos a seguir, css_14.html e arq_estilo_6.css, demonstram o uso da propriedade text para alguns componentes:

```
<html>
<head>
<title>CSS - Estilos</title>
<link rel="stylesheet" type="text/css" href="arq_estilo_6.css">
</head>
<body>
<h1> Título 1 (h1).</h1>
<h2> Título 2 (h2).</h2>
<h3> título 3 (h3).</h3>
<h4> Título 4 (h4).</h4>
<p> Exemplo parágrafo (body). Se não houver configuração específica ao componente,
as definições de padronização do body serão aplicadas aos componentes. Isto está
ocorrendo com os componentes (h4) e (p).</p>
```

```

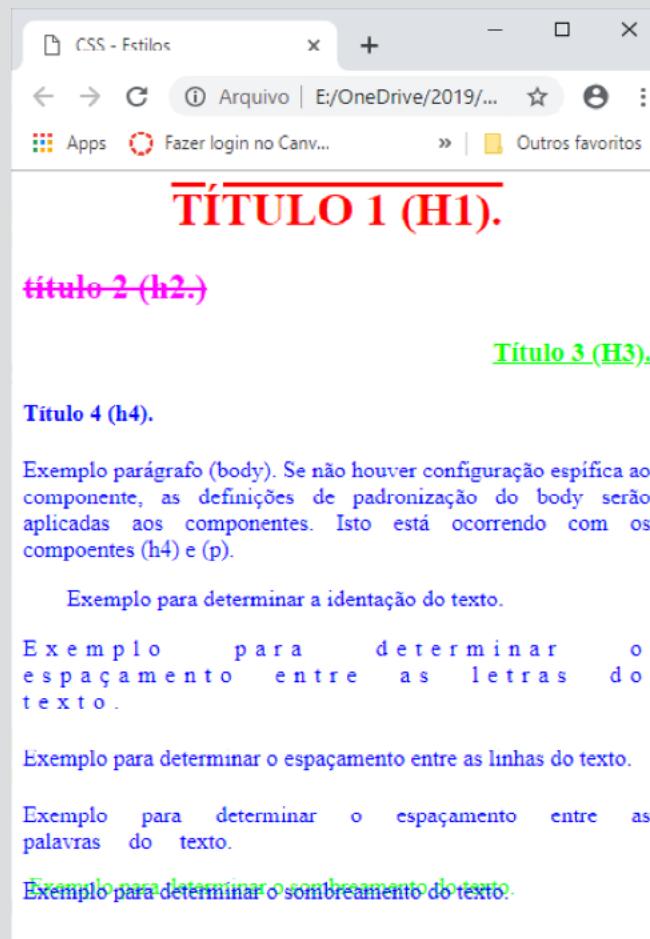
<p style="text-indent: 30px;"> Exemplo para determinar a identação do texto.</p>
<p style="letter-spacing: 6px;"> Exemplo para determinar o espaçamento entre as
letras do texto.</p>
<p style="line-height: 1.8pc;"> Exemplo para determinar o espaçamento entre as linhas
do texto.</p>
<p style="word-spacing: 15px;"> Exemplo para determinar o espaçamento entre as
palavras do texto.</p>
<p style="text-shadow: 4px -3px lime;"> Exemplo para determinar o sombreamento do
texto.</p>
</body>
</html>

```

```

body { color: blue; text-align: justify; text-decoration: none; }
h1 { color: #ff0000; text-align: center; text-decoration: overline; text-transform:
uppercase; }
h2 { color: rgb(255,0,255); text-align: left; text-decoration: line-through; text-transform:
lowercase; }
h3 { color: #00ff00; text-align: right; text-decoration: underline; text-transform: capitalize; }

```



Você deve analisar cada propriedade e realizar alterações nos valores para observar melhor o que ocorre em cada propriedade. Valores negativos no sombreamento permitem que a sombra fique antes ou depois do texto ou acima ou abaixo, conforme a necessidade.

Font (fonte) – esta propriedade é responsável pela definição da fonte de texto que será aplicada ao componente.

Principais propriedades:

- *font* – determina as declarações de fontes em uma única declaração. A fonte pode ser determinada por meio do grupo (família – *generic family*) ao qual pertence ou do nome específico (*font family*).

font-family – esta propriedade determina a família da fonte que será usada no componente.

Exemplo: `font-family: "Times New Roman";`

- *font-size* – esta propriedade determina o tamanho da fonte, podendo ser definida em: %; em; px; cm; mm; in; pt; pc.

Exemplo: `font-size: 32px;`

- *font-style* – esta propriedade determina o estilo da fonte.
 - *normal* – o texto é apresentado de forma normal.
 - *italic* – o texto é apresentado em itálico.
 - *oblique* – o texto é apresentado em forma oblíqua (semelhante ao itálico).
-
- *font-variant* – esta propriedade determina as variações de exibição da fonte.

Exemplos: `font-variant: normal;` e `font-variant: small-caps;`

- *font-weight* – esta propriedade determina a largura da fonte entre normal e negrito.

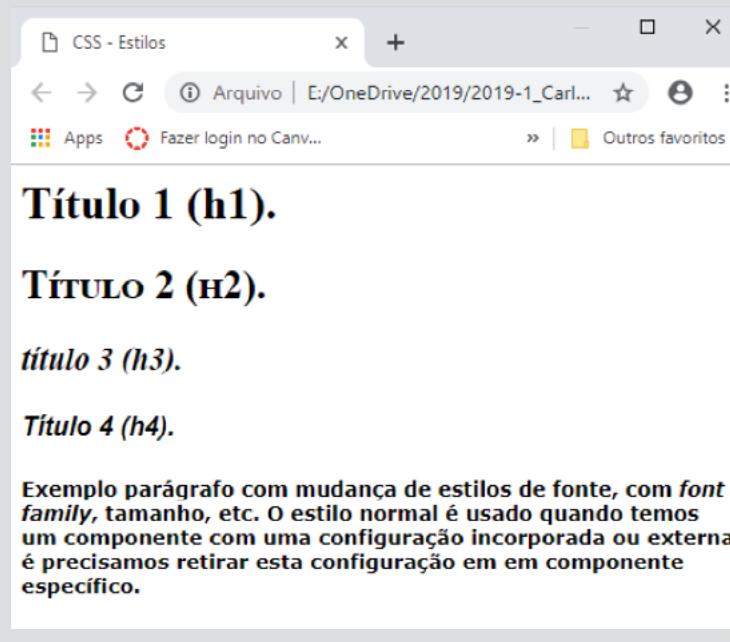
Exemplos: `font-weight: normal;` e `font-weight: bold;`

O exemplo a seguir, `css_15.html`, demonstra o uso da propriedade *font* para alguns componentes:

```

<html>
<head>
<title>CSS - Estilos</title>
</head>
<body>
<h1 style="font-family: Times; font-size: 32px;"> Título 1 (h1).</h1>
<h2 style="font-family: Times New Roman; font-size: 28px; font-variant: small-caps;">
Título 2 (h2).</h2>
<h3 style="font-family: Serif; font-size: 22px; font-style: italic;"> título 3 (h3).</h3>
<h4 style="font-family: Arial; font-size: 18px; font-style: oblique;"> Título 4 (h4).</h4>
<p style="font-family: Verdana ; font-size: 14px; font-weight: bold;"> Exemplo parágrafo
com mudança de estilos de fonte, com <i>font family</i>, tamanho, etc. O estilo normal
é usado quando temos um componente com uma configuração incorporada ou externa
é precisamos retirar esta configuração em em componente específico.</p>
</body>
</html>

```



Colors (cores) – esta propriedade é responsável pela definição das cores que serão aplicadas ao componente.

As cores podem ser determinadas de diferentes formas:

- Nome da cor – color: lime; // Já visto anteriormente.
- RGB: color – rgb(255, 128, 0); // Já visto anteriormente.

- Hexadecimal – color: #ff8000; // Já visto anteriormente.
- HSL – color: hsl(12, 90%, 70%); // Inclui a matiz (0 a 360), saturação (0% a 100%) e luminosidade (0% a 100%).
- RGBA – color: rgba(255, 128, 0, 0.8); // Inclui, além da formação das cores, a opacidade da cor (0.0 a 1.0), sendo 0.0 totalmente transparente e 1.0 não transparente.
- HSLA – color: hsl(12, 90%, 70%, 0.7); // Inclui, além da matiz, saturação e luminosidade, a opacidade da cor (0.0 a 1.0).

Principais propriedades:

- *background-color*– determina a cor de fundo de um componente.

Exemplo: background-color: DodgerBlue;

- *color*– determina a cor do texto de um componente.

Exemplo: color: Pink;

- *border*– determina o tamanho, o estilo e a cor da borda de um componente.

Exemplo: border:2px solid Lime;

O exemplo a seguir, css_16.html, demonstra o uso da propriedade colors para alguns componentes:

```
<html>
<head>
<title>CSS - Estilos</title>
</head>
<body>
  <h1 style="background-color: rgb(255, 128, 0); color: blue; border:5px solid Lime;">
    Título 1 (h1).</h1>

  <h2 style="background-color: #ff0080; color: lime; border:5px dotted blue;">
    Título 2 (h2).</h2>

  <h3 style="background-color: hsl(120, 90%, 70%); color: red; border:5px double
    DodgerBlue;">
    Título 3 (h3).</h3>

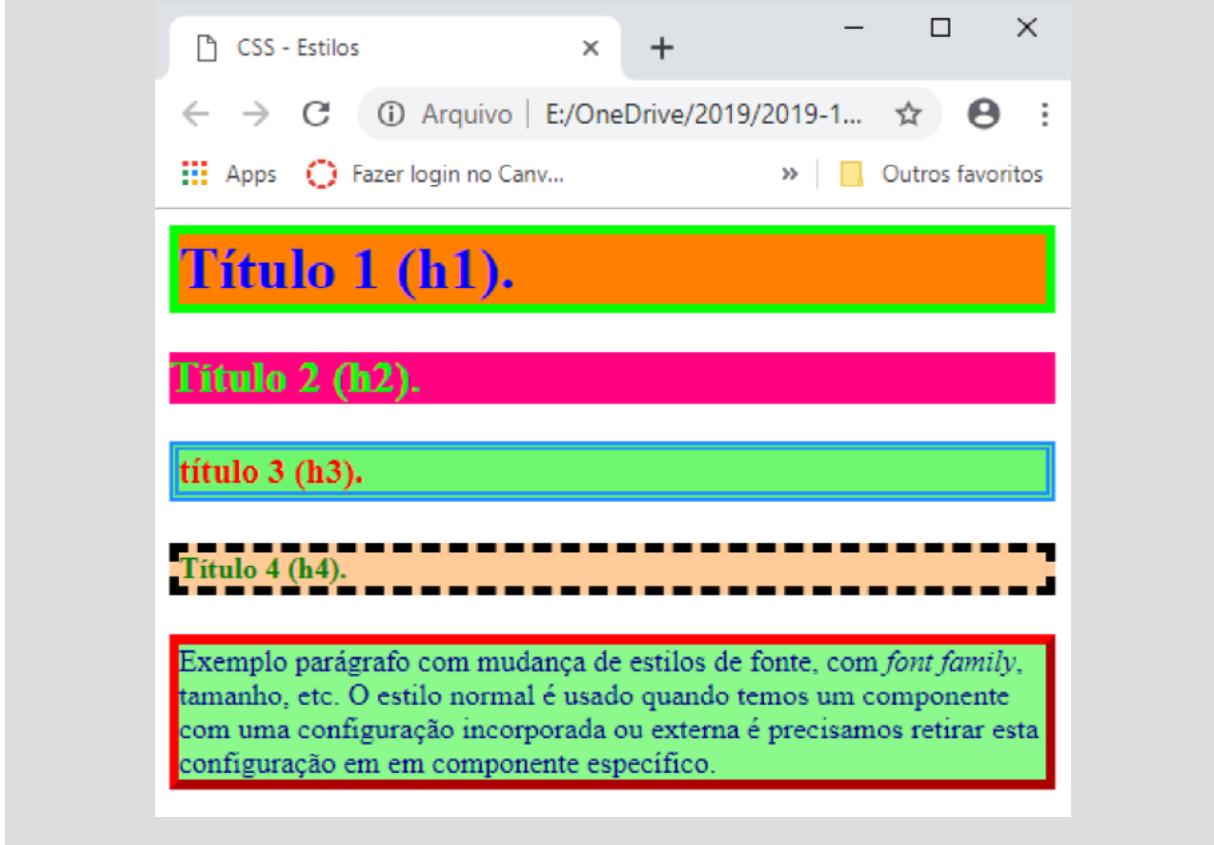
```

```

<h4 style="background-color: rgba(255, 128, 0, 0.4); color: green; border:5px dashed black;"> Título 4 (h4)</h4>

<p style="background-color: hsla(120, 90%, 70%, 0.8); color: DarkBlue; border:5px outset red;"> Exemplo parágrafo com mudança de estilos de fonte, com <i>font family</i>, tamanho, etc. O estilo normal é usado quando temos um componente com uma configuração incorporada ou externa é precisamos retirar esta configuração em um componente específico.</p>
</body>
</html>

```



Background (fundo) – esta propriedade é responsável pela definição dos padrões de fundo da página.

