Jefersson Santiago

Web Standard

Senac

Dados Internacionais de Catalogação na Publicação (CIP)
(Jeane Passos de Souza – CRB 8ª/6189)

Santiago, Jefferson
Web Standards / Jefferson Santiago. – São Paulo : Editora Senac São Paulo, 2020. (Série Universitária)

Bibliografia.
e-ISBN 978-65-5536-221-3 (ePub/2020)
e-ISBN 978-65-5536-222-0 (PDF/2020)

1. HTML (Linguagem de programação) 2. CSS (Linguagem de programação) 3. Websites – Criação 4. Websites – Desenvolvimento 5. Websites – Design I. Título. II. Série.

20-1173t

CDD – 005.133
BISAC COM060130
COM051270

Índice para catálogo sistemático
1. HTML : Linguagem de programação : Computadores : Processamento de dados 005.133
2. CSS : Linguagem de programação : Computadores : Processamento de dados 005.133

WEB STANDARDS

Jefferson Santiago

Administração Regional do Senac no Estado de São Paulo

Presidente do Conselho Regional
Abram Szajman

Diretor do Departamento Regional
Luiz Francisco de A. Salgado

Superintendente Universitário e de Desenvolvimento
Luiz Carlos Dourado

Editora Senac São Paulo

Conselho Editorial
Luiz Francisco de A. Salgado
Luiz Carlos Dourado
Darcio Sayad Maia
Lucila Mara Sbrana Sciotti
Jeane Passos de Souza

Gerente/Publisher
Jeane Passos de Souza (jpassos@sp.senac.br)

Coordenação Editorial/Prospecção
Luís Américo Tousi Botelho (luis.tbotelho@sp.senac.br)
Dolores Crisci Manzano (dolores.cmanzano@sp.senac.br)

Administrativo
grupoedsadministrativo@sp.senac.br

Comercial
comercial@editorasenacsp.com.br

Acompanhamento Pedagógico
Otacília da Paz Pereira

Designer Educacional
Sueli Brianezi Carvalho

Revisão Técnica
Thyago Conchado Quintas

Preparação e Revisão de Texto
Bianca Rocha

Projeto Gráfico
Alexandre Lemes da Silva
Emília Corrêa Abreu

Capa
Antonio Carlos De Angelis

Editoração Eletrônica
Stephanie dos Reis Baldin

Ilustrações
Stephanie dos Reis Baldin

Imagens
Adobe Stock

E-pub
Ricardo Diana

Editora Senac São Paulo
Rua 24 de Maio, 208 – 3º andar
Centro – CEP 01041-000 – São Paulo – SP
Caixa Postal 1120 – CEP 01032-970 – São Paulo – SP
Tel. (11) 2187-4450 – Fax (11) 2187-4486
E-mail: editora@sp.senac.br
Home page: http://www.livrariasenac.com.br

Sumário

Capítulo 1

Introdução ao HTML

A ideia deste capítulo é fornecer subsídios para que você seja capaz de desenvolver a estrutura básica de uma página HTML, de acordo com os padrões web recomendados pelo World Wide Web Consortium (W3C), principal organização de padronização da web.

Além disso, você aprenderá um pouco sobre a origem da web e da linguagem HTML. Vai aprender que ambas foram idealizadas para serem acessadas independentemente de cultura, idioma e tipo de dispositivo e que, atualmente, conceitos como "acessibilidade" e "usabilidade" passam a fazer parte desses ideais, fazendo com que a web seja mais inclusiva.

Esta obra pretende mostrar a você, leitor e futuro desenvolvedor, que os pais da web a idealizaram, mas o poder e a responsabilidade de concretizar essas ideias estão em suas mãos.

Esperamos que você se sinta inspirado a absorver o conteúdo destes capítulos, desenvolvendo páginas web para todos, respeitando as normas e sendo um facilitador de acesso para pessoas com deficiência (PcD). Enfim, que seu código seja "mais humano".

Prepare-se, futuro desenvolvedor. Aqui começa uma de suas jornadas!

1 A internet antes do HTML

Ok, você está com este livro nas mãos, com um monte de ideias de projetos e vontade de já começar a escrever códigos, mas tenha calma, porque você precisa antes entender um pouco da história do HTML. Afinal, um desenvolvedor pleno precisa saber não apenas como desenvolver, mas também o motivo de existir a linguagem que ele usa.

Vamos começar entendendo, de forma resumida, como era a internet antes da web (sim, são coisas diferentes).

