



APRENDIZAGEM EM FOCO

LINGUAGEM E PADRÕES WEB



APRESENTAÇÃO DA DISCIPLINA

Autoria: Anderson da Silva Marcolino

Leitura crítica: Gabriela Silveira

O acesso à Internet e seu uso como meio de obter informações para lazer, trabalho e educação, é realidade em todo o mundo. Contudo, para que isso ocorra, é necessário criar formas dos usuários usufruirem de tais benefícios.

Desse modo, entender como funciona a Internet e como são criados os websites e aplicações web, é essencial para que sejam produzidos conteúdos atrativos e dinâmicos, proporcionando uma boa experiência aos usuários.

Dessa maneira, uma das etapas iniciais para se tornar um desenvolvedor web é entender linguagens e padrões web que são utilizados atualmente. Em especial a base que permite a construção de páginas básicas e avançadas. Dentre as tecnologias que se destacam por essas características, temos a linguagem de marcação de hipertexto (HTML), as folhas de estilo em cascata (CSS) e a linguagem JavaScript (JS).

Esta disciplina busca apresentar essas tecnologias essenciais para a construção de conteúdos para Internet, cobrindo as marcações (*tags*) básicas e avançadas para criação de páginas em HTML; propriedades e utilização de folhas de estilo; linguagem de programação interpretada JavaScript. Por meio da utilização delas, será realizada a criação de projetos de páginas web com boas práticas, de acordo com os padrões do consórcio *World Wide Web*, que padroniza e disponibiliza documentação e especificações destas e de outras tecnologias.

Com os fundamentos e conhecimentos dessas tecnologias, você poderá ser capaz de criar websites e aplicações web e terá base necessária para se aprofundar no uso de *frameworks* web conhecidos.

Bons estudos!

INTRODUÇÃO

Olá, aluno (a)! A *Aprendizagem em Foco* visa destacar, de maneira direta e assertiva, os principais conceitos inerentes à temática abordada na disciplina. Além disso, também pretende provocar reflexões que estimulem a aplicação da teoria na prática profissional. Vem conosco!



TEMA 1

Dominando a linguagem de marcação de hipertexto – HTML

Autoria: Anderson da Silva Marcolino

Leitura crítica: Gabriela Silveira



DIRETO AO PONTO

Um website ou aplicação web utiliza recursos oriundos dos documentos criados com a linguagem de marcação de hipertexto (HTML). Os arquivos de extensão *.html* são definidos por meio de diversas *tags* ou, em português, marcações.

As *tags* do HTML formam elementos HTML. Algumas delas integram *tags* de fechamento e abertura. Por exemplo, *<p>* é a *tag* de abertura e *</p>* é a *tag* de fechamento para o elemento de parágrafo. Há também *tags* que são *self close*, ou seja, possuem apenas a *tag* de abertura e não necessitam de uma *tag* de fechamento, como é o caso da *tag* *
* para quebra de linha, ou ** para inserir uma imagem.

Tal como os documentos HTML, temos ainda a linguagem de marcação extensível, XML, que possibilita seu uso para a especificação de diversos documentos. Diferencia-se dos documentos em HTML por sempre apresentar *tags* de abertura e de fechamento. Exemplo: *<informacao>*, que seria a *tag* de abertura e *</informacao>*, que indica a *tag* de fechamento, devido à barra presente após o sinal de menor.

Já no caso dos arquivos XHTML, é a linguagem de marcação extensível de hipertexto. Faz parte da família de linguagens de marcação XML e integra especificações do XML com HTML. Em outras palavras, usa as *tags* do HTML, mas sempre considerando o par abertura e fechamento para criar um documento com arquivo de extensão *.xhtml*. É importante mencionar que essas diferenças são as mais perceptíveis e que, nas versões mais recentes do HTML, opta-se por *tags* que delimitem a abertura e fechamento, mesmo ainda existindo a possibilidade do uso de *tags self close*, segundo Laura (2019).

Assim, para definir um documento HTML, precisamos utilizar a especificação da linguagem de marcação. Por assim ser conhecida,

não podemos falar que programamos em HTML, já que não se trata de uma linguagem de programação. A especificação do HTML é mantida pela *World Wide Web Consortium*, que é a principal organização de padronização da *World Wide Web*, logo, fornece os melhores materiais para estudo, visto que, ao surgir uma nova versão do HTML, será em seus sites que tal especificação será divulgada primeiro.

A W3C (2020) define algumas categorias de elementos HTML. O Quadro 1 resume as principais *tags* necessárias para criar um website básico.