Principais propriedades:

- *background* – determina as propriedades de fundo em uma única declaração. As cores podem ser especificadas por meio do nome, hexadecimal ou RGB.

Exemplo: `background: #ffffff url("imagem.png") no-repeat right top;`

- *background-attachment* – determina se a imagem de fundo será fixa ou irá rolar com a página. Podem ser usados .GIF, .JPG, .PNG, entre outros formatos.

Exemplo: background-attachment: fixed;

- *background-color* – determina a cor de fundo de um componente.

Exemplo: background-color: darkblue;

- *background-image* – determina a imagem de fundo de um componente. Podem ser usados .GIF, .JPG, .PNG, entre outros formatos.

Exemplo: background-image: url("imagem.gif");

- *background-position* – determina a posição inicial de imagem de fundo.

Exemplo: background-position: right top;

- *background-repeat* – determina como uma imagem de fundo será repetida.

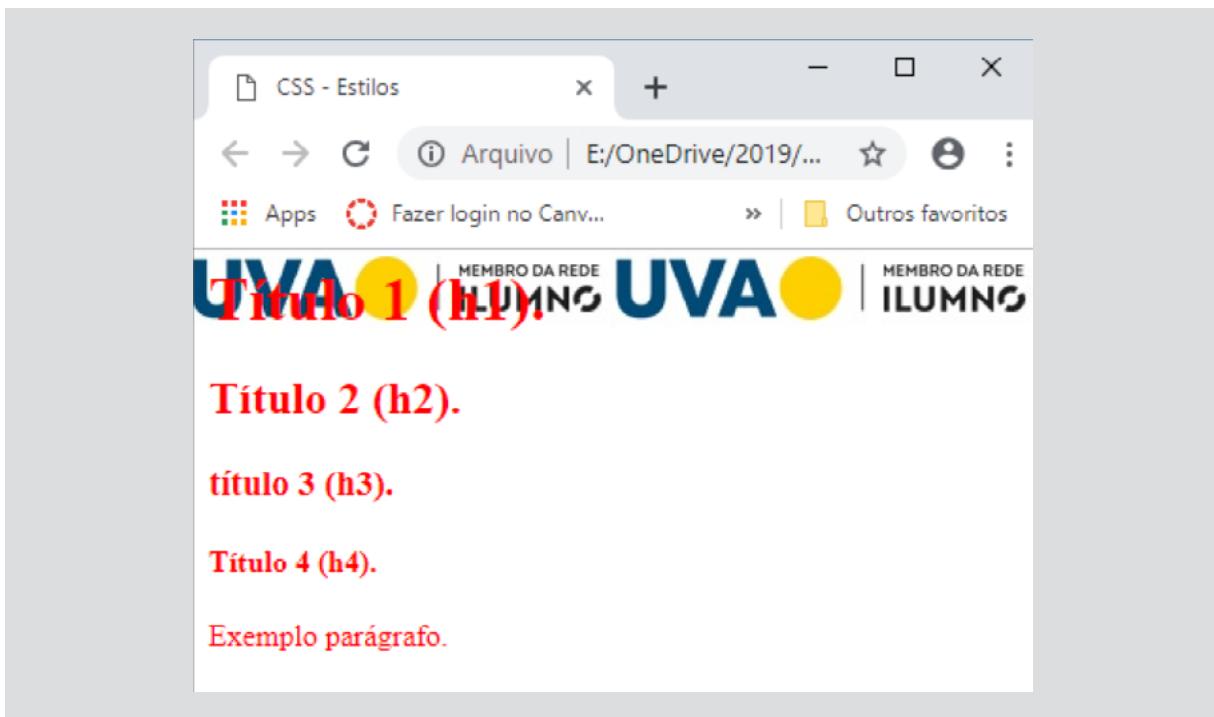
Exemplo: background-repeat: repeat-x; // repete a imagem horizontalmente.

background-repeat: repeat-y; // repete a imagem verticalmente.

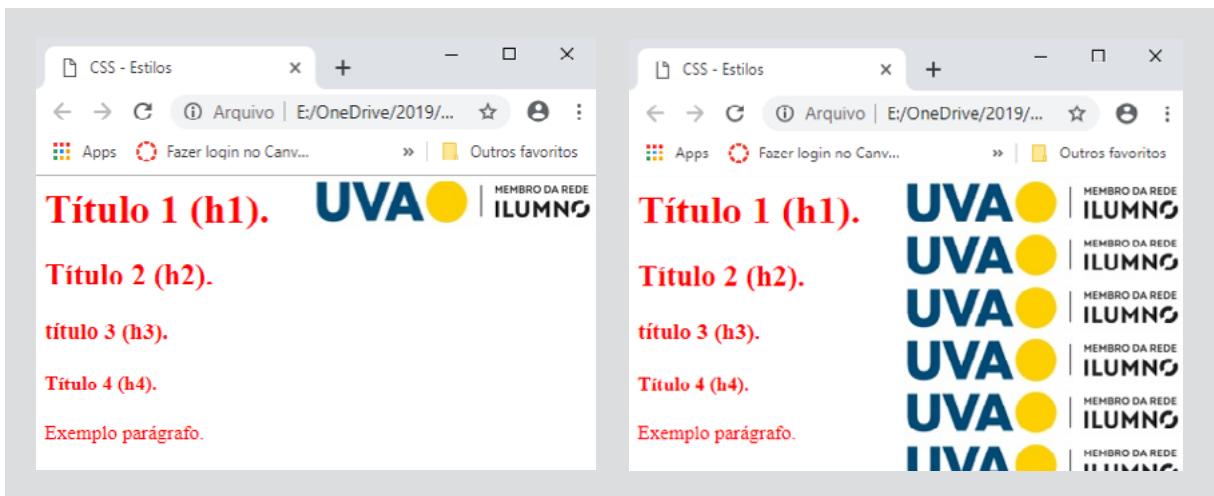
background-repeat: no-repeat; // não repete a imagem.

O exemplo a seguir, css_17.html, demonstra o uso da propriedade *background* para alguns componentes:

```
<html>
<head>
<title>CSS - Estilos</title>
</head>
<body style="color: red; background-repeat: no-repeat; background-position: right top;
background-image: url(ilumno.jpg)">
<h1> Título 1 (h1).</h1>
<h2> Título 2 (h2).</h2>
<h3> título 3 (h3).</h3>
<h4> Título 4 (h4).</h4>
<p> Exemplo parágrafo.</p>
</body>
</html>
```



Faça alterações no código e analise os resultados ao usar repeat-x, repeat-y, além de outras alterações:



Se você desejar que a imagem fique fixa em uma página, mesmo com a rolagem da janela, use o exemplo a seguir, css_18.html:

```

<html>
<head>
<title>CSS - Estilos</title>
</head>
<body style="color: red; background-repeat: no-repeat; background-attachment: fixed;
background-position: right top; background-image: url(ilumno.jpg)">
<h1> Título 1 (h1).</h1>
<h2> Título 2 (h2).</h2>
<h3> título 3 (h3).</h3>
<h4> Título 4 (h4).</h4>
<p> Exemplo parágrafo.</p>
</body>
</html>

```

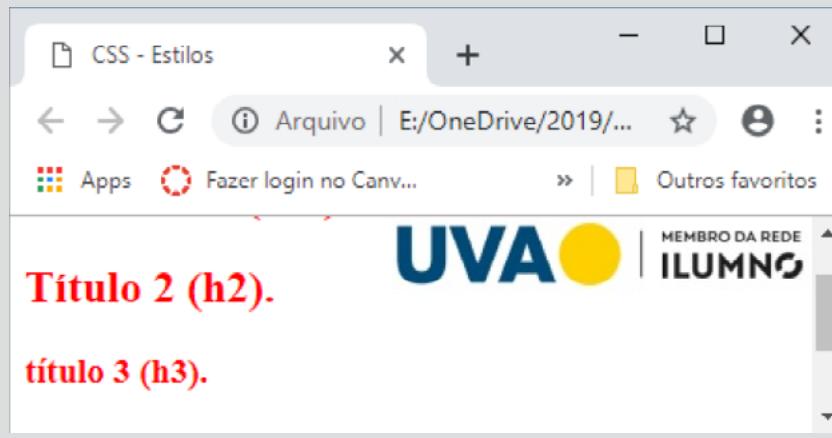


Table (tabela) – esta propriedade é responsável pela definição dos padrões que serão aplicados às tabelas.

Principais propriedades:

- *border* - determina todas as propriedades de borda em uma única declaração.

Exemplo: *border: 1px solid blue;*

- *border-collapse* - determina as bordas da tabela que devem ou não ser recolhidas.

Exemplo: *border-collapse: collapse;*
border-collapse: separate;

- *border-spacing* - determina a distância entre as bordas de células que sejam adjacentes.

Exemplo: border-spacing: 15px 50px;

- *caption-side* - determina o posicionamento da legenda da tabela.

Exemplo: caption-side: bottom;
caption-side: top;

- *empty-cells* - determina se serão ou não exibidas bordas e plano de fundo para células vazias de uma tabela.

Exemplo: empty-cells: hide;

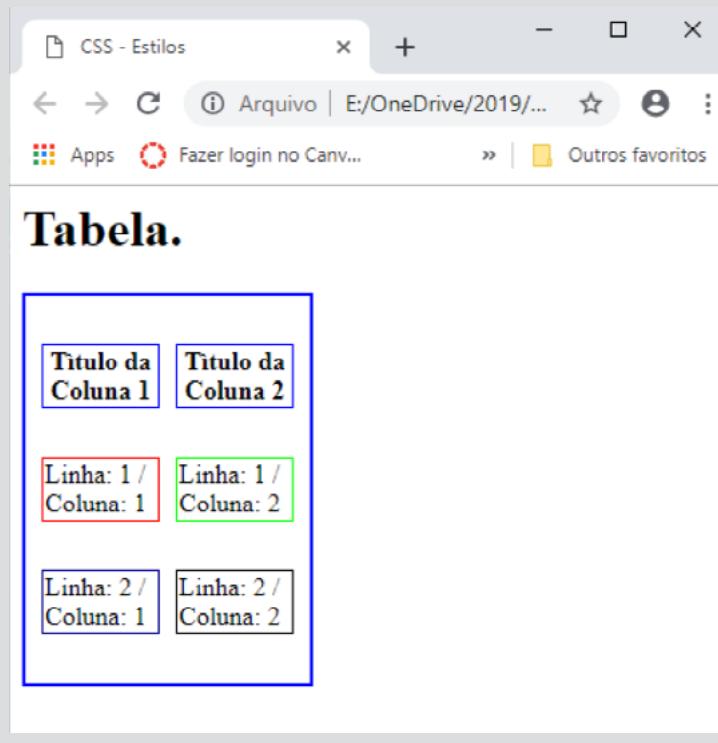
- *table-layout* - determina a forma de montagem a ser usada em uma tabela. *Auto* permite que o navegador determine a altura e a largura, e *fixed* fará com que as células tenham valores fixos.