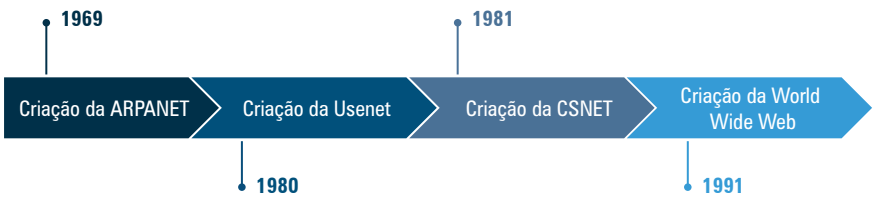
1.1 Algumas redes

Basicamente, tudo começou com a ARPANET, que era uma rede de computadores financiada pela Agência de Projetos e Pesquisa Avançada (ARPA) do Departamento de Defesa dos Estados Unidos.

Depois, surgiram outras redes, como a Usenet, meio de comunicação com a função de juntar grupos de discussão (newsgroups), também chamados de fóruns, e a Rede de Ciências da Computação (CSNET).

Por fim, surgiu a World Wide Web, conhecida também como web. Essa rede foi criada por Tim Berners-Lee, que, atualmente, é diretor do W3C. Trataremos do W3C novamente quando abordarmos o assunto "normas web".

Figura 1 – Linha do tempo do surgimento das redes



PARA SABER MAIS

Um dos poucos documentos que contam a história da ARPANET é *A history of the ARPANET: the first decade* (BOLT BERANEK AND NEWMAN, 1981).

1.2 Os protocolos

Muito bem. Agora que entendemos que existiam e existem outras redes além da web, vamos entender como era feito o acesso a cada uma delas.

Um dos protocolos mais antigos é o Telnet, usado para transmissão de dados e comunicação por texto. O acesso era muito semelhante ao sistema operacional MS-DOS, que usa comandos em um terminal.

Outro protocolo, mas não tão antigo, era o IRC, que servia basicamente de chat e troca de arquivos, e precisava de um cliente, como o software mIRC, para ser acessado.

Um dos protocolos mais conhecido é o HTTP, também criado por Tim Berners-Lee e que tem por finalidade fazer a troca de hipertextos, ou seja, elementos conectados por palavras, imagens, documentos, etc., que, quando conectados, formam uma grande rede de informação. Porém, quando ele foi idealizado, não existia o conceito de sites e navegadores como conhecemos hoje.

Na atualidade, existem outros diversos tipos de protocolos, como o FTP, usado para transferir arquivos (você vai usá-lo para colocar as páginas do seu site no servidor), o SMTP, usado para envio de e-mail, e o POP3, usado para receber e-mail.

IMPORTANTE

Talvez você esteja se perguntando: "Por que é importante conhecer esses protocolos?". Em sua carreira profissional, muito provavelmente você vai se deparar com situações em que precisará desenvolver soluções para clientes e prestar consultoria sobre as possibilidades existentes ao implementar um projeto. Nesse momento, conhecer quais são os protocolos usados para cada finalidade é fundamental para se ter êxito.

1.3 Uma breve explicação sobre a deep web

Conhecemos sites que podem ser rastreáveis pelos mecanismos de busca e que possuem um Domain Name Server (DNS), que é o que dá uma identificação para um site. Essa identificação é chamada de domínio. Um exemplo de site com identificação é: https://sp.senac.br.

Também é possível encontrar um site quando ele não possui um domínio. Para isso, usamos a identificação do computador na rede, como o número IP. Um exemplo é o endereço IP que todo usuário pode acessar para ver as configurações do seu roteador, como: http://192.168.0.1 ou http://192.168.1.1.

IMPORTANTE

Com o uso do protocolo HTTP, é possível acessar páginas web, estando elas em um servidor na web ou armazenadas no próprio computador da pessoa. Temos outros tipos de acesso que podemos fazer, tal como transferir arquivos. Um exemplo de acesso de transferência de arquivo é: ftp://nomedodominio.co.uk.

Mas será que é possível guardar alguma informação na internet que não seja rastreada pelos navegadores ou acessada por HTTP? Sim, isso ocorre na deep web. Pessoas leigas podem achar que a deep web é um

lugar cheio de hackers, ladrões e outros tipos de crimes virtuais. Ok, isso pode ser verdade, mas o que a maioria não sabe é que a deep web também possui uma "superfície" e que não se trata de um lugar ruim.

Nesse local, você pode guardar arquivos inofensivos que não deseja que sejam vistos pelos visitantes do seu site. Por exemplo, se você colocar um arquivo de extensão .psd no servidor do seu site, seus leitores não terão acesso a ele. Essa camada na qual ficam salvos esses arquivos inofensivos também faz parte da deep web. Interessante, não?

1.4 A guerra dos navegadores

Como você já sabe, para poder visualizar as páginas de internet, foram criados softwares conhecidos como navegadores.