Quadro 1 – Elementos do HTML5

Tag	Descrição	Exemplo
<!DOCTYPE>	Define o tipo de documento.	<!DOCTYPE HTML 4>
<html>	Define um documento HTML. O atributo <i>lang</i> indica qual a linguagem a ser utilizada nos textos do documento, a fim de facilitar possíveis ferramentas de tradução dos navegadores.	<!DOCTYPE html> <html lang="pt-BR"> <head> <title>Título do Documento</title> </head> <body> <h1>Este é um Título</h1> <p>Um parágrafo qualquer.</p> </body> </html>
<head>	Seção do documento HTML que contém metadados e informações do documento. Não inclui conteúdos das páginas.	
<title>	Define o título do documento.	
<body>	Define o corpo do documento. É onde serão incluídas as <i>tags</i> de conteúdo da página.	
<h1> até <h6>	Define os títulos de um documento HTML, com tamanhos e formatações diferentes para cada número indicado na <i>tag</i> .	<h1></h1>
<p>	Define um parágrafo no documento HTML.	<p>Exemplo de parágrafo!</p>

 	Insere uma quebra de linha no documento. É uma <i>tag self close</i> .	
<hr>	Insere uma linha ou divisão de conteúdo, como uma quebra de seção. É uma <i>tag self close</i> .	<hr>
<!-- ... -->	Define um bloco de comentário no documento, não sendo exibido na página web.	<!--Exemplo de comentário no documento HTML-->
Links		
<a>	Define um hiperlink ou hiperligação. É um elemento que, ao ser clicado pelo usuário, leva a outra página ou conteúdo na Internet. É o termo que dá origem ao H na sigla HTML.	Visite W3Schools.com!
<link>	Define uma relação entre um documento HTML e um recurso externo, como uma folha em estilo cascata (arquivo de extensão .css). É uma <i>tag self close</i> .	<link rel="stylesheet" href="styles.css">
<nav>	Define links de navegação, ou seja, concentra mais de um elemento identificado pela <i>tag</i> <link>, formando um menu.	<nav> HTML CSS </nav>

Fonte: adaptado e traduzido de W3Schools (2020).

É possível notar que temos *tags* que utilizam um conjunto chave-valor, como, por exemplo, a *tag* de hiperlink <a>, com o atributo *href* que se refere à origem do hiperlink, sendo a chave o valor antes do igual e o valor, depois: Visite W3Schools.com!. Note que o atributo está inserido no corpo da *tag*. Há vários atributos que são utilizados para incluir e ampliar o uso dos documentos HTML como, por exemplo, o atributo *class* que cria um

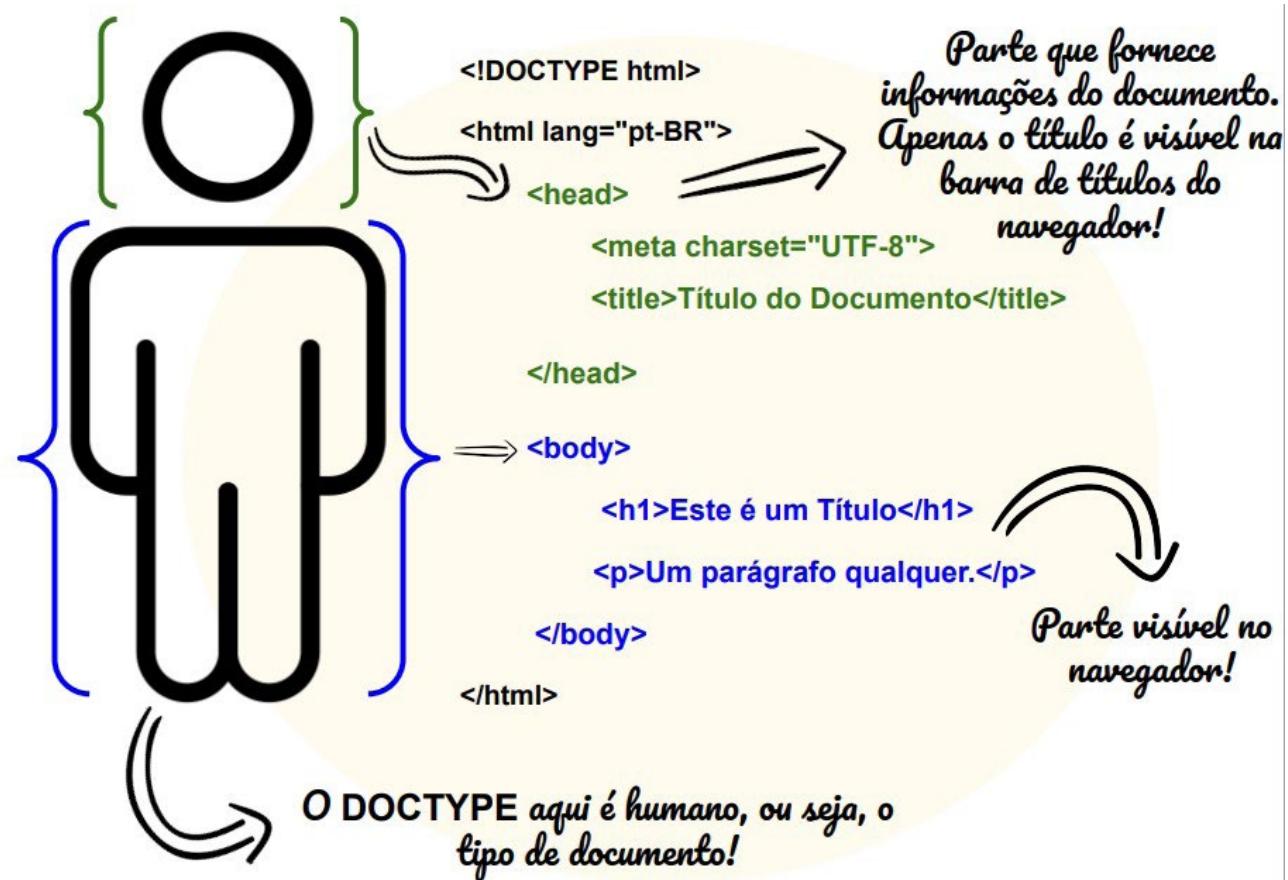
seletor para incluir um estilo de formatação ao documento, vinculado à tag `<style></style>`, que é inserida, por sua vez, no cabeçalho, entre o elemento `<head>` do documento HTML, ou que é inserido por meio de um arquivo de folha de estilos em cascata (CSS).