Exemplo: table-layout: auto;
table-layout: fixed;

O exemplo a seguir, css_19.html, demonstra o uso da propriedade table para uma tabela:

```
<html>
<head>
<title>CSS - Estilos</title>
</head>
<body>
<h1> Tabela.</h1>
<table style="border: 2px solid blue; border-spacing: 10px 30px; table-layout: auto; width: 180px; ">
    <tr>
        <th style="border: 1px solid blue;">Título da Coluna 1</th>
        <th style="border: 1px solid blue;">Título da Coluna 2</th>
    </tr>
    <tr>
        <td style="border: 1px solid red;">Linha: 1 / Coluna: 1</td>
        <td style="border: 1px solid lime;">Linha: 1 / Coluna: 2</td>
    </tr>
</table>
```

```
</tr>
<tr>
<td style="border: 1px solid darkblue;">Linha: 2 / Coluna: 1</td>
<td style="border: 1px solid black;">Linha: 2 / Coluna: 2</td>
</tr>
</table>
</body>
</html>
```



MEDIATECA

Acesse a mediateca da Unidade 3 e veja o conteúdo complementar indicado pelo professor sobre como colocar um vídeo como fundo em uma página HTML com CSS.



Saiba mais

Para saber mais sobre as propriedades CSS para a padronização de sites, leia o Apêndice D (p. 963-998) do seguinte livro:

LEMAY, L. **Aprenda a criar páginas web com HTML e XHTML em 21 dias.** São Paulo: Pearson Education do Brasil, 2002. Biblioteca Virtual.



Dica

Para conhecer mais propriedades de configuração de estilos CSS, vale a pena conhecer o site de tutorial do w3schools, cujo entendimento técnico, mesmo estando em inglês, é simples, além de possuir inúmeros exemplos que você pode aproveitar em seus sites.

A padronização de estilos possui uma grande quantidade de propriedades, o nosso objetivo foi conhecer as principais propriedades e principalmente, como aplicar os estilos em nossos sites.



NA PRÁTICA

A grande maioria dos sites possui configurações de padronização, mas não temos acesso aos padrões externos diretamente, uma vez que eles não fazem parte da página. Entretanto, você pode analisar e conhecer os códigos de padronização de estilos incorporados e locais aplicados a qualquer outro site. Basta acessar qualquer site (como sugestão, use o site do Museu de Arte do Rio de Janeiro©) e, após o navegador carregar a página, utilize o recurso do navegador de exibição da codificação da página (F12).

Resumo da Unidade 3

Nesta unidade, você entendeu a importância do uso de estilos na padronização de sites com grande número de páginas. Compreendeu a ordem de prioridade na aplicação de estilos a partir da aplicação de estilos de forma genérica por meio de um arquivo externo, até a aplicação específica em uma linha específica de uma página. Observou que usar a padronização de estilos permite realizar a manutenção e mudanças em grandes sites com pouco esforço de programação.



CONCEITO

Nesta unidade, destacaram-se os seguintes conceitos:

- Folhas de estilos.
- Tags CSS.
- Estilos em cascata e ordem de prioridade.

Referências

LEMAY, L. **Aprenda a criar páginas web com HTML e XHTML em 21 dias.** São Paulo: Pearson Education do Brasil, 2002. Biblioteca Virtual.

MILETTO, E.; BERTAGNOLLI, S. **Desenvolvimento de software II:** introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. Minha Biblioteca.

SEGURADO, V. S. **Projeto de interface com o usuário.** São Paulo: Pearson Education do Brasil, 2015. Biblioteca Virtual.

UNIDADE 4

Introdução ao desenvolvimento de
aplicações Web com JavaScript

INTRODUÇÃO

O tratamento por meio de uma linguagem na parte de servidor da aplicação é fundamental para o gerenciamento do sistema, e as mais comuns são Java, C#, Python e PHP. Por outro lado, também é necessário que tenhamos uma linguagem de programação na parte cliente para verificar se todos os dados foram preenchidos, analisar e criticar os valores preenchidos antes do envio e realizar ações que possam facilitar o uso do sistema pelo usuário ou torna o processo mais eficiente em tempo de resposta. Nesse caso, a linguagem mais popular é o JavaScript. Nesta unidade, aprenderemos como utilizar o JavaScript em nossas aplicações Web Cliente. Para deixar claro, JavaScript, apesar dos nomes parecidos, não tem nenhuma relação com a linguagem Java, sendo assim duas linguagens diferentes e com objetivos diferentes. JavaScript foi criada pela equipe de desenvolvimento da NetScape e, inicialmente, seu nome era LiveScript. Trata-se de uma linguagem interpretada e está disponível em todos os navegadores de internet.



OBJETIVO

Nesta unidade você será capaz de:

- Construir aplicações *front-end* para a web com o uso de JavaScript.

Estrutura de aplicações com programação com JavaScript e tipos de dados e expressões

JavaScript trabalha preferencialmente com funções, em que cada função deve realizar uma ação específica que será relacionada a um evento de um componente da página. Uma função pode chamar outras funções. As funções podem retornar valores para a página ou componente ou apenas exibir mensagens ou conteúdo na página. Cada função é delimitada como um bloco por meio do uso de { para iniciar um bloco ou função até o }, que determina o encerramento do bloco ou função.

Quando a programação em JavaScript é incorporada à página HTML, seus códigos podem ser acessados de forma simples, utilizando-se a função F12 do navegador. A linguagem pode ser usada para configurar dinamicamente uma página Web, mas o seu principal uso é em conjunto com formulários HTML e na configuração de interfaces gráficas com o usuário.

A linguagem JavaScript é fundamental para o desenvolvimento de aplicações clientes para a Web.

A codificação em JavaScript deve ser colocada em uma tag <script> ... </script>, que é responsável por identificar os códigos que deverão ser interpretados pelo navegador. O código deve ficar em um trecho de comentários para evitar que seu conteúdo seja apresentado junto ao conteúdo acidentalmente. A tag <script> pode ficar tanto no cabeçalho <head> da página como no corpo <body>.

JavaScript pode ser incluído diretamente na página (incorporado) ou por meio de um arquivo (externo). O seu uso é normalmente:

```
<script Language="JavaScript">
<!--
// 
-->
</script>
```

Incorporado <head>

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript">
      //inclusão aqui do seu código
    </script>
  </head>
  <body>
    </body>
</html>
```

Incorporado <body>

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">
      //inclusão aqui do seu código
    </script>
  </body>
</html>
```

Externo: Arquivo

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript"
      src="codigos.js"></script>
  </head>
  <body>
    </body>
</html>
```

Externo: URL

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript"
      src="https://site/js/codigos.js">
    </script>
  </head>
  <body>
    </body>
</html>
```

A estrutura da linguagem é semelhante à linguagem C, e ambas possuem várias semelhanças.

Você encontrará normalmente duas formas de declaração da linguagem:

- <script type="text/javascript">
- <script Language="JavaScript">

Operadores relacionais e lógicos

Os operadores lógicos são usados para designar um resultado baseado em uma relação lógica entre operados. Esses operadores são utilizados em estruturas condicionais e podem ser usados em conjunto em expressões lógicas.

Operador	Operação lógica	Exemplos
<code>==</code>	Comparação de igualdade	<code>(a == b)</code>
<code>!=</code>	Comparação de diferença	<code>(a != b)</code>
<code>></code>	Comparação de maior que	<code>(a > b)</code>
<code>>=</code>	Comparação de maior que ou igual a	<code>(a >= b)</code>
<code><</code>	Comparação de menor que	<code>(a < b)</code>
<code><=</code>	Comparação de menor que ou igual a	<code>(a <= b)</code>

Observação: o operador `=`, quando usado sozinho, não faz a comparação, é usado apenas para atribuição, e não para comparação (relação).

Em expressões lógicas devemos utilizar os operadores das tabelas-verdade de NÃO, E ou OU.

Operador	Operação lógica	Exemplos
<code>!</code>	Negação lógica (inverte um valor lógico)	<code>(!a>=b)</code>
<code>&&</code>	E lógico em uma expressão lógica	<code>(a >= b && a>=c)</code>
<code> </code>	OU lógico em uma expressão lógica	<code>(a >= b a>=c)</code>

Em expressões lógicas podemos usar mais de um operador ao mesmo tempo.

Exemplo: `(a >= b || a>=c && a>=d)`

O operador de negação (`!`) tem maior precedência em uma expressão lógica e o E (`&&`) tem precedência sobre o operador OU (`||`). Dessa forma, no exemplo acima será verificada primeiramente a expressão E (`a>=c && a>=d`), para somente depois ser avaliado o operador OU. Parênteses podem ser usados para alterar a precedência.

Operadores aritméticos

Os operadores matemáticos são responsáveis por realizar cálculos a partir de expressões aritméticas. A linguagem usa o símbolo de = para a atribuição em expressões aritméticas.

Operador	Operação aritmética	Exemplos
*	Multiplicação	$a = b * c$
/	Divisão	$a = b / c$
%	Resto da divisão inteira	$a = b \% c$
+	Soma	$a = b + c$
-	Subtração	$a = b - c$

Observação: para concatenar duas ou mais *strings* (textos), basta utilizar o símbolo de adição (+ operador de soma), conforme o exemplo:

A = "João"

B = "da Silva"

C = A + B

C receberá então "João da Silva".

Podem ainda ser utilizadas formas mais curtas em alguns casos:

Operador	Expressão aritmética	Exemplos
*=	$a = a * b$	$a *= b$
/=	$a = a / b$	$a /= b$
%=	$a = a \% b$	$a \%= b$
+=	$a = a + b$	$a += b$
-=	$a = a - b$	$a -= b$

Ainda podemos usar o incremento (++) e o decremento (--).

Principais funções embutidas da linguagem:

Função	Descrição	Exemplo
eval(v)	Converte uma string (texto) em um valor.	valor=eval(var); valor=eval("123.45");
Date()	Retorna a data e hora.	alert(Date());
parseInt(v)	Converte o valor do parâmetro para um valor inteiro.	valor=parseInt(var);
parseFloat(v)	Converte o valor do parâmetro para um valor real (ponto flutuante).	valor=parseFloat(var);
isNaN(v)	Retorna true, caso o valor contido na variável não seja numérico.	if(isNaN(var)) // true se for texto // false se for um valor

A função isNaN é muito utilizada para identificar se o que foi preenchido em uma caixa de texto é um valor que pode ser convertido para um tipo numérico ou não. Utilizaremos essa função para verificar se o preenchimento de um determinado campo está de acordo com o solicitado.