O primeiro navegador foi criado em 1991 por Tim Berners-Lee e se chamava World Wide Web. Ficou confuso? Pois é, Tim e sua equipe perceberam que colocar no navegador o mesmo nome da rede criada um ano antes iria causar confusão, por isso mudaram o nome do navegador para Nexus. Porém, o que mais se popularizou foi um navegador criado em 1993, chamado Mosaic. Ele se tornou mais popular porque rodava em Windows, e não baseado em texto, como eram os outros.

O líder do projeto Mosaic era Marc Andreessen, que desistiu dele para fundar sua própria empresa, a Netscape. Em 1994, a Netscape lançou o Navigator, que se tornou o navegador mais popular da web.

Foi nessa época que começou a guerra dos navegadores. A Microsoft decidiu entrar para a internet e criou seu próprio navegador, o Internet Explorer, e suas próprias extensões para o HTML. A Netscape fez o mesmo. Cada uma com o objetivo de fazer o usuário escolher o seu software.

Essa guerra comercial ajudou a fazer com que muitas pessoas tivessem acesso à web, mas de formas diferentes, dependendo do navegador que usavam. Isso fez com que houvesse, ainda mais, a necessidade de padronização na web, que é o que vamos conhecer na sequência.

1.5 Normas web

Web Standards (conhecido no Brasil como Padrões Web) é um conjunto de normas com a finalidade de criar uma web padronizada e universal, ou seja, permitir o acesso independentemente da cultura do usuário e de seu idioma e incorporando ferramentas de acessibilidade, permitindo que as pessoas com deficiência possam também navegar e interagir com o mundo à sua volta.

Além disso, estamos em um momento de grande desenvolvimento da Internet das Coisas, um conceito no qual os objetos que usamos no nosso cotidiano estariam conectados também por meio da internet. Para que tantos objetos com finalidades diferentes e de marcas diferentes possam trabalhar sem conflitos, é importante que existam padronizações.

Sobre isso, o W3C diz o seguinte:

> O W3C está focado em tecnologias para permitir o acesso à web em qualquer lugar, a qualquer hora, usando qualquer dispositivo. Isso inclui o acesso da web a partir de telefones celulares e outros dispositivos móveis, bem como o uso da tecnologia da web em eletrônicos de consumo, impressoras, televisão interativa e até automóveis. (W3C, 2019)

Agora que você já sabe como a internet evoluiu a ponto de ser padronizada, chegou a hora de conhecer um pouco mais da história e evolução da linguagem-mãe da web: o HTML, tema do nosso próximo tópico.

2 História do HTML

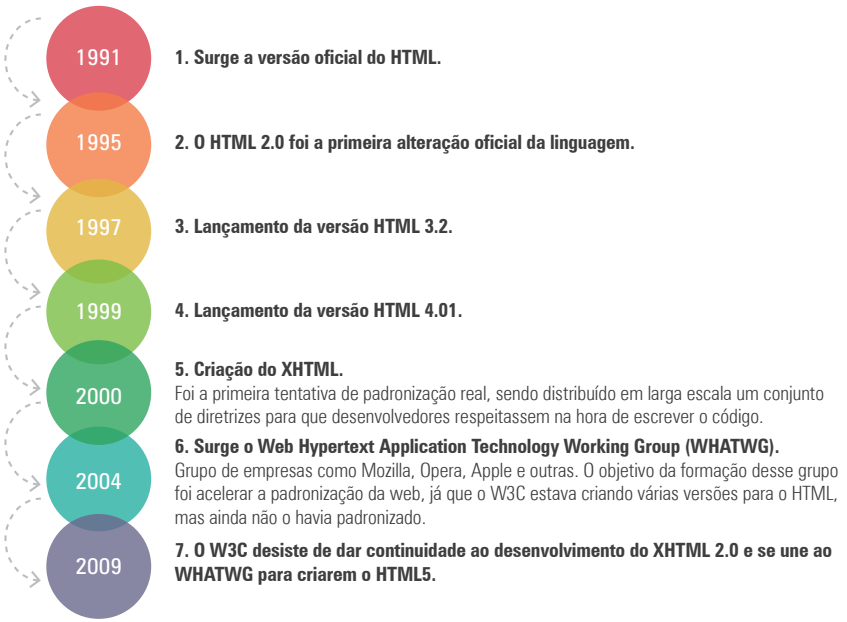
O HTML é a principal linguagem da web, e seu nome é a abreviação de HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto.

Como o próprio nome diz, o HTML utiliza marcações (conhecidas como tags) para a publicação de conteúdo (textos, imagens, vídeos, áudios, etc.) na web, e seu criador foi Tim Berners-Lee. Sim, o mesmo que criou a World Wide Web e o protocolo HTTP, que é justamente o protocolo que permite a troca de páginas feitas em HTML.