Os estilos são o que tornam as páginas HTML mais atraentes aos usuários, pois modifica significativamente sua aparência. Entretanto, é necessário aprender primeiro a construir a estrutura base em HTML, para depois aplicar folhas de estilos. Como prática para o aprendizado mais aprofundado sobre HTML, recomenda-se utilizar o exemplo dado a partir da tag `<html>` do Quadro 1 e incluir os demais elementos.

Para testar seu primeiro arquivo HTML, abra o bloco de notas, ou seu editor de textos predileto, e cole as marcações. Na sequência, escolha *Salvar como* e altere o tipo de arquivo para todos e, ao nomeá-lo, atribua o nome `index.html`. O termo `index` é utilizado para indicar ao navegador qual é a página inicial. Nada impede de salvar o arquivo com outro nome, mas lembre-se de que, em um projeto de website, será sempre necessário ter um arquivo `index.html` para indicar ao navegador qual é a página inicial daquele website.

O infográfico da Figura 1 apresenta as partes de um documento HTML e sua comparação com as partes do corpo humano. Com destaque para a cabeça, que corresponde à tag `<head>` e o corpo que corresponde à tag `<body>`. O elemento ou seção de cabeçalho integra informações do documento, sendo exibido apenas o título atribuído na tag `<title>` na barra de títulos do navegador. No HTML, a tag `<head>` pode ser omitida e seu conteúdo ser inserido entre a tag `<html>` e `<body>`. Enquanto todo o conteúdo do corpo, ou seja, do elemento `<body>`, é exibido no navegador, no momento em que o usuário acessa a página ou abre o arquivo `.html` no navegador.

Figura 1 – Infográfico da Anatomia de um Documento HTML



Fonte: elaborada pelo autor.

Referências bibliográficas

WORLD WIDE WEB CONSORTIUM (W3C). **HTML 5.2**. 2017. Disponível em <https://www.w3.org/TR/html52/>. Acesso em: 4 mar. 2021.

LAURA, A. G. **XHTML/CSS**: criação de páginas web. São Paulo: Senac, 2019.



PARA SABER MAIS

Você sabia que pode utilizar o bloco de notas para criar seu primeiro website?