Principais funções matemáticas:

Função	Descrição	Exemplo
Math.abs(v)	Tem a função de retornar o valor absoluto de um número. Por exemplo, caso seja definido o valor -123, ele será convertido para 123.	valor=math.abs(-123); alert(valor);

Math.acos(v)	Retornará o arco cosseno (em radianos) do valor.	valor=math.acos(0.12); alert(valor);
Math.asin(v)	Retornará o arco seno (em radianos) do valor.	valor=math.asin(0.12); document.write(valor);
Math.ceil(v)	Retorna um inteiro maior ou igual a um número. O resultado desse método é equivalente ao arredondamento de um número.	valor=math.ceil(12.6); valor2=math.ceil(-12.6); alert(valor); alert(valor2);
Math.cos(v)	Retornará o cosseno (em radianos) do valor.	valor=math.cos(0.12); alert(valor);
Math.E	Representa o valor constante de E (base logarítmico).	valor=math.exp(0.0009); alert(valor);
Math.exp(v)	Retornará o logaritmo do número na base E.	valor=math.exp(0.0009); alert(valor);
Math.floor(v)	Retorna o maior inteiro menor ou igual a um número.	valor=math.floor(101.25); valor2=math.floor(-101.25); alert(valor); alert(valor2);
Math.log(v)	Retorna o logaritmo natural de um número (base E).	valor=math.log(1.1); alert(valor);
Math.max(n1, n2)	Retorna o maior valor entre dois números.	valor=math.max(5,10); alert(valor);
Math.min(n1, n2)	Retorna o menor valor entre dois números.	valor=math.min(5,10); alert(valor);
isNaN(v)	Retorna true, caso o valor contido na variável não seja numérico.	if(isNaN(var)) // true se for texto // false se for um valor
Math.pow((x, y) base, expoente	Retorna a base elevada à potência do expoente.	valor=math.pow(12,2); alert(valor);

Math.random()	Retorna um número aleatório entre zero e um com até 15 dígitos. Esse número aleatório é definido pelo relógio do computador.	alert(math.random());
Math.round(v)	Com o método round, é possível arredondar um valor.	valor=math.round(125.6); alert(valor);
Math.sin(v)	Esse método retorna o seno de um número.	valor=math.sin(1.6); alert(valor);
Math.sqrt(v)	Retorna a raiz quadrada de um número.	valor=math.sqrt(49); alert(valor);
Math.pi	Representa o valor constante de Pi.	valor=Math.PI * 3;
Math.tan(v)	Retorna a tangente de um número.	valor=math.tan(1.5); alert(valor);

Principais funções para manipulação de string (texto):

Função	Descrição	Exemplo
String.length	Retorna a quantidade de bytes de uma string (tamanho da string).	var txt = "Alô Mundo!"; var r = txt.length; // r receberá 10
String.charAt(n)	Retorna o caractere que se encontra na posição indicada por n da string.	var txt = "Alô, Mundo!"; var r = txt.charAt(5); // r receberá 'M'
String.indexOf("string")	Retorna o valor da posição na string onde está a primeira ocorrência da string.	var txt = "Alô Mundo, Alô!"; var r = txt.indexOf("Alô"); // r receberá 0
String.lastIndexOf("string")	Retorna o valor da posição na string onde está a última ocorrência da string.	var txt = "Alô Mundo, Alô!"; var r = txt.lastIndexOf("A- lô"); // r receberá 11

String.substr(i1, n)	Retorna uma string contendo o texto da posição i1 e mais os caracteres consecutivos definidos por n.	var txt = "Alô Mundo, Alô!"; var r = txt.substr(4,5); // r receberá "Mundo"
String.substring(i1, i2)	Retorna uma string contendo apenas o texto entre as posições i1 e i2 da string.	var txt = "Alô Mundo, Alô!"; var r = txt.substring(4,9); // r receberá "Mundo"

Observações: o índice de uma *string* sempre começa da posição 0. As funções *substr* e *substring* são diferentes – a primeira retorna os n caracteres a partir da posição i1, e a segunda retorna os caracteres compreendidos entre as posições i1 e i2 da *string*.

Principais funções para manipulação de datas:

Podemos criar uma variável com a data de hoje: DataHoje = new Date();

Função	Descrição	Exemplo
getDate()	Retorna o dia do mês (1 a 31).	dia = DataHoje.getDate();
getDay()	Retorna o dia da semana (0 a 6).	semana = DataHoje.getDay();
getMonth()	Retorna o mês (0 a 11).	mes = DataHoje.getMonth();
getYear()	Retorna o ano.	ano = DataHoje.getYear();
getHours()	Retorna a hora (0 a 23).	hora = DataHoje.getHours();
getMinutes()	Retorna os minutos (0 a 59).	minuto = DataHoje.getMinutes();
getSeconds()	Retorna os segundos (0 a 59).	segundo = DataHoje.getSeconds();

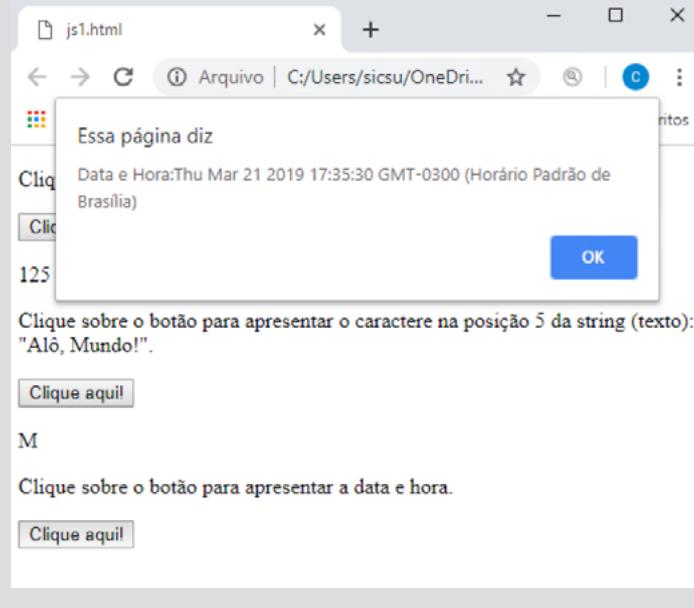
As variáveis em JavaScript são fracamente tipadas, ou seja, não é necessário definir o tipo das variáveis. Sendo assim, podemos usar:

```
var nome;  
nome = "João da Silva";  
var preco = 30;  
preco = (30 + 5) * 1.5;
```

Exemplo: js1.html:

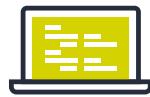
```
<!DOCTYPE html>  
<html>  
<body>  
<p>Clique sobre o botão para apresentar o valor de x=5 elevado a y=3.</p>  
<button onclick="funcao1()">Clique aqui!</button>  
<p id="exemplo1"></p>  
<p>Clique sobre o botão para apresentar o caractere na posição 5 da string (texto): "Alô,  
Mundo!".</p>  
<button onclick="funcao2()">Clique aqui!</button>  
<p id="exemplo2"></p>  
<p>Clique sobre o botão para apresentar a data e hora.</p>  
<button onclick="funcao3()">Clique aqui!</button>  
<p id="exemplo3"></p>  
<script type="text/javascript">  
!--  
function funcao1() {  
    var x=5;  
    var y=3;  
    var r = Math.pow(x, y);  
    document.getElementById("exemplo1").innerHTML = r;  
}  
function funcao2() {  
    var str = "Alô, Mundo!";  
    var r = str.charAt(5);  
    document.getElementById("exemplo2").innerHTML = r;  
}  
function funcao3() {  
    var dataHoje = new Date();  
    alert("Data e Hora:" + dataHoje);  
}
```

```
-->  
</script>  
</body>  
</html>
```



Analizando o código:

- Na criação de cada botão, foi definido o evento onClick, que ainda será mais bem detalhado, mas que identifica a função JavaScript, que será executada ao se clicar sobre o respectivo botão.
- As funções foram criadas dentro da tag <script>, e o script foi detalhado como JavaScript na propriedade type.
- Foram criados três parágrafos para apresentar os resultados, um para cada exemplo e com a sua respectiva identificação.
- Ao se incluir um conteúdo na página com o resultado de cada função, ele é incluído ao respectivo parágrafo, evitando que os resultados se embaralhem. A instrução document.getElementById("exemplo1").innerHTML = r; acrescenta ao conteúdo da página no elemento "exemplo1", que é um parágrafo (identificado), o valor (conteúdo) da variável, no caso, a variável r.
- Uma mensagem pode ser exibida ao usuário por meio da função alert(), que apresenta uma caixa de mensagem sobre a tela do navegador. No nosso exemplo isso foi feito para apresentar a data e hora do sistema por meio da função 3 (funcao3).



MEDIATECA

Acesse a midiateca da Unidade 4 e veja o conteúdo complementar indicado pelo professor sobre a história da linguagem JavaScript e sua compatibilidade com os diversos navegadores.



Saiba mais

Para saber mais sobre sintaxe, operadores e funções em JavaScript, leia sobre JavaScript na unidade 2 (p. 67-93) do seguinte livro:

SEGURADO, V. S. **Projeto de interface com o usuário**. São Paulo: Pearson, 2015. Biblioteca Virtual.

Estruturas de controle de fluxo

As estruturas de controle de fluxo são responsáveis por definir ou alterar a sequência lógica de execução em um programa de computador. São usadas para definir a ordem de execução, determinar desvios e repetições na execução do programa.

Sequência – É a estrutura que determina a ordem sequencial de execução de um conjunto de instruções, que são organizadas de forma lógica com o propósito de solucionar um problema.

Sintaxe

```
instrução_1;  
instrução_2;  
instrução_3;
```

Exemplo

```
function exemplo1(){  
    var raio = 2.5;  
    var areaCirc = Math.PI * Math.pow(raio,2);  
    alert("Área da Circunferência= ") + areaCirc;  
}
```

Decisões – São estruturas que determinam uma mudança no fluxo sequencial, de forma que o programa, após a verificação de uma condição, desvie o fluxo a fim de atender a uma situação específica do programa.

- SE (*if*) – Estrutura condicional de mudança de fluxo que desvia o fluxo de controle para uma ou mais instruções em função de uma relação lógica que seja verdadeira. É usada em situações em que não seja necessária execução de uma ou mais instruções, caso o resultado da relação seja falso.

Sintaxe

```
if (condição) {  
    instruções;  
}
```

Exemplo

```
function exemplo2(){  
    if(idade>=18){  
        alert("Você já tem a maior idade.");  
    }  
}
```

- SE/SENÃO (*if/else*) – Estrutura condicional de mudança de fluxo que desvia o fluxo para um entre dois conjuntos com uma ou mais instruções em função de uma relação lógica que seja verdadeira ou falsa. É usada em situações em que seja necessária execução de uma ou mais instruções, caso o resultado da relação seja falso.

Sintaxe

```
if (condição) {  
    instruções;  
} else {  
    instruções;  
}
```

Exemplo

```
function exemplo3(){  
    if(idade>=18){  
        alert("Você já tem a maior idade.");  
    } else {  
        alert("Você ainda não tem a maior idade.");  
    }  
}
```

- SE/SENÃO/SE (*if/else if*) – Estrutura condicional de mudança de fluxo que desvia o fluxo para um entre vários conjuntos com uma ou mais instruções em função de uma relação lógica que seja verdadeira ou falsa em vários casos distintos de condições. É usada em situações em que sejam necessárias várias condicionais SEs aninhadas (uma dentro da outra) ou em substituição à estrutura CASO, em situações em que sejam necessárias verificações de valores que não sejam de igualdade.

Sintaxe

```
if (condição1) {
    instruções;
} else if (condição2){
    instruções;
} else if (condição3){
    instruções;
} else {
    Instruções;
}
```

Exemplo

```
var U = 110;
var R = 10000;
var I = U/R;
//alert(I);
if(I<=Math.pow(10,-9)){
    alert("Corrente menor que 10 elevado a -9.");
} else if (I<=Math.pow(10,-6)){
    alert("Corrente menor que 10 elevado a -6.");
} else if (I<=Math.pow(10,-3)){
    alert("Corrente menor que 10 elevado a -3.");
} else if (I<=Math.pow(10,0)){
    alert("Corrente menor que 10 elevado a 0.");
} else if (I<=Math.pow(10,3)){
    alert("Corrente menor que 10 elevado a 3.");
} else if (I<=Math.pow(10,6)){
    alert("Corrente menor que 10 elevado a 6.");
} else if (I<=Math.pow(10,9)){
```

```
    alert("Corrente menor que 10 elevado a 9.");
} else if (I<=Math.pow(10,12)){
    alert("Corrente menor que 10 elevado a 12.");
} else {
    alert("Corrente menor que 10 elevado a 15.");
}
```

- CASO (*switch/case*) – Estrutura condicional de verificação de valores por igualdade, em que uma variável ou expressão determina um valor que deve alterar o fluxo de controle nesse caso específico. O valor armazenado na variável ou determinado na expressão é testado com todas as constantes (casos) e o fluxo é desviado para o caso em que os valores sejam iguais.