Você percebeu que a guerra dos navegadores incentivou ainda mais que houvesse uma padronização na web, certo? Todos esses esforços visam manter o HTML como uma linguagem independente de plataformas, impedindo que a web pertença a uma base proprietária ou tenha seu destino definido por ela.

A seguir, vamos entender como se deu a evolução do HTML.

Figura 2 – Evolução do HTML



Apesar de o HTML5 ter começado a ser projetado em 2009, como podemos perceber na figura 2, apenas em 2014 ele foi lançado, mas de uma forma revolucionária. Para ficar mais claro o impacto que o HTML

causou na web, no início era apenas uma linguagem que permitia que o usuário acessasse o conteúdo de forma passiva. Com o HTML5, foi possível criar verdadeiros apps de edição de imagem, construção de sites e edição de vídeo, permitindo que o usuário tenha uma navegação mais ativa e seja cocriador da web. Em 2016, ele passou por sua primeira atualização, sendo chamado de HTML5.1, e, mais recentemente, em 2017, surgiu sua nova versão, o HTML5.2

É com essa linguagem que iniciamos o estudo de Web Standards. Porém, para uma experiência mais completa, também usaremos outras duas linguagens, cada uma com uma função específica.

3 Fundamentos de HTML, CSS e JS

Além do HTML, o desenvolvedor precisa conhecer o CSS e o JavaScript (JS).

Mas, se desde o começo, o HTML sempre foi suficiente para criar páginas para a web, qual o motivo de terem criado mais duas linguagens? Há mais de uma resposta para isso.

A primeira e mais simples é que os usuários já foram iniciados nas novas tecnologias e possibilidades, e apresentar um conteúdo pobre em recursos, sejam visuais, em agilidade ou em interação, tende a ser considerado como irrelevante ou obsoleto. Isso impacta, por exemplo, o posicionamento das suas páginas nos mecanismos de busca.

A segunda é que, visando facilitar a compreensão, a manutenção e o desenvolvimento do código, foram separadas três partes: marcação, apresentação e comportamento.

- **Marcação:** é feita pela linguagem HTML, que é responsável por indicar o que cada conteúdo na página representa e fazer as ligações (links) com outros hipertextos.

- **Apresentação:** é feita pela linguagem CSS, que é uma linguagem de estilo e é responsável por indicar como cada conteúdo será apresentado na página. O HTML já cuidava do estilo antes, mas, com a criação do conceito tableless (no qual é recomendado que os desenvolvedores não utilizem tabelas para dividir as áreas da página nem utilizem o HTML para fazer a formatação), essa função fica exclusivamente restrita ao CSS.

- **Comportamento:** é feito pela linguagem JavaScript, que é uma linguagem de script responsável por deixar as páginas web mais dinâmicas, por meio de recursos como controle de eventos, que são ações disparadas assim que ocorre um evento na página (por exemplo: um clique ou quando a página é carregada, ou, ainda, quando um formulário for enviado, a página for rolada, etc.). É importante enfatizar que o JavaScript é uma linguagem de script que roda client-side, ou seja, no lado do cliente (significa que roda direto no navegador do usuário), diferentemente de outras linguagens, como PHP, por exemplo, que rodam no servidor.

Agora que você já sabe o que cada linguagem faz, vamos voltar para o HTML e estudar suas especificidades, começando pelos elementos, tags e atributos.

4 Elementos, tags e atributos do HTML

Olhe bem para a linha acima e para esta linha que você está lendo. Você consegue distinguir o que é um título do que é um parágrafo por causa do tamanho da fonte. É assim que nós, humanos, conseguimos distinguir uma informação de outra. Mas, quando você desenvolve um site, precisa pensar tanto nos humanos quanto nos robôs. Os robôs não conseguem distinguir visualmente um título de um parágrafo.

Portanto, cabe ao desenvolvedor marcar toda a informação usando tags específicas para cada finalidade, a fim de que ela possa ser

identificada não só para humanos, mas também para os diversos tipos de robôs (como mecanismos de busca) e softwares (como leitores de tela para pessoas com deficiência visual). Vamos ver como fazer isso usando os elementos do HTML.

4.1 Elementos

Os elementos são o que definem o que é cada coisa no código. Observe o texto a seguir:

Descascar as laranjas.

Ligar o espremedor.

Espremer as laranjas.

Desligar o espremedor.

Pegar um copo.

Colocar o suco no copo.

Beber o suco.

Anúncio em texto do Google. Clique aqui para comprar um espremedor.

Agora, imagine que você tem um robô e que dá o seguinte comando para ele: "Execute as instruções que estão em lista numerada".