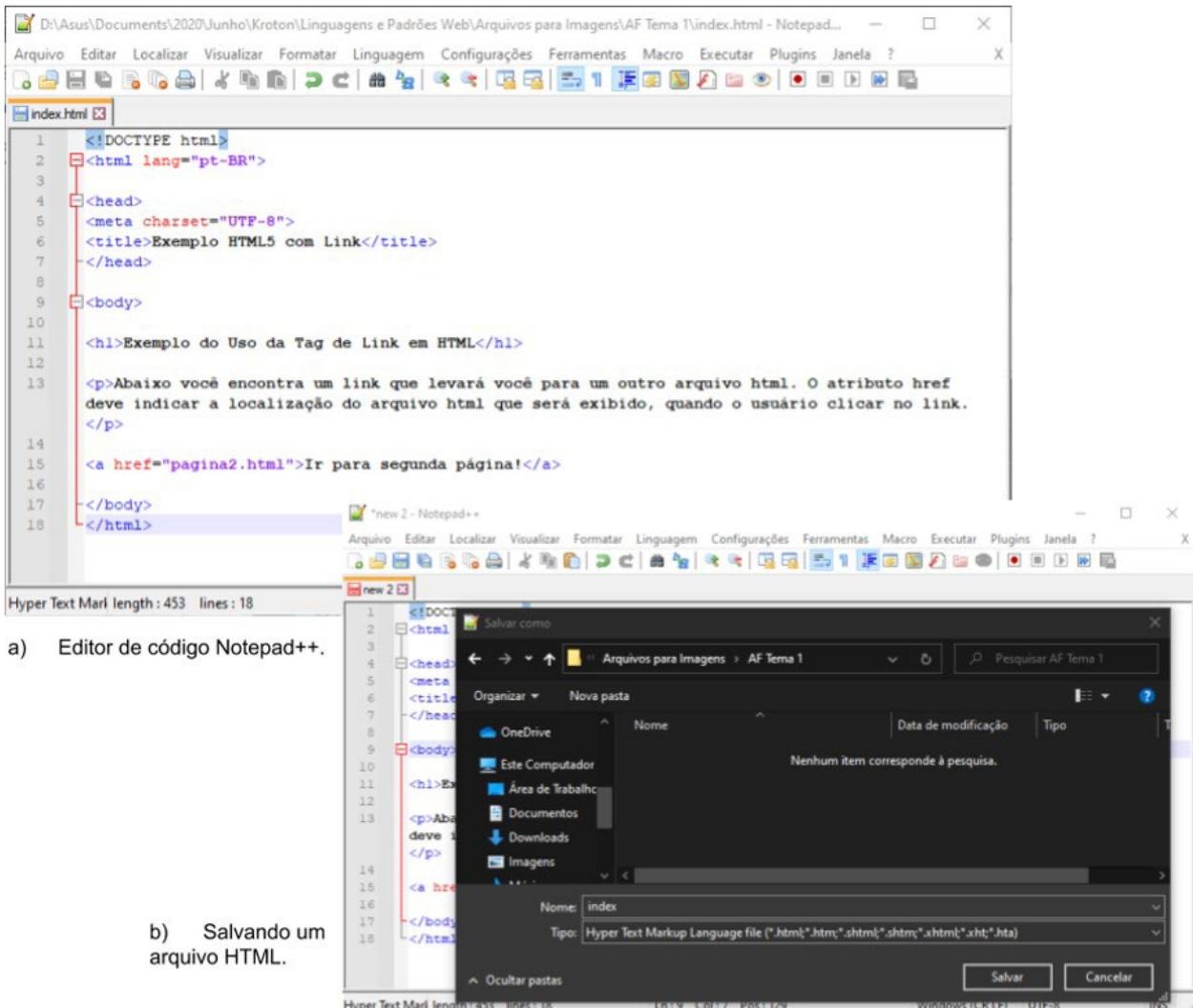
Sim! É possível utilizar o bloco de notas para criar um documento HTML, XML, XHTML ou outros. Contudo, alguns ambientes integrados de desenvolvimento (IDE) trazem mais vantagens que utilizar o bloco de notas (W3C, 2020). Existem ainda algumas outras opções gratuitas, mais robustas que o bloco de notas do Windows, a saber:

- **Notepad++**: editor que permite instalar *plug-ins*, ou seja, complementos. Permite a seleção da linguagem de programação ou marcação a ser utilizada.
- **Visual Studio Code**: o editor de código mais utilizado atualmente. Robusto e também permite a instalação de *plug-ins*. É muito utilizado no desenvolvimento de aplicações web e websites.

Vamos identificar como salvar um arquivo HTML, usando tais editores?

A Figura 2, em a), apresenta a tela do *Notepad++* e um projeto de uma página HTML. É recomendado testar o código, digitando-o, para aprimorar sua prática!

Figura 2 – Notepad++



a) Editor de código Notepad++.

b) Salvando um arquivo HTML.

Fonte: capturada de tela de *Notepad++*.