Sintaxe

```
switch (expressão/variável){
    case CONST1: instruções;
        break;

    case CONST2: instruções;
        break;

    default: instruções;
}
```

Exemplo

```
switch(document.formulario.rdUnidade.value)
{
    case "T" : R = R * Math.pow(10,12);
                break;
    case "G" : R = R * Math.pow(10,9);
                break;
    case "M" : R = R * Math.pow(10,6);
                break;
    case "k" : R = R * Math.pow(10,3);
                break;
```

```

        case "-": R = R * Math.pow(10,0);
                    break;
        case "m": R = R * Math.pow(10,-3);
                    break;
        case "mi": R = R * Math.pow(10,-6);
                    break;
        case "n": R = R * Math.pow(10,-9);
                    break;
        case "p": R = R * Math.pow(10,-12);
                    break;
        default: alert("Unidade não identificada");
    }
}

```

Exemplo js2.html:

```

<!DOCTYPE html>
<html>
    <body>
        <form name="formulario" method="post">
            <p>Clique sobre o botão para apresentar o exemplo de estrutura sequencial.</p>
            <button onclick="exemplo1()">Clique aqui!</button>
            <p>Clique sobre o botão para apresentar o exemplo de estrutura condicional SE(if).</p>
            <p>Idade:<input type="text" name="txtIdade"></p>
            <button onclick="exemplo2()">Clique aqui!</button>
            <p>Clique sobre o botão para apresentar o exemplo de estrutura condicional SE/SENÃO(if/else).</p>
            <p>Idade:<input type="text" name="txtIdade2"></p>
            <button onclick="exemplo3()">Clique aqui!</button>
            <p>Clique sobre o botão para apresentar o exemplo das estruturas condicionais CASO (switch) e SE/SENÃO SE(if/else if).</p>
            <p><input type="radio" name="rdTensao" value="110" checked>110v
            <input type="radio" name="rdTensao" value="220">220v
            Resistência:<input type="text" name="txtResistencia" value="1"></p>
            <p><input type="radio" name="rdUnidade" value="T">Tera
            <input type="radio" name="rdUnidade" value="G">Giga
            <input type="radio" name="rdUnidade" value="M">Mega
            <input type="radio" name="rdUnidade" value="k">kilo
        </form>
    </body>
</html>

```

```

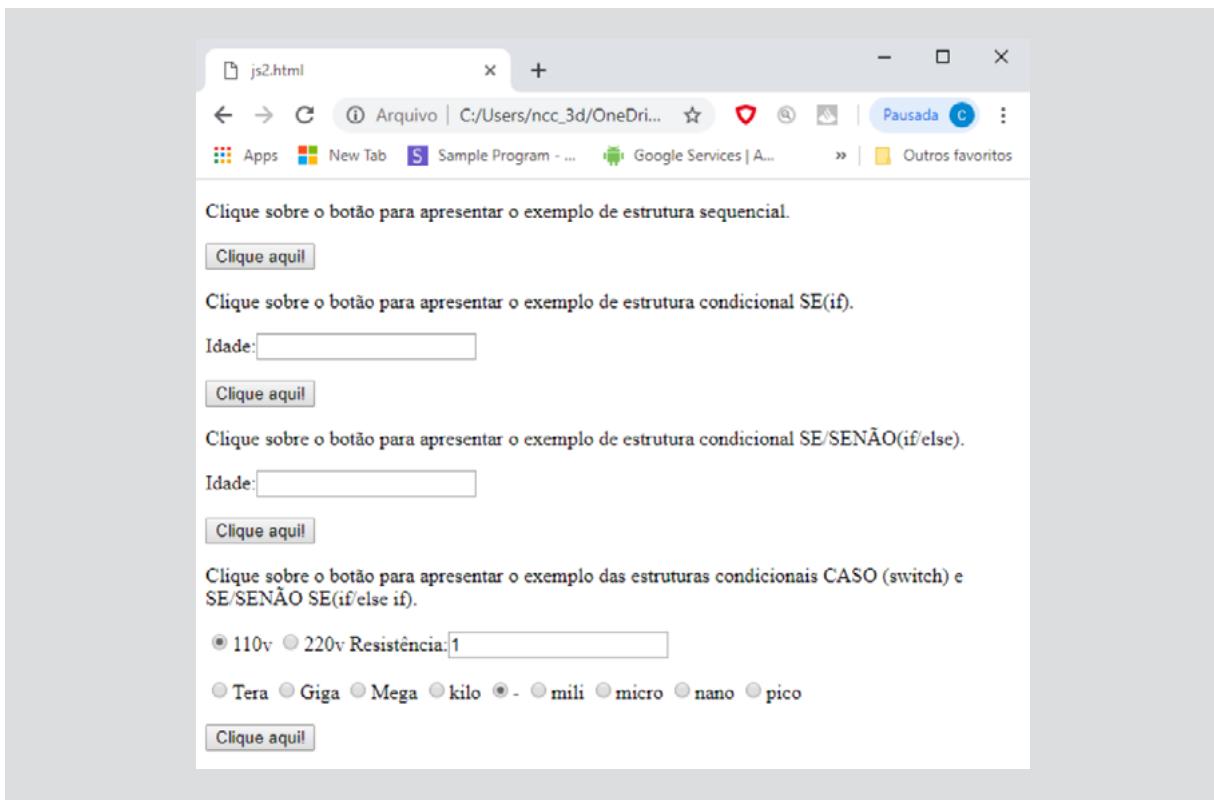
<input type="radio" name="rdUnidade" value="-" checked>
<input type="radio" name="rdUnidade" value="m">mili
<input type="radio" name="rdUnidade" value="mi">micro
<input type="radio" name="rdUnidade" value="n">nano
<input type="radio" name="rdUnidade" value="p">pico
</p>
<button onclick="exemplo4()">Clique aqui!</button>
</form>
<script type="text/javascript">
<!--
function exemplo1(){ // Estrutura sequencial
    var raio = 2.5;
    var areaCirc = Math.PI * Math.pow(raio,2);
    alert("Área da Circunferência= " + areaCirc);
}
function exemplo2(){ // Estrutura condicional SE
    var idade = document.formulario.txtIdade.value;
    if(idade>=18){
        alert("Você já tem a maior idade.");
    }
}
function exemplo3(){ // Estrutura condicional SE/SENÃO
    var idade = document.formulario.txtIdade2.value;
    if(idade>=18){
        alert("Você já tem a maior idade.");
    } else {
        alert("Você ainda não tem a maior idade.");
    }
}
function exemplo4(){// Estrutura condicional SE/SENÃO SE e CASO
var U = document.formulario.rdTensao.value;
    var R = document.formulario.txtResistencia.value;
    switch(document.formulario.rdUnidade.value) {
        case "T"      : R = R * Math.pow(10,12);
        case "G"      : R = R * Math.pow(10,9);
                        break;
        case "M"      : R = R * Math.pow(10,6);
                        break;
        case "k"      : R = R * Math.pow(10,3);
                        break;
    }
}
-->

```

```

        case “-”      : R = R * Math.pow(10,0);
                        break;
        case “m”       : R = R * Math.pow(10,-3);
                        break;
        case “mi”      : R = R * Math.pow(10,-6);
                        break;
        case “n”       : R = R * Math.pow(10,-9);
                        break;
        case “p”       : R = R * Math.pow(10,-12);
                        break;
        default:        alert(“Unidade não identificada”);
    }
    var I = U/R;
    if(I<=Math.pow(10,-9)){
        alert(“Corrente menor que 10 elevado a -9.”);
    } else if (I<=Math.pow(10,-6)){
        alert(“Corrente menor que 10 elevado a -6.”);
    } else if (I<=Math.pow(10,-3)){
        alert(“Corrente menor que 10 elevado a -3.”);
    } else if (I<=Math.pow(10,0)){
        alert(“Corrente menor que 10 elevado a 0.”);
    } else if (I<=Math.pow(10,3)){
        alert(“Corrente menor que 10 elevado a 3.”);
    } else if (I<=Math.pow(10,6)){
        alert(“Corrente menor que 10 elevado a 6.”);
    } else if (I<=Math.pow(10,9)){
        alert(“Corrente menor que 10 elevado a 9.”);
    } else if (I<=Math.pow(10,12)){
        alert(“Corrente menor que 10 elevado a 12.”);
    } else {
        alert(“Corrente menor do que 10 elevado a 15.”);
    }
}
-->
</script>
</body>
</html>

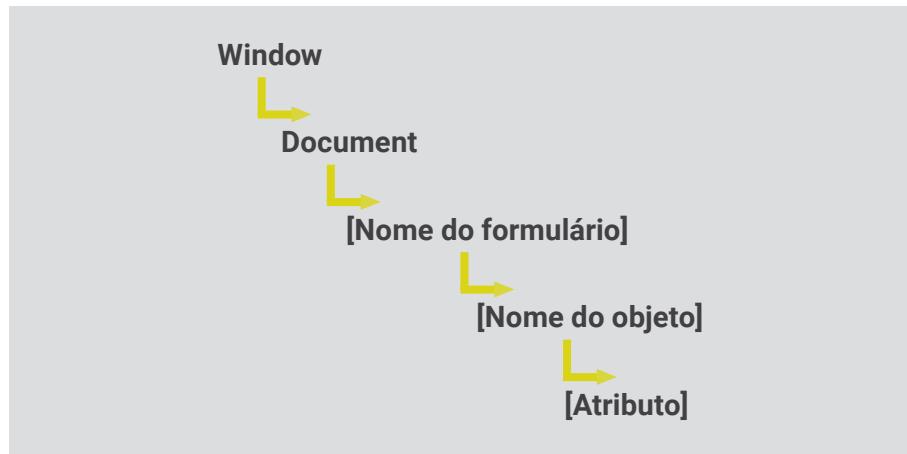
```



Experimente esse exemplo, que utiliza na prática a estrutura sequencial e todas as estruturas condicionais que aprendemos nesta unidade. Note que, para o cálculo corrente de um circuito, temos a entrada da tensão por meio de botões de rádio, nos quais podemos escolher entre 110 e 220v. A unidade de medida da resistência é escolhida entre nove diferentes grandezas. A grandeza escolhida será usada para determinar o valor correto da resistência por meio de uma estrutura CASO e a mensagem para o usuário será determinada por meio de uma estrutura SE/SENÃO SE. Realize os testes com as diferentes grandezas. Não é necessário alterar o valor da resistência, que foi inicialmente definido como 1.

A instrução `var idade = document.formulario.txtIdade.value;` busca para a função o valor digitado pelo usuário no `document` (página), nome do formulário e nome do componente do formulário e armazena o valor na variável `idade`. Esse modo pode ser usado para buscar valores digitados ou escolhidos pelo usuário em qualquer formulário.

Podemos alterar ou consultar todos os atributos de um objeto. Para isso, devemos encontrá-lo em sua hierarquia de objetos:



Vetores – Podem ser definidos com valores específicos ou declarados para receber os valores posteriormente:

```
var nomes = ["Ana", "Carlos", "João"];
```

ou

```
var nomes = new array("Ana", "Carlos", "João");
```

```
var coeficientes = [12, 13.5, 10, 6, 7.8];
```

ou

```
var coeficientes = new array(12, 13.5, 10, 6, 7.8);
```

Alguns exemplos de uso de vetores:

- Você pode alterar um valor com: nomes[0] = "Maria".
- Você pode saber a quantidade de elementos de vetor: qtd = nomes.length.
- Você pode incluir um elemento: nomes[nomes.length + 1] = "Marcelo".
- Você pode excluir o último elemento: nomes.pop().