Percebeu que a lista anterior não estava numerada? Mas, com o comando que você deu, visualmente a sua lista ficaria assim:

1. *Descascar as laranjas.*
2. *Ligar o espremedor.*
3. *Espremer as laranjas.*
4. *Desligar o espremedor.*

5. *Pegar um copo.*

6. *Colocar o suco no copo.*

7. *Beber o suco.*

Anúncio em texto do Google. Clique aqui para comprar um espremedor.

Pronto, agora está mais organizado, certo?

No primeiro exemplo, nós, humanos, conseguimos identificar as instruções, mas, se o robô tentasse executá-las mesmo assim, é capaz de que, no final, ele ainda comprasse um espremedor, por não conseguir distinguir a lista de um anúncio em texto!

Esta é a finalidade dos elementos: definir o que cada coisa representa, separando lista do que é anúncio, separando imagem do que é parágrafo, etc.

Alguns exemplos de elementos são:

- *label* (rótulo);
- *p* (parágrafo);
- *img* (imagem).

Ou seja, a letra "p" foi reservada para indicar que o texto relacionado a ela é um parágrafo. Mas, se simplesmente colocarmos a letra "p" na página, os navegadores não conseguirão distinguir o que é um elemento do que é simplesmente uma letra no texto da página. Por isso, todo elemento deve ser formado por tags.

4.2 Tags

As tags são elementos cercados por parênteses angulares (< >). Veja como ficaram os três exemplos anteriores usando as tags:

<label>

<p>

Existem dois tipos de tags: as que precisam de fechamento e as que não precisam de fechamento.

Para as tags que precisam de fechamento, abrimos com um parêntese angular (<), seguido do nome do elemento, e fechamos com o parêntese angular (>), e assim abrimos uma tag. Para o fechamento da tag, abrimos com um parêntese angular (<), seguido de barra (/) e do nome do elemento, e fechamos com um parêntese angular (>). Veja um exemplo de uma tag que precisa de fechamento:

```
<label>Rótulo da imagem</label>
```

Nesse caso, a frase "Rótulo da imagem" passa a ser o texto do elemento *label*. As tags servem para indicar onde o rótulo começa e onde ele termina. Qualquer texto antes ou depois das tags não será interpretado pelo navegador como um rótulo.

Agora, vamos ver um exemplo de tag que não precisa de fechamento:

```
<img src="foto.jpg">
```

Nesse caso, não usamos fechamento porque uma imagem não contém nada dentro e, portanto, não precisamos indicar que ela começa ou termina em determinado lugar. Você notou que, dentro da tag, além do elemento *img*, nós temos outras informações? Vamos ver sobre isso agora.

4.3 Atributos

O atributo serve para tornar o objetivo do elemento mais específico. Por exemplo, a seguir temos a tag do elemento *img*:

Sabemos que é uma imagem, mas não está sendo especificada qual.

Agora, veja o próximo exemplo:

<img src="perdido_em_Acapulco.jpg">

Nesse exemplo, temos o atributo *src* e o valor *perdido_em_Acapulco.jpg*, tornando a tag mais específica.

5 Estrutura básica

Existe uma estrutura básica que você SEMPRE deve colocar no seu código HTML. A finalidade é, basicamente, separar o conteúdo que ficará visível para o usuário daquela informação que será lida apenas pelos robôs, navegadores e outros softwares.

Para ficar mais fácil de explicar, quando vamos escrever uma carta para um cliente, separamos (virtualmente) nosso texto em três partes: cabeçalho, corpo do texto e rodapé.

- No cabeçalho, ficam informações como o logo da empresa.
- No corpo do texto, fica o texto de interesse do leitor.
- No rodapé, ficam informações como o endereço da empresa.

Em HTML, funciona mais ou menos da mesma forma. Temos o cabeçalho (representado pela tag <head>) e o corpo do texto (representado pela tag <body>).

Veja a estrutura básica de uma página HTML:

```
<!doctype html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8">
            <title></title>
      </head>
      <body>
      </body>
</html>
```

5.1 Doctype

<!doctype html>

O Doctype sempre deve ser a primeira linha antes do código HTML (não, não se trata de uma tag). Ele serve para passar uma instrução para os navegadores e outros dispositivos sobre qual a versão de marcação em que o código foi escrito. Vimos a evolução do HTML anteriormente, lembra-se?

Bem, quando eram usadas as versões anteriores, como o XHTML, era necessário informar o DTD (Document Type Definition) diretamente no código Doctype. Com o HTML5, essa referência passou a ficar a cargo do próprio navegador.

IMPORTANTE

Como os navegadores passaram a ficar responsáveis por informar o Doctype de todas as páginas, é recomendado que não sejam utilizados navegadores antigos, para que as páginas possam ser exibidas de acordo com os padrões de que o HTML5 precisa.