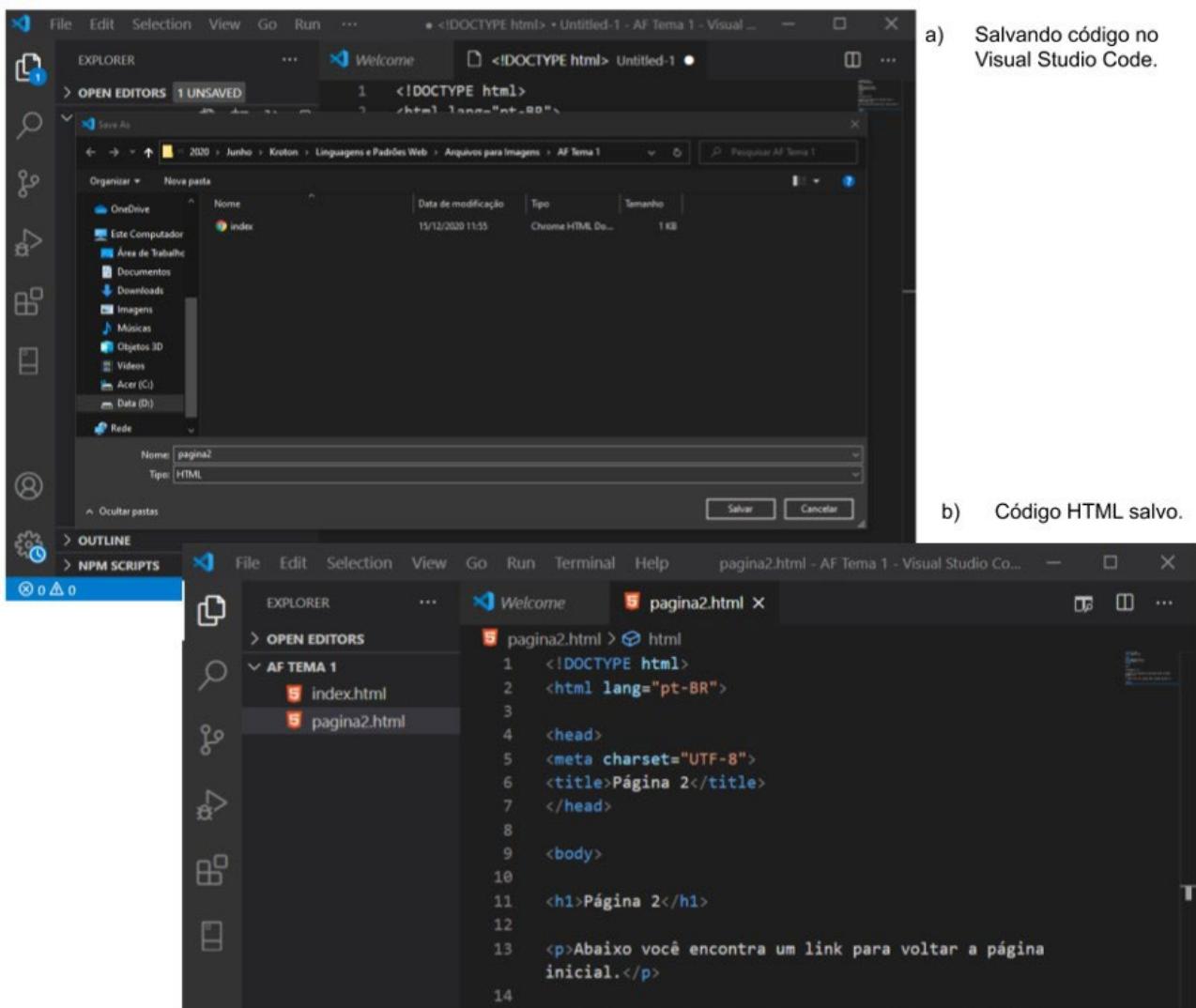
Para permitir realçar as palavras da linguagem de marcação, acesse, no menu principal do *Notepad++*, a opção *Linguagem*. Em seguida, *H* e, finalmente, *HTML*. Adicionalmente, você pode também configurar para o editor quebrar as linhas de textos mais extensas automaticamente, acessando, no menu principal, *Visualizar* e, em seguida, *Quebrar linhas automaticamente*.

A Figura 2, em b), mostra a opção para salvar o arquivo. Acesse, no menu principal, *Arquivo*. Em seguida, *Salvar como*. Garanta que, na entrada *Tipo*, esteja a opção *Hyper Text Markup Language file*. Ao

selecionar o tipo de linguagem, como descrito anteriormente, a opção de salvamento já indicará esse tipo de extensão.

A Figura 3 apresenta o *Visual Studio Code* com o código da *pagina2.html*, referenciada no elemento de link do código HTML, da Figura 2, e como salvar o arquivo utilizando tal editor.

Figura 3 – Visual Studio Code



Fonte: captura de tela de *Visual Studio Code*.

Diferentemente do *Notepad++*, o código em HTML, do *Visual Studio Code*, só destacará as palavras reservadas, como exibido na Figura 3, em b), quando o arquivo for salvo com tipo HTML.

Para salvar um arquivo no *Visual Studio Code*, basta acessar, no menu principal, a opção *Arquivo* e, na sequência, *Salvar como*, como pode ser visto na Figura 3, em a), alterando o tipo do arquivo para HTML e indicando o nome do arquivo. Em nosso exemplo, o arquivo salvo foi o *pagina2.html*, pois corresponde ao arquivo referenciado na Figura 2. Como prática de estudo, indica-se o uso das ferramentas para verificar qual melhor se adapta e, então, criar os projetos de ambos os arquivos html, testando-os! Bons estudos!

Referências bibliográficas

WORLD WIDE WEB CONSORTIUM (W3C). **HTML 5.2. 2017.** Disponível em <https://www.w3.org/TR/html52/>. Acesso em: 4 mar. 2021.

TEORIA EM PRÁTICA

Um website pode apresentar diferentes elementos e tornar-se motivo de impulsionamento de negócios, ou não. Tudo dependerá do quanto atrativo for a página criada. Considerando o uso somente do HTML, sabemos que as páginas ficam relativamente simples. Nesta perspectiva, crie um site que utilize, com qualidade, os diferentes elementos HTML existentes. Essa página deverá apresentar uma lista de especificações de um produto único, que uma empresa busca vender por meio do website, além de um formulário para envio de mensagens (nome, e-mail e mensagem). Utilize sua criatividade para criar tal página, sem qualquer estilização.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.



LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Este livro ensina a utilizar como base os arquivos XHTML, apresentando uma comparação entre tal linguagem de marcação com HTML para a criação de páginas web. Indo além da utilização de XHTML, o livro ensina a utilização de folhas de estilos.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra, no parceiro Biblioteca Senac.

LAURA, A. G. **XHTML/CSS**: criação de páginas web. São Paulo: Senac, 2019.

Indicação 2

Este livro apresenta uma visão geral da Internet e das tecnologias web. Mostra a criação de tais tecnologias e apresenta tendências que já estão em exploração, hoje, no contexto de desenvolvimento nesta área. Além disso, quanto às tendências ainda pouco estudadas, mostra que podem se tornar uma oportunidade para quem está ingressando agora como desenvolvedor web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra, no parceiro Biblioteca Senac.

FERRAZ, R. **Tendências da web. Série Universitária**. São Paulo: Senac, 2018.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. O desenvolvimento de páginas estáticas adota HTML como linguagem de marcação essencial para a criação de tal conteúdo. Considerando a existência de outras linguagens de marcação, em especial XML e XHTML, assinale a alternativa correta:
 - a. A linguagem de marcação de hipertexto, HTML, é menos restrita que XHTML e mais especializada que XML, e tem como padrão de extensão de seus arquivos a extensão *.html*.
 - b. A linguagem de marcação extensível, XML, é uma linguagem menos restrita que XHTML, permitindo a representação de dados ou criação de outras linguagens de marcação e tem como extensão os arquivos *.xms*.
 - c. A linguagem de marcação HTML não é uma linguagem de programação e é menos restritiva que XML e XHTML, criando documentos com a extensão *.lmth*.
 - d. A linguagem de marcação XHTML une elementos de HTML com XML, criando uma versão mais restritiva para o desenvolvimento web, gerando arquivos com extensão *.XMLH*.
 - e. A linguagem XML é utilizada para a criação de páginas web mais flexíveis e gera arquivos com extensão *.XML*.

2. A linguagem HTML é uma linguagem composta por marcações ou as chamadas *tags*. Tais *tags*, por sua vez, são estendidas semanticamente por meio do uso de atributos do tipo chave-valor. Considerando as *tags* e seus atributos, assinale a alternativa correta:
- a. A tag `<!DOCTYPE>` é uma *tag* que fornece informações ao navegador, especificando qual a versão do navegador que deve ser utilizado.
 - b. O atributo *lang* e seu valor “*pt-BR*” (*lang=“pt-BR”*) informa para o navegador a linguagem que o mesmo deverá traduzir tal documento.
 - c. O atributo *href* indica a referência do hiperlink definido pela tag `<k>` em um documento HTML.
 - d. O atributo *href* na tag `<link>` indica qual arquivo o link clicado levará o usuário.
 - e. O atributo *class* cria um seletor para que o navegador insira uma imagem de fundo na página web.



GABARITO

Questão 1 – Resposta A

Resolução: A linguagem HTML gera arquivos com extensão *.HTML*, enquanto a XML gera arquivos com extensão *.XML* e *XHTML* com a extensão *.XHTML*. HTML é especializada e mais restritiva que XML, que, por sua vez, é utilizada para representar, na maioria das vezes, dados e informações ou permitir a especificação de outras marcações e linguagens. Como é o caso do XHTML, que usa as *tags* do HTML e as restrições do XML para criar uma linguagem de marcação web mais restrita e, consequentemente, mais padronizada.

Questão 2 – Resposta C

Resolução: A tag `<!DOCTYPE>` indica qual é a versão da linguagem de marcação utilizada. Por exemplo, se indicado HTML, o navegador entenderá que está sendo utilizada a versão 5 do HTML. Já o atributo `lang="pt-BR"` indica qual a linguagem adotada para escrever aquele website, permitindo que o navegador oferte ferramentas de tradução, caso algum usuário de fora do Brasil queira traduzir este site para outra linguagem como, por exemplo, o inglês. O atributo `href` tem diferentes usos, dependendo da tag. No caso de seu uso com a tag `<a>` e não `<k>`, indica o arquivo `.html` que será aberto quando o usuário clicar no link. Já na tag `<link>`, indica o arquivo de referência que será inserido por meio daquele link no documento HTML que utilizar esta tag. Finalmente, o atributo `class` permite a indicação de um arquivo de estilo ou a aplicação de um estilo, que pode inserir não só uma imagem de fundo da página, mas também outros elementos gráficos de modo a enriquecer a interface da página web.



TEMA 2

Aprimoramento das páginas Web: as folhas de estilo em Cascata

Autoria: Anderson da Silva Marcolino

Leitura crítica: Gabriela Silveira



DIRETO AO PONTO

As folhas de estilo em cascata, do termo em inglês, *cascading style sheets* (CSS) são arquivos, geralmente, com extensão .css, que integram regras que mudam a aparência de um ou mais elementos de um documento HTML. Tal como a linguagem de marcação de hipertexto (HTML), que não representa uma linguagem de programação, CSS é uma linguagem de folha de estilos. Portanto, não é correto dizer que se programa em CSS.