Repetição – Estrutura que determina a repetição de um conjunto de instruções um determinado número de vezes ou até que uma condição seja satisfeita.

- **ENQUANTO (while)** – Estrutura de repetição que repetirá o bloco de código enquanto a condição de controle for verdadeira, alterando o controle de fluxo normal. O bloco será repetido até a condição se tornar falsa. Dessa forma, a repetição se encerrará e o fluxo de controle seguirá para a primeira linha de instrução após o encerramento do bloco. Caso a condição seja falsa no início, a repetição não ocorrerá.

Sintaxe

```
while(condição) {  
    //instruções  
}
```

Exemplo

```
var nomes = ["Ana", "Carlos", "João"];  
var i = 0;  
while(i < 3) {  
    alert("Olá " + nomes[i]);  
    i++;  
}
```

- FAÇA/ENQUANTO (*do/while*) – Estrutura de repetição que repetirá o bloco de código enquanto a condição de controle for verdadeira, mas o teste só é realizado ao final do bloco, o que faz com que o bloco seja executado pelo menos uma vez, mesmo que a condição seja falsa, fazendo com que o controle de fluxo normal seja alterado. O bloco será executado pelo menos uma vez e será repetido até a condição se tornar falsa. Dessa forma, a repetição se encerrará e o fluxo de controle seguirá para a primeira linha de instrução após o encerramento do bloco.

Sintaxe

```
do {  
    //instruções  
} while(condição);
```

Exemplo

```
var nomes = ["Ana", "Carlos", "João"];
var i = 0;
do {
    alert("Olá " + nomes[i]);
    i++;
}while(i < 3);
```

- PARA (*for*) – Estrutura de repetição que permite que um bloco de código seja repetido um determinado número de vezes. A condição de controle dentro do *for* repetirá o bloco enquanto a condição *for* verdadeira e, assim, o controle do fluxo será alterado. O bloco será repetido de acordo com uma variável contadora e até que a condição (em função da variável contadora) se torne falsa. Dessa forma, a repetição se encerrará e o fluxo de controle seguirá para a primeira linha de instrução após o encerramento do bloco. Caso a condição seja falsa no início, a repetição não ocorrerá. Devem ser definidos: o valor inicial da variável de controle (contadora), a condição em função da variável de controle e o passo de mudança a cada interação da variável de controle.

Sintaxe

```
for(inicio; condição; passo){
    //instruções
}
```

OU

```
for(item in array){
    //instruções
}
```

Exemplo

```
var nomes = ["Ana", "Carlos", "João"];
for(i=0; i<3;i++){
    alert("Olá " + nomes[i])
}
```

OU

```
var nomes = ["Ana", "Carlos", "João"];
for (var i in nomes) {
    alert("Olá " + nomes[i]);
}
```



MEDIATECA

Acesse a midiateca da Unidade 4 e veja o conteúdo complementar indicado pelo professor sobre introdução ao uso da programação JavaScript.



Saiba mais

Para saber mais sobre as estruturas de programação em JavaScript, leia a seção **Programação em JavaScript básica** (p. 579-583) do seguinte livro:

LEMAY, L. **Aprenda a criar páginas web com HTML e XHTML em 21 dias.** São Paulo: Pearson, 2002. Biblioteca Virtual.

Desenvolvimento de aplicações com o uso de programação com JavaScript

Uma aplicação cliente para a web deve responder à interação do usuário, seja em um componente (elemento da página), ao se carregar ou descarregar a página, ou na hora de submeter os dados de um formulário a um servidor.

Vamos analisar os exemplos a seguir de configuração de eventos que são comumente utilizados em aplicações web clientes.

Principais eventos

Eventos permitem que ações possam ser realizadas a partir de uma interação do usuário. Podemos relacionar um componente a uma função JavaScript por meio da configuração de um evento que, ao ser disparado, execute uma determinada função da aplicação.

Evento	Descrição
onInput	Ocorre quando um elemento do tipo input tem seu valor modificado.
onClick	Ocorre quando é realizado um clique com o mouse sobre o componente.
onDbclick	Ocorre quando é realizado um duplo clique com o mouse sobre o componente.
onMousemove	Ocorre quando o mouse é movimentado sobre um componente.
onMousedown	Ocorre quando se aperta o botão do mouse sobre o componente.
onMouseup	Ocorre quando se solta o botão do mouse, útil para se trabalhar com o drag'n'drop.
onKeypress	Ocorre quando se pressiona e solta uma tecla em um componente de entrada.
onKeydown	Ocorre quando se pressiona uma tecla em um componente de entrada.
onKeyup	Ocorre quando se solta uma tecla em um componente de entrada.
onBlur	Ocorre quando um elemento deixa de estar em foco e perde o foco para outro componente. É muito útil para analisar dados preenchidos ou escondidos após o usuário trocar de campo.

onFocus	Ocorre quando um elemento ganha o foco e passa a ser o elemento de interface do usuário. É muito útil para realização de operações de configuração de um componente em função de escolhas feitas pelo usuário em outros componentes.
onChange	Ocorre quando um input, select ou textarea tem seu valor alterado pelo usuário.
onLoad	Ocorre quando a página é carregada e se podem realizar configurações específicas para o usuário.
onUnload	Ocorre quando a página é fechada (encerrada).
onSubmit	Ocorre quando o usuário aciona o botão de submit. É muito útil para realizar análises e verificações dos dados preenchidos pelo usuário antes de enviá-los ao servidor.

Ainda existem outros eventos, mas eles não serão discutidos aqui.

Caixas de texto

Exemplo: js4.html

```
<!DOCTYPE html>
<html>
<head><title>Validando Dados</title></head>
<body bgColor=white>
<center><font face="arial" size="5">Dados do Cliente:</font></center>
<form name="formulario" method="post">
<table border=0>
<tr>
<td>Nome: </td>
<td><input size="30" name="nome"> </td>
<td>Telefone: </td>
<td><input size="15" name="telefone"> </td>
</tr>
<tr>
<td>Data:</td>
<td><input size="3" name="dia"></input size="3" name="mes"></input size="3" name="ano">
</td>
</tr>
```

```

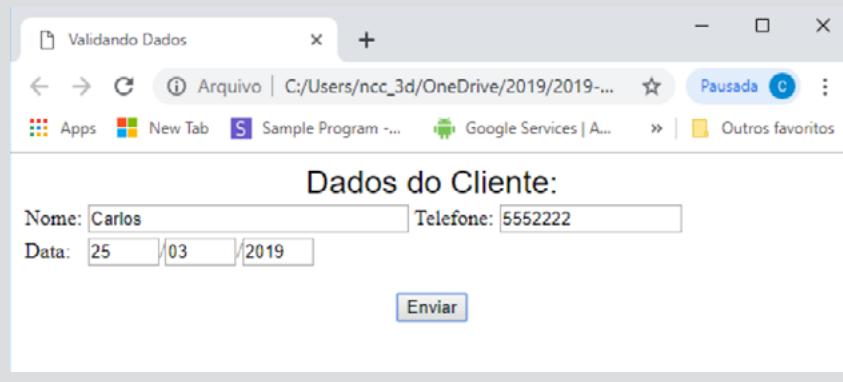
</table>
</center> <br>
<center><input onClick="validar()" type="button" value="Enviar" name="botao">
</center>
</form>
<script Language="JavaScript">
<!--
function validar(){
var nome=document.formulario.nome.value;
var telefone=document.formulario.telefone.value;
var dia=document.formulario.dia.value;
var mes=document.formulario.mes.value;
var ano=document.formulario.ano.value;
var validaNome=false;
var validaTelefone=false;
var validaData=false;
if(nome == "") {
    alert("O campo nome está vazio!");
    document.formulario.nome.focus();
    return false;
}
if(nome.length < 4) {
    alert("O seu nome parece incompleto!");
    document.formulario.nome.focus();
    return false;
}
if(isNaN(nome)== false) {
    alert("Digite um nome, você digitou um número!");
    document.formulario.nome.focus();
    return false;
}
if(isNaN(nome)== true) {
    validaNome=true;
}
if(telefone== "") {
    alert("O campo telefone está vazio!");
    document.formulario.telefone.focus();
    return false;
}
}

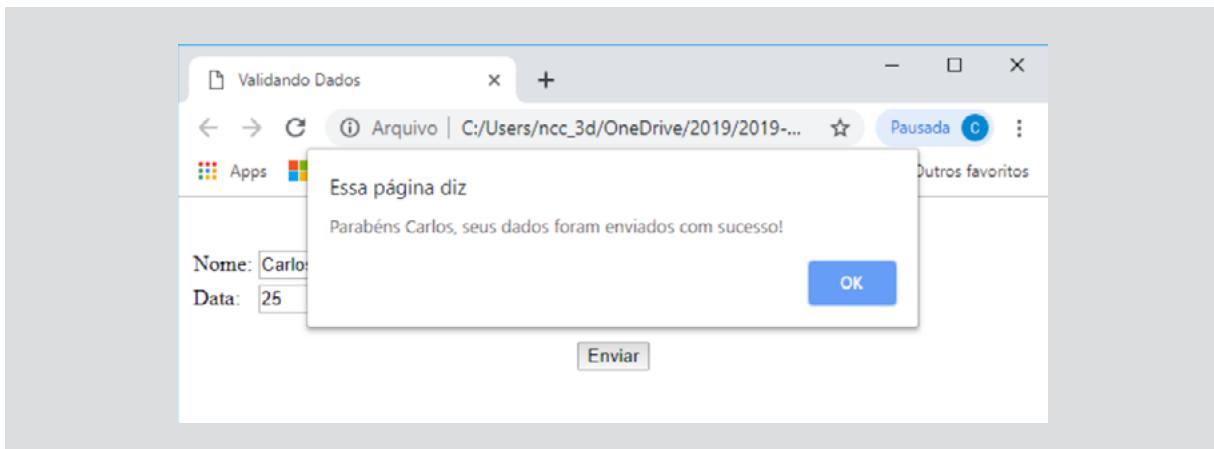
```

```

}
if(isNaN(telefone)== true) {
    alert("Digite somente números!");
    document.telefone.focus();
    return false;
}
if(isNaN(telefone)== false) {
    validaTelefone=true;
}
if(dia=="" || mes=="" || ano=="") {
    alert("Algum campo da data está vazio!");
    document.formulario.dia.focus();
    return false;
}
if(isNaN(dia)==true || isNaN(mes)==true || isNaN(ano)== true) {
    alert("Digite somente números nos campos da data!");
    document.formulario.dia.focus();
    return false;
}
if(isNaN(dia)==false && isNaN(mes)==false && isNaN(ano)==false) {
    validaData=true;
}
if(validaNome==true && validaTelefone==true && validaData==true) {
    alert("Parabéns " + nome + ", seus dados foram enviados com sucesso!");
    return true;
}
}
-->
</script>
</body>
</html>