5.2 Elemento HTML

<html lang="pt-br">

A partir daqui, começa a marcação do código.

A primeira tag do código sempre será <html>. Note que temos o atributo *lang* (abreviação de *language*) e o valor *pt-br*, que indica que o idioma da página é o português do Brasil. Essa informação é útil principalmente para robôs dos mecanismos de busca.

Uma coisa que é importante você saber é que existem elementos pais e filhos. Os elementos pais sempre ficam acima, e os elementos filhos sempre ficam dentro e abaixo dos elementos pais. Tendo como exemplo a estrutura básica do HTML, o elemento *title* é filho do elemento *head*, e este é filho do elemento HTML.

5.2.1 Elemento *head*

<head>

O elemento *head* representa o cabeçalho da página. Nessa área do código, colocamos tudo aquilo que servirá para dispositivos e navegadores lerem.

Percebeu que usamos a palavra "área"? Com o passar do tempo, você vai desenvolvendo essa visão de que o código possui "áreas", pois, entre a abertura <head> e o fechamento </head>, podemos ter uma quantidade imensa de informação, e é por isso que desenvolvedores se referem a esse bloco de código como uma "área".

5.2.2 Elemento *meta*

<meta charset="UTF-8">

Lembra que a web foi criada para ser universal, acessada de qualquer lugar, por qualquer pessoa, em qualquer idioma? Pois bem, como cada país possui um conjunto de caracteres, criou-se uma tabela única, chamada de Unicode.

Veremos na prática por que o Unicode é importante. Vamos supor que você tenha um cliente cuja marca seja Código em Ação e ele lhe pede para arrumar seu site, que está apresentando o nome da empresa da seguinte forma:

Codigo em A��o

ou

Codigo em AÃ§Ã£o

Veremos como solucionar esse problema. Provavelmente, no desenvolvimento do site, não foi informada qual tabela de caracteres deve ser usada para apresentar caracteres acentuados. No Brasil, utilizamos o padrão de caracteres *iso-8859-1*, que é a tabela do padrão europeu ocidental, usada por países que utilizam acentuação, como Portugal, França, Espanha, etc.

Portanto, para resolver o problema, bastaria informar o *charset* conforme o código a seguir:

```
<html>
      <head>
           <meta charset="iso-8859-1">
```

Porém, se você tentar utilizar o navegador de um computador de outro país que não utiliza esse padrão de caracteres, pode acontecer de o navegador não conseguir exibir alguns caracteres na tela, por estes não existirem na sua tabela.

Para evitar esses problemas, o mais adequado é utilizar o *charset UTF-8* (ou seja, o Unicode) conforme mostrado a seguir:

```
<html>
      <head>
           <meta charset="utf-8">
```

IMPORTANTE

Meta tag não possui fechamento.

5.2.3 Elemento *title*

<title>

A finalidade da tag <title> é informar qual o título da página. Serve tanto para mostrar o título da página na guia do navegador quanto para mostrar no resultado da busca do Google.

```
<html>
      <head>
           <title>Título da página</title>
```

5.2.4 Elemento *body*

<body>

Esta é a parte visível da página. É onde fica toda a informação que realmente importa para o visitante do site, como o corpo do texto em uma carta. Quando se pensa em texto, imagem, formulário, vídeo, áudio, tudo vai dentro dessa parte.

6 Títulos

Você deve ter percebido que a maioria das pessoas que leem jornais ou sites de notícias tende a procurar a manchete primeiro, porque é o título mais importante do texto da notícia.

Quando você for criar uma página HTML, deve se lembrar dessa informação e pensar nessa hierarquia do texto. No HTML, os títulos são chamados de *heading* e possuem seis níveis:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

A forma como o navegador apresenta o resultado após a leitura do código pode ser observada na figura 3:

Figura 3 – Apresentação dos níveis dos títulos em HTML

Mais adiante, veremos que existem algumas convenções entre desenvolvedores, principalmente os que se preocupam em aplicar técnicas de SEO (otimização para mecanismos de busca), em que a tag H1 é usada para informar o assunto da página e a tag H2 passa a ser o segundo título mais importante.

IMPORTANTE

Você nunca deve utilizar uma tag de título por causa do tamanho no qual ela deixa o texto. Toda a parte de formatação, você deverá fazer usando código CSS, que é a linguagem de estilo, a qual estudaremos mais adiante.

7 Parágrafos

Como aprendemos anteriormente, a tag é usada para criar parágrafos. Se você digitar um texto livremente no código, sem usar a tag , o navegador vai ignorar qualquer quebra de linha que você tenha dado com a tecla *Enter* e colocar o texto em sequência. Por isso, você deve sempre iniciar um parágrafo com e indicar que o parágrafo terminou usando a tag .