As regras incluídas nos documentos HTML são definidas e aplicadas em elementos do HTML por meio de seletores. Há diversos tipos de seletores, sendo os principais:

- Seletor de elemento.
- Seletor de identificador.
- Seletor por classe.
- Seletor por elemento e classe.
- Seletor universal.
- Seletor de grupo.

Após definir o tipo de seletor a ser utilizado e adaptar o HTML e suas tags para que tais seletores possam aplicar um estilo ao elemento da página, deve-se criar a declaração da regra em CSS. Um bloco de declaração é definido pela abertura e fechamento de chaves ("{ ... }"). Neste bloco, são inseridas propriedades e seus valores. São tais propriedades que resultarão, de fato, na modificação da página HTML final.

Uma página pode receber as regras do CSS de diferentes formas: no cabeçalho da página; *inline*, ou seja, na própria linha; ou por meio de um ou mais arquivos .css, inseridos por meio da tag *link* no cabeçalho de um documento HTML. A utilização mais indicada, inclusive pela W3C e padrão HTML5, é esta última, pois a criação de arquivos separados permite sua reutilização, além de melhor organização para manutenção.

As duas regras do CSS, a seguir, especificam dois blocos aplicados para dois seletores. O primeiro deles é aplicado para a tag *body*, ou seja, a partir do momento em que o arquivo CSS for inserido no documento HTML, a tag *body* será estilizada, considerando as propriedades inseridas no corpo da definição da regra. No segundo exemplo, é necessário a inclusão do atributo *class="titulo"*, para que esta seja aplicada. Por exemplo: *<p class="titulo">Texto do parágrafo.</p>*, cujo o código que aplica a regra do seletor *titulo* para o elemento de parágrafo:

```
body {  
background-color: lightgrey;  
color: blue;  
}  
  
.titulo {  
background-color: black;  
color: white;  
}
```

As propriedades alteradas nas regras estão relacionadas à cor do plano de fundo e cor do texto. Todos os elementos que estiverem inseridos no elemento *body* do HTML terão como plano de fundo a

cor cinza claro (*lightgrey*) e o texto será exibido na cor azul (*blue*). As *tags* e elementos que possuírem o atributo *class="titulo"* terão, por sua vez, o plano de fundo preto (*black*) e a cor branca (*white*).

Uma vez declaradas e salvas as regras em um arquivo *.css*, que pode ser criado, inclusive, pelo bloco de notas, utilizando a opção *Salvar como* e indicando o tipo *Todos os arquivos*, bem como um nome qualquer e a extensão *.css*, que pode ser integrada à uma página HTML.

A integração no documento HTML ocorre entre as *tags* de abertura e fechamento do cabeçalho ou em tal seção sem tais *tags* (quando utilizado HTML5 tal *tag* é opcional), como é indicado nesse excerto de código: *<link href="nome_qualquer.css" rel="stylesheet">*.

Não há necessidade de qualquer compilação ou outra modificação, além das já mencionadas indicações dos seletores e inclusão do arquivo *.css*. A única observação é que deve ser indicado, no atributo *href* da *tag link*, o caminho e nome exato do arquivo *.css*, para que este possa ser integrado e aplicado ao website.

Finalmente, é importante destacar que uma página pode ter um ou mais arquivos *.css*. Contudo, o aumento expressivo sem um planejamento prévio bem definido pode gerar confusão e resultar em maiores esforços para a equipe de desenvolvimento, que realizará possíveis manutenções nas páginas que usarem tais arquivos. Assim, a melhor maneira de se avaliar quantos arquivos devem ser criados, é projetar! Geralmente, um arquivo que centralize as regras gerais, comuns para todas as páginas HTML, e um arquivo específico para cada uma das páginas HTML é a boa prática adotada na indústria de desenvolvimento.

A Figura 1 apresenta três exemplos de páginas HTML, sendo que dois deles integram folhas de estilo em cascata. Na Figura 1, em a), utilizou-se HTML puro. Na Figura 1, em b), as folhas de estilo são

aplicadas na própria página, por meio do elemento *style* inserido no cabeçalho do documento HTML. Já na Figura 1, em c), a folha de estilos é inserida por meio da tag *link* no cabeçalho (*head*) do documento HTML.

Figura 1 – Uma página HTML com a) sem CSS, b) com CSS no próprio arquivo HTM e c) com arquivo .css

The diagram illustrates three methods for including CSS in an HTML page:

- a) HTML "Puro"**: Shows a plain HTML file with no CSS. The title is "Meu Primeiro Exemplo com CSS". The content is "Este é um parágrafo.".
- b) HTML e folhas de estilos na página.**: Shows an HTML file containing inline CSS within the *style* tag in the *head*. The title is "Meu Primeiro Exemplo com CSS". The content is "Este é um parágrafo.".
- c) HTML e folhas de estilos em arquivo .css.**: Shows an HTML file linking to an external CSS file named "exemplol.css" using the *link* tag in the *head*. The title is "Meu Primeiro Exemplo com CSS". The content is "Este é um parágrafo.".