```





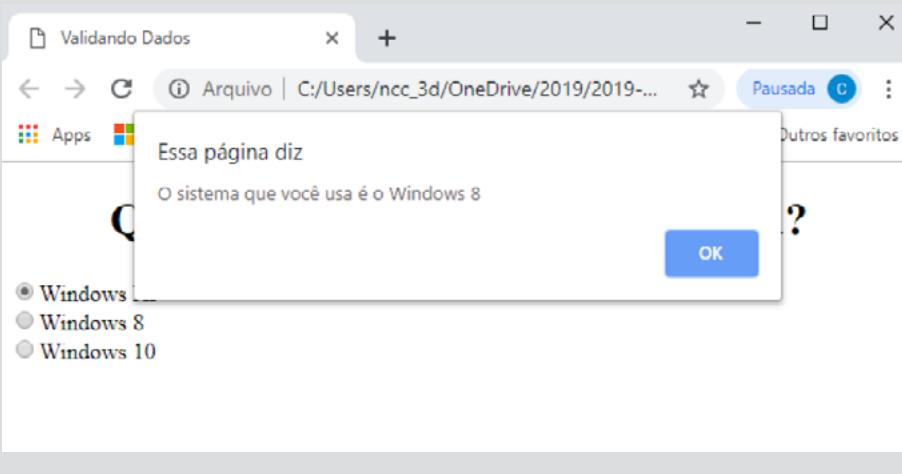
Analizando o código:

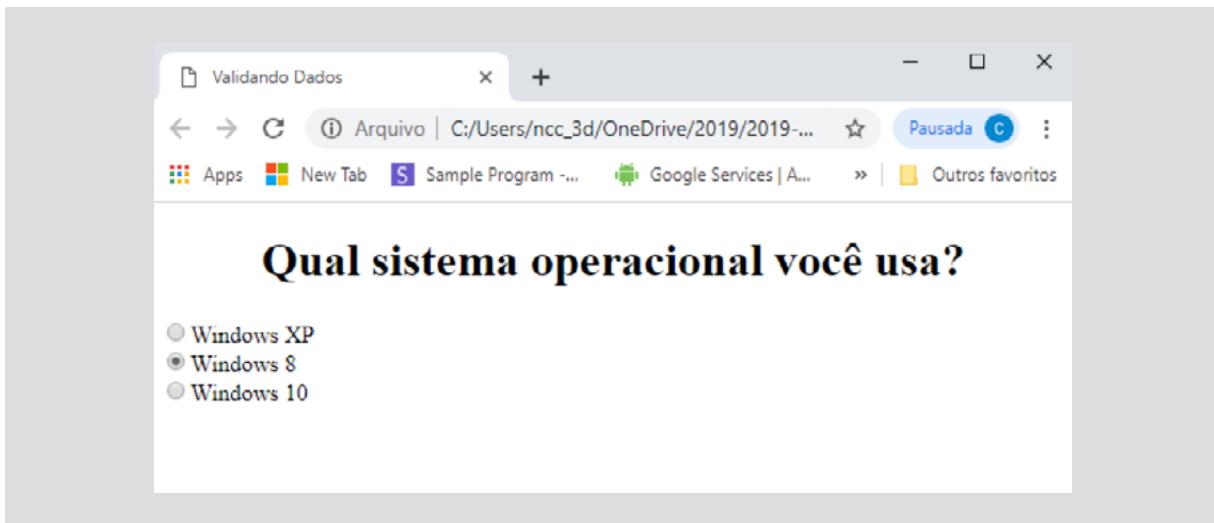
- A função de validação de dados foi relacionada ao elemento botão (botao) por meio do evento: onClick="validar()".
- Foram criadas cinco variáveis, uma para cada entrada de dados, e elas tiveram seus valores recebidos do componente relacionado.
- Foram criadas três variáveis para identificar se os dados do nome, telefone e data foram validados.
- Para o campo nome, foi verificado se ele se encontra vazio, com texto menor do que quatro letras, ou se foram digitados números, em vez do nome. Caso o campo nome seja validado, a variável validaNome recebe true.
- Para o campo telefone, foi verificado se ele se encontra vazio ou se foi digitado um texto. Caso o campo telefone seja validado, a variável validaTelefone recebe true.
- Para a data, foi verificado se os campos dia, mês e ano se encontram vazios ou se foi digitado um texto em algum deles. Caso o campo data seja validado, a variável validaData recebe true.
- Após os testes de validação, se as variáveis validaNome, validaTelefone e validaData estiverem true, a mensagem de parabéns será exibida e o conteúdo poderá ser enviado ao servidor, já que a função retornou true.

Botões de rádio (radio)

Exemplo: js5.html

```
<!DOCTYPE html>
<html>
<head><title>Validando Dados</title></head>
<body bgColor=white>
<h1 align=center>Qual sistema operacional você usa?</h1>
<form name=form1>
<input type=radio name=r1 onClick="verifica(0)" >Windows XP<br>
<input type=radio name=r1 onClick="verifica(1)" >Windows 8<br>
<input type=radio name=r1 onClick="verifica(2)" >Windows 10<br>
</form>
<script Language="JavaScript">
<!--
function verifica(escolha){
var msistema=escolha;
var sistemas=new Array();
sistemas[0]="Windows XP";
sistemas[1]="Windows 8";
sistemas[2]="Windows 10";
alert("O sistema que você usa é o " + sistemas[msistema])
}
//-->
</script>
</body>
</html>
```





Analizando o código:

- A função verifica recebe como parâmetro o valor equivalente à escolha do usuário e, a partir da escolha, é determinado o elemento do vetor de sistemas operacionais que está relacionado à escolha do usuário.
- O evento onClick foi relacionado ao clique de cada botão de rádio.
- A execução da função ocorre antes mesmo da atualização do botão de rádio.
- Somente após o fechamento da mensagem é que a página é atualizada.

Caixa de confirmação (checkbox)

Exemplo: js6.html

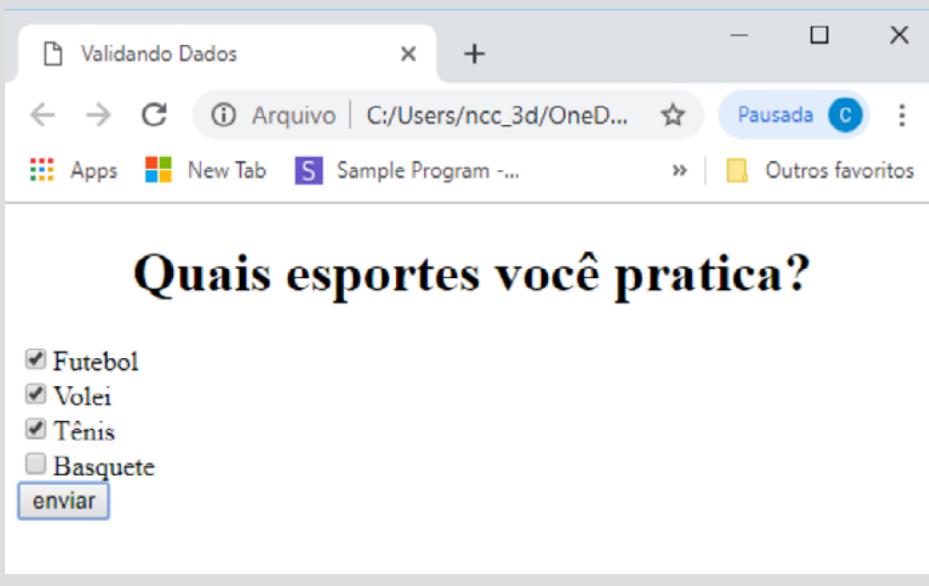
```
<!DOCTYPE html>
<html>
<head><title>Validando Dados</title></head>
<body bgColor=white>
<h1 align=center>Quais esportes você pratica?</h1>
<form name=form1>
<input type=checkbox name=chkfutebol>Futebol<br>
<input type=checkbox name=chkvolei>Volei<br>
<input type=checkbox name=chktenis>Tênis<br>
<input type=checkbox name=chkbasquete>Basquete<br>
```

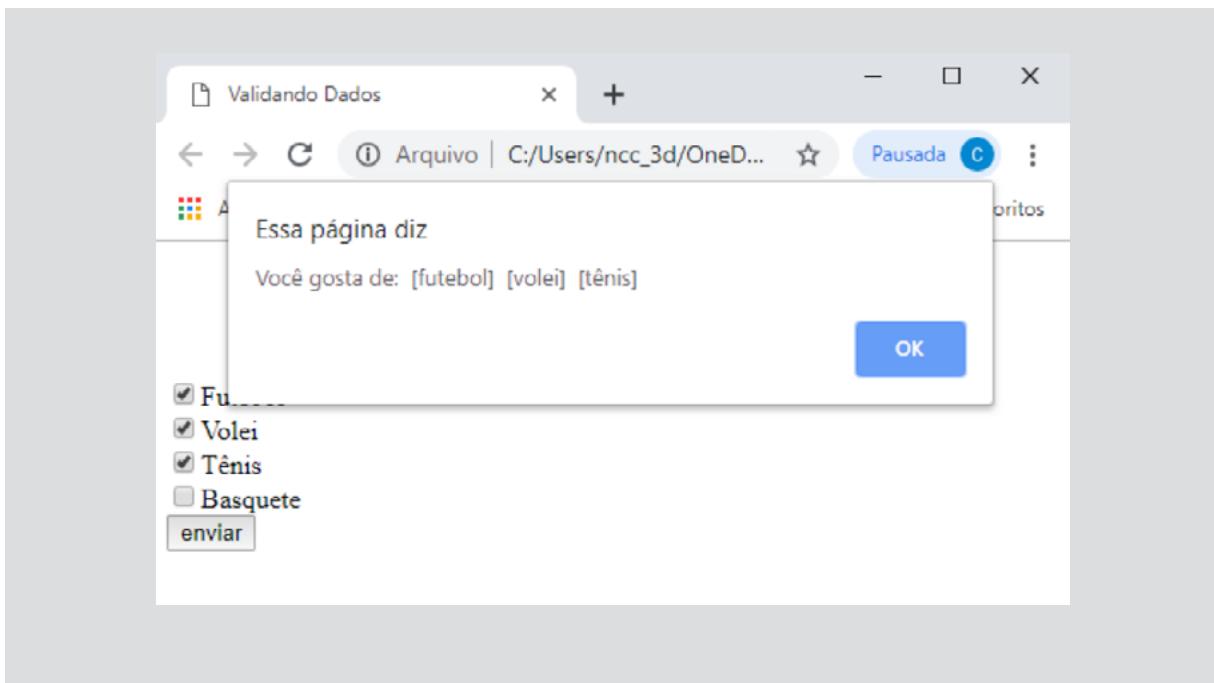
```

<input type=button value=enviar onClick="ver()">
</form>

<script Language="JavaScript">
<!--
<!--
function ver(){
var resposta="";
if(document.form1.chkfutebol.checked==true) {
resposta += " [futebol] ";
}
if(document.form1.chkvolei.checked==true) {
resposta += " [volei] ";
}
if(document.form1.chktenis.checked==true) {
resposta += " [tênis] ";
}
if(document.form1.chkbasquete.checked==true) {
resposta += " [basquete] ";
}
alert("Você gosta de: " + resposta);
}
//-->
</script>
</body>
</html>

```





Analizando o código:

- A função ver foi relacionada ao clique do botão.
- Quando o botão é acionado, a função analisa as caixas marcadas e prepara a mensagem acrescentando os esportes que foram selecionados.
- Ao final, é exibida a mensagem com todos os esportes escolhidos pelo usuário.