Exemplo:

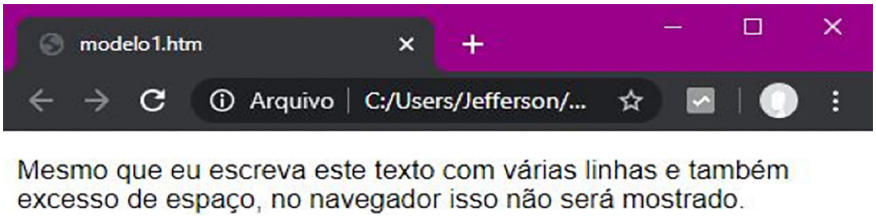
```
<p>Este é um exemplo de parágrafo.</p>
```

Ainda que você utilize a tag , se você colocar excesso de espaço ou se der *Enter* no texto, o navegador vai ignorar.

Exemplo:

```
<p>
Mesmo que eu
escreva este texto com
várias linhas e também
excesso de espaço,
no navegador isso não será mostrado.
</p>
```

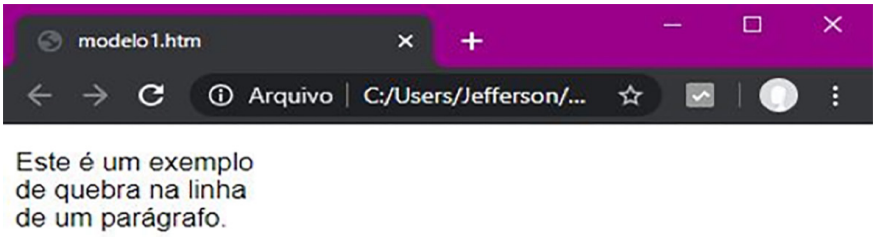
Figura 4 – Apresentação do texto usando a tag <p>



Caso precise, você pode efetuar uma quebra na linha do seu texto usando a tag
:

```
<p>Este é um exemplo<br>de quebra na linha<br>de um parágrafo.</p>
```

**Figura 5 – Apresentação do texto usando a tag
**



8 Imagens

Anteriormente, dissemos que a tag <img> não possui fechamento, diferentemente do parágrafo ou do título, em que é preciso informar onde começam e onde terminam. No entanto, essa tag possui um atributo obrigatório, que é o *src*.

O atributo *src* (que é abreviação de *source* e significa origem) serve para indicar o caminho no qual se encontra o arquivo de imagem. Veja um exemplo de um arquivo chamado "logo.jpg", que está armazenado no mesmo local que a página web:

<img src="logo.jpg">

Esse é um exemplo de que, no mesmo local da página web, existe um diretório chamado "imagens", e dentro dele tem um arquivo chamado "logo.jpg". Chamamos isso de caminho relativo, assim como no primeiro exemplo:

<img src="imagens/logo.jpg">

A seguir, temos um exemplo de como exibir uma imagem que está em outro endereço de internet. Chamamos isso de caminho absoluto.

<img src="https://www.sp.senac.br/image/65814.jpg">

O atributo *alt* não é obrigatório, mas é muito recomendado que você o utilize por dois motivos:

1. Quando ocorre algum problema no carregamento da página, o texto que você informar no atributo *alt* pode ser exibido no lugar da imagem, fazendo com que o leitor saiba do que se trata. Por isso, é importante que seja uma breve descrição da imagem, e não um texto qualquer.

2. Aplicativos leitores de tela podem ler a descrição inserida no atributo *alt*. Isso ajuda pessoas com deficiência visual a saberem o que contém a imagem exibida.

A seguir, temos um exemplo do uso do atributo *alt* para acessibilidade.

Figura 6 – Exemplo de texto para atributo *alt*


```
<img src="https://www.sp.senac.br/image/65814.jpg" alt="anúncio mostrando uma
jovem com cabelos cacheados ruivos, olhando com um leve sorriso, e à direita a
frase Sou Graduação Senac EAD">
```

NA PRÁTICA

O sistema operacional Windows 10 possui um leitor de tela chamado Narrador. Basta ir ao Menu Iniciar e procurar a pasta *Facilitador de Acesso*. Em outras versões, é a pasta *Acessibilidade*.

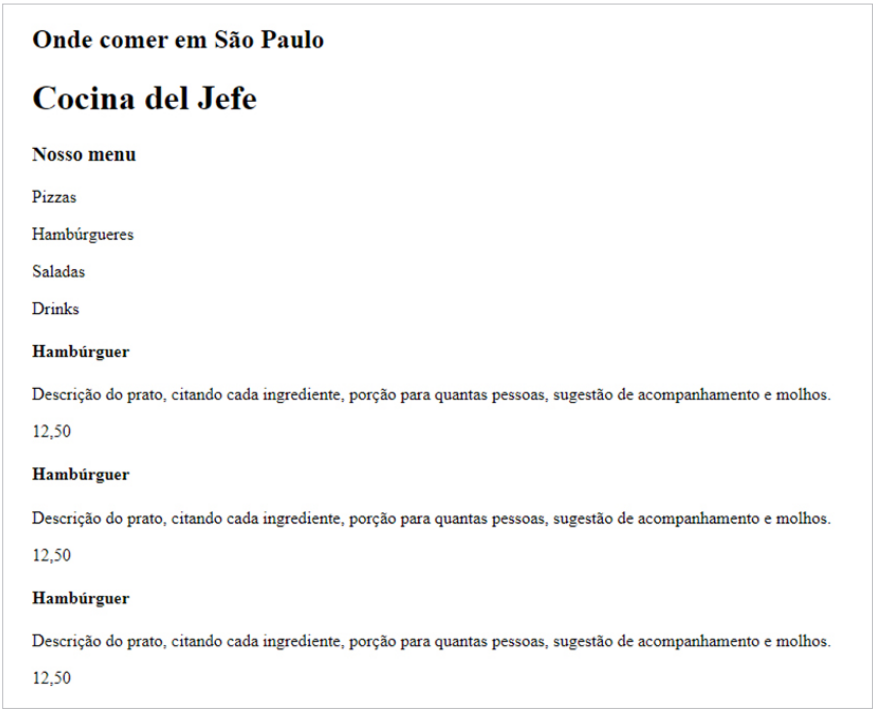
Utilize esse software para testar a acessibilidade das páginas que você criar nas atividades.

É importante que você saiba que alguns navegadores não conseguem ler alguns formatos de imagem. Procure utilizar os formatos mais comuns, tais como JPG, PNG, BMP e GIF.

9 Atividade

Agora, chegou a sua vez de colocar a mão na massa: utilizando as tags de título e parágrafo e a estrutura básica do HTML5, crie uma página com a mesma aparência do modelo proposto a seguir.

**Figura 5 – Apresentação do texto usando a tag
**



IMPORTANTE

Para desenvolver páginas HTML, você pode usar um editor de texto simples do seu computador ou baixar o Visual Studio Code, que é um software livre.

Caso você utilize um editor de texto, lembre-se: na hora de salvar seu trabalho, ele deve ficar com a extensão .htm ou .html. Isso é o que tornará seu trabalho uma página web, e não apenas um arquivo de texto.

Considerações finais

Neste capítulo, vimos como a web foi criada e também como chegamos ao conceito de hipertexto, vinculando uma informação a outra. Para isso, surgiu a linguagem HTML, idealizada para ser universal, acessível em qualquer dispositivo, independentemente de idioma ou cultura. Apesar disso, pessoas com deficiência visual ainda possuem dificuldade de navegar pela web, mesmo utilizando softwares de leitura de tela. Então, terminamos este capítulo propondo a você que reflita sobre alguns pontos: será que estamos criando uma web realmente mais acessível e universal? Será que estamos dando o devido valor ao atributo *alt*? Será que profissionais web estão preparados para serem fornecedores de soluções para essas pessoas? As universidades possuem monitores e professores treinados para orientar alunos com deficiência a utilizarem as ferramentas que existem no Windows, no Android e no iOS? As empresas estão preparadas para fornecer as ferramentas para essas pessoas trabalharem?

Trazemos todos esses questionamentos para que você perceba que está em suas mãos criar soluções web acessíveis para todos. Faça um trabalho bom no código, faça um layout harmonioso, e que seu trabalho sirva para ser inclusivo.

Referências

BOLT BERANEK AND NEWMAN. **A history of the ARPANET**: the first decade. Arlington: DARPA, 1981.

GIGANEWS. Usenet e newsgroups: o que é o Usenet? **Giganews**. [*s. d.*]. Disponível em: https://br.giganews.com/usenet.html. Acesso em: 7 jan. 2020

W3C. Sir Timothy Berners-Lee OM, KBE, FRS, FREng, FRSA: longer biography. **W3C**. 2020. Disponível em: https://www.w3.org/People/Berners-Lee/Longer.html. Acesso em: 7 jan. 2020.

W3C. Standards. **W3C**. 2019. Disponível em: https://www.w3.org/standards. Acesso em: 7 jan. 2020.

W3C. Web Architecture. **W3C**. 2015. Disponível em: https://www.w3.org/standards/webarch. Acesso em: 7 jan. 2020.