Verificando datas

Exemplo: js7.html

```
<!DOCTYPE html>
<html>
<head><title>Validando Dados</title></head>
<body bgColor=white>
<center><font face="arial" size="5">Data:</font></center>
<form name="formulario" method="post">
<table border=0>
<tr>
<td>Data:</td>
<td><input size="2" name="dia" onInput="testaDia()" />
<input size="2" name="mes" onInput="testaMes()" />
<input size="4" name="ano" />
```

```

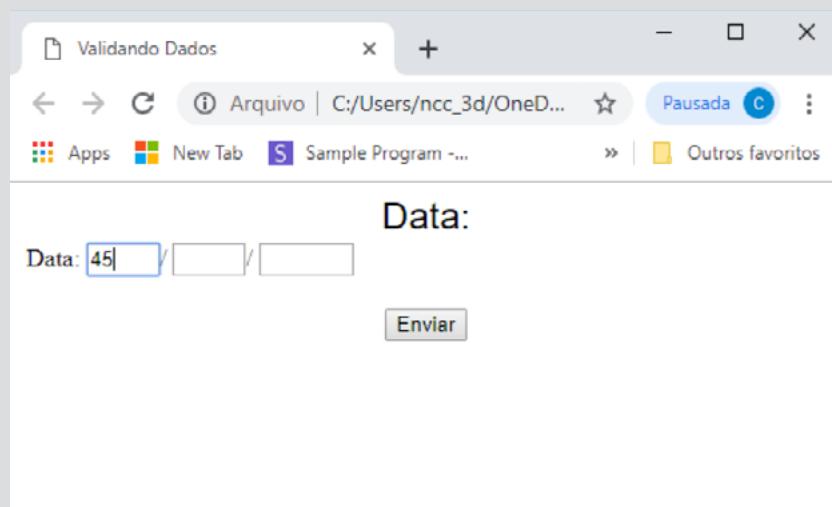
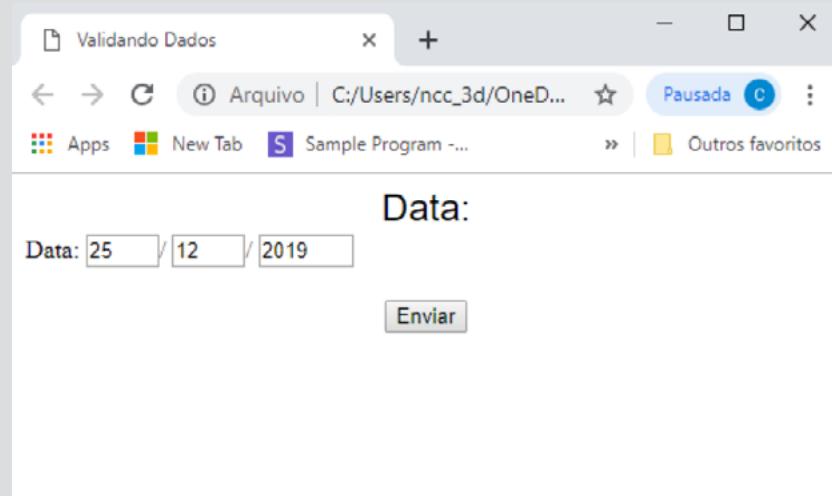
</tr>
</table>
</center> <br>
<center><input onClick="validar()" type="button" value="Enviar" name="botao"></
center>
</form>
<script Language="JavaScript">
<!--
function testaDia(){
var dia=document.formulario.dia.value;
if(isNaN(dia)==false && dia.length == 2 && dia >= 1 && dia <=31){
document.formulario.mes.focus();
}
else {
document.formulario.dia.focus();
}
}
function testaMes(){
var mes=document.formulario.mes.value;
if(isNaN(mes)==false && mes.length == 2 && mes >= 1 && mes <=12){
document.formulario.ano.focus();
}
else {
document.formulario.mes.focus();
}
}
function validar(){
var dia=document.formulario.dia.value;
var mes=document.formulario.mes.value;
var ano=document.formulario.ano.value;
var validaData=false;
if(dia=="" || mes=="" || ano=="") {
alert("Algum campo da data está vazio!");
document.formulario.dia.focus();
return false;
}
if(isNaN(dia)==true || isNaN(mes)==true || isNaN(ano)== true) {
alert("Digite somente números nos campos da data!");
document.formulario.dia.focus();
return false;
}

```

```

}
if(isNaN(dia)==false && isNaN(mes)==false && isNaN(ano)==false) {
    validaData=true;
}
if(validaData==true) {
    alert("Parabéns " + nome + ", seus dados foram enviados com sucesso!");
    return true;
}
}
-->
</script>
</body>
</html>

```



Analizando o código:

- As funções testaDia() e testaMes() analisam cada valor digitado pelo usuário e, se o dia estiver entre 1 e 31 com dois dígitos, passa o foco para o campo mês; caso contrário, o foco permanece no campo dia. Já se o mês estiver entre 1 e 12 e contiver dois dígitos, o foco irá para o campo ano; caso contrário, permanece no campo mês. As respectivas funções analisam se o conteúdo digitado pelo usuário é numérico, possui dois dígitos (usar zero na frente quando for o caso) e se o valor se encontra dentro dos limites inferior e superior.

Verificando e-mails

Exemplo: js8.html

```
<!DOCTYPE html>
<html>
<head><title>Validando Dados</title></head>
<body bgColor=white>
<center><font face="arial" size="5">E-mail:</font></center>
<form name="formulario" method="post">
<table border=0>
<tr>
<td>Data:</td>
<td><input size="40" name="email" onBlur="testEmail()">
</td>
</tr>
</table>
</center> <br>
<center><input onClick="validar()" type="button" value="Enviar" name="botao"></
center>
</form>
<script Language="JavaScript">
<!--
function testEmail(){
    var email=document.formulario.email.value;
    var validaEmail=false;
    if(email.indexOf("@")>0 && email.indexOf(".")>0 && email.length>10){
        validaEmail=true;
        document.formulario.botao.focus();
    }
}
```

```

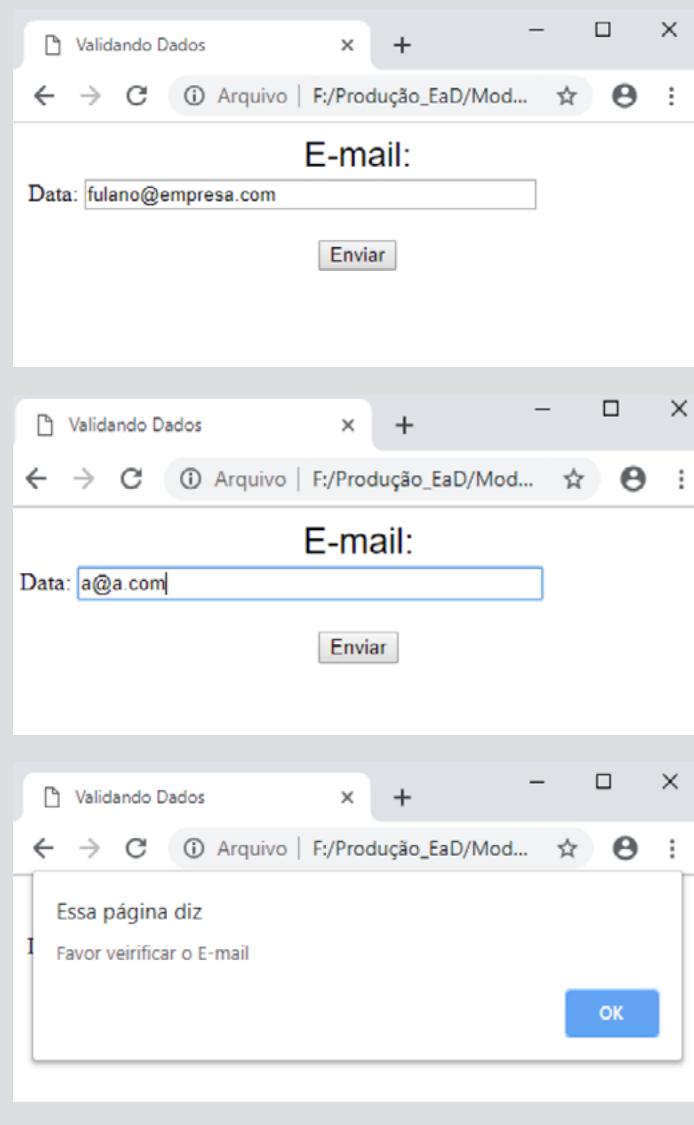
else {
    alert("Favor veirificar o E-mail");
    document.formulario.email.focus();
}
}

function validar(){
if (validaEmail==true){
    return true;
}
}

-->

</script>
</body>
</html>

```



Analizando o código:

- A função testaEmail() analisa o e-mail digitado pelo usuário e verifica se ele contém um símbolo de "@" e um ponto ".", além de determinar que um e-mail deve ter no mínimo 10 caracteres, incluindo os símbolos. Caso a condição seja verdadeira, o e-mail é considerado dentro do padrão e o foco é mudado para o botão; caso contrário, é exibida uma mensagem de alerta e o foco retorna ao campo e-mail. O evento onBlur ocorre quando o foco sai do elemento, ou seja, será executado quando o usuário mudar de elemento na interface.
- A função valida verifica se os dados estão dentro do padrão para serem enviados ao servidor.

Existem muitas outras formas de análise de dados utilizando o JavaScript antes do envio, mas apenas a prática e o contexto do problema permitirão definir adequadamente quais análises deverão ser realizadas sobre os dados antes do envio.



MEDIATECA

Acesse a midiateca da Unidade 4 e veja o conteúdo complementar indicado pelo professor sobre como criar e usar funções na linguagem JavaScript e como trabalhar com passagem de parâmetros.

Para ver os eventos possíveis em JavaScript, acesse: [Eventos HTML DOM](#).



Saiba mais

Para saber mais sobre o uso de métodos e validação de dados com JavaScript, leia o Capítulo 5 (p. 102-118) do seguinte livro:

MILETTO, E.; BERTAGNOLLI, S. **Desenvolvimento de software II**: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. Minha Biblioteca.



Dica

Para ter acesso a toda a biblioteca da linguagem JavaScript, vale a pena conhecer o site de tutorial do w3schools, que, mesmo estando em inglês, possui muitos exemplos práticos fáceis de entender e usar.



NA PRÁTICA

Todos os sites de venda pela internet necessitam não só de validação de dados, mas da execução de cálculos e outras funções. Sendo assim, você pode acessar qualquer site de vendas, tal como o <https://www.amazon.com.br/>, e analisar os códigos JavaScript utilizando a função (F12). Assim, você pode analisar e conhecer a programação JavaScript incorporada à página.

Resumo da Unidade 4

Nesta unidade, abordamos a programação JavaScript, que é a linguagem mais usada para validar dados em aplicações Web clientes. É uma linguagem de fácil aprendizado e suportada por todos os principais navegadores para internet. Vimos como criar funções JavaScript para realizar operações na página, assim como aplicamos esses conhecimentos na análise de dados a serem enviados ao servidor. Conhecemos as principais funções JavaScript para tratar dados, assim como aprendemos a relacionar a execução de determinadas funções a ações realizadas pelo usuário com determinados componentes. Dessa forma, com emprego de JavaScript em nossas interfaces (páginas clientes), elas se tornam mais poderosas e de fácil uso pelos usuários, assim como evitam maior trânsito de dados, uma vez que os dados são criticados antes do envio. Isso evita que os dados sejam enviados ao servidor e que o servidor retorne informando que um ou mais dados não estão de acordo com o padrão determinado pela aplicação.



CONCEITO

Nesta unidade, destacou-se o emprego da linguagem JavaScript no aperfeiçoamento de páginas Web clientes. O emprego dessa linguagem permite incluir ações dinâmicas em nossas páginas, assim como verificar separadamente cada dado digitado pelo usuário, evitando que sejam enviados ao servidor dados incorretos. Essa capacidade de utilizar ações programadas nas aplicações Web clientes evita que o tráfego de dados entre a aplicação cliente e servidor aumente, sobrecarregando o serviço e, logicamente, o servidor.

Referências

LEMAY, L. **Aprenda a criar páginas web com HTML e XHTML em 21 dias.** São Paulo: Pearson, 2002. Biblioteca Virtual.

MILETTO, E.; BERTAGNOLLI, S. **Desenvolvimento de software II:** introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. Minha Biblioteca.

SEGURADO, V. S. **Projeto de interface com o usuário.** São Paulo: Pearson, 2015. Biblioteca Virtual.