Arrows point from each method's description to its corresponding screenshot. The screenshots show the code in a text editor and the resulting browser output. A separate screenshot shows the external CSS file "exemplol.css" in a Notepad window.

Code Examples:

- a) HTML "Puro"**: Plain HTML with no CSS.
- b) HTML e folhas de estilos na página.**: HTML with inline CSS in the *style* tag.
- c) HTML e folhas de estilos em arquivo .css.**: HTML linking to an external CSS file.

External CSS Content (exemplol.css):

```
body { background-color: lightblue; }
h1 { color: white; text-align: center; }
p { font-family: verdana; font-size: 20px; }
```

Fonte: elaborada pelo autor.

Referências bibliográficas

W3SCHOOLS. **CSS Reference**. 2020. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 4 mar. 2021.

LAURA, A. G. **XHTML/CSS: criação de páginas web**. São Paulo: Senac, 2019.



PARA SABER MAIS

Você sabia que o *box model*, ou modelo de caixa, na tradução para português, é amplamente utilizado na criação de páginas e aplicações web?

Ao utilizarmos folhas de estilos, o posicionamento, bordas, margens, espaçamento e o conteúdo em si, influenciam e impactam os elementos a serem apresentados em um website. O modelo de caixa torna a compreensão de posicionamento e a distribuição dos elementos mais simples. Analisemos a regra de CSS abaixo:

```
.classe {  
    width: 50px;  
    height: 50px;  
    border: 2px solid gray;  
    padding: 10px 20px;  
}
```

Ao analisarmos um elemento que receba tal estilo, temos um conteúdo de 50px por 50px. Para saber a altura e a largura total de tal elemento, devemos somar todos os elementos, considerando, ainda, que as bordas são incluídas em dois lados. Desse modo, temos de largura a seguinte soma: = largura do conteúdo (50px)

+ espaço à esquerda (20px) + espaço à direita (20px) + borda da esquerda (2px) + borda da direita (2px) = 94px. Já a altura será: = altura do conteúdo (50px) + espaço do topo (10px) + espaço do rodapé (10px) + borda do topo (2px) + borda do rodapé (2px) = 74px de altura.

Isso mostra que podemos cair em uma falácia ao não considerar as bordas, espaços e margens de um elemento. Para resolver este problema, podemos usar uma propriedade específica: a *box-sizing*.

Com a propriedade *box-sizing*, é possível criar um elemento com base no tamanho de um outro elemento. Isso pode ser visto na Figura 2.

Figura 2 – Model Box e *box-sizing*

The diagram illustrates the difference between the `border-box` and `padding-box` box-sizing models. On the left, a code snippet shows two CSS rules: `.elemento1` and `.elemento2`. `.elemento1` has a width of 250px and height of 50px. `.elemento2` has a width of 250px, height of 50px, and padding of 0 50px. An arrow points from the code to a visual representation. The top part shows a grey box labeled `.elemento1` with a width of 250px. The bottom part shows a larger grey box labeled `.elemento2` with a total width of 250px, divided into a 50px padding area on the left and a 200px content area (the original 250px width minus the 50px padding). A curved arrow indicates the padding area of `.elemento2`.

```
1. .elemento1,
2. .elemento2 { box-sizing: border-box; }
3.
4. .elemento1 {
5.   background: #ddd;
6.   width: 250px;
7.   height: 50px;
8. }
9.
10. .elemento2 {
11.   background: #ddd;
12.   width: 250px;
13.   height: 50px;
14.   padding: 0 50px;
15. }
```

Fonte: adaptado de BOX (2020).

O resultado do uso da propriedade *box-sizing*, incluído na linha 2, da Figura 2, resulta na alteração do tamanho do “elemento2”, que ficará com tamanho máximo de 250px, ainda que some os espaços em brando (padding) adicionados na regra com seletor “elemento2”.

Finalmente, é importante mencionar que o CSS3 tem versões de algumas de suas propriedades específicas para cada navegador, assim recomenda-se utilizar todas as regras do trecho a seguir, para garantir seu funcionamento nos navegadores mais utilizados:

```
-webkit-box-sizing: border-box;  
-moz-box-sizing: border-box;  
-ms-box-sizing: border-box;  
box-sizing: border-box;
```

Como prática de estudo, indica-se a utilização dessas propriedades em diversos projetos de websites! Bons estudos!

Referências bibliográficas

BOX MODEL. **Tableless**. 2020. Disponível em: <https://tableless.github.io/iniciantes/manual/css/box-model.html>. Acesso em: 4 mar. 2021.

TEORIA EM PRÁTICA

Os websites se tornam atrativos e costumam seguir receitas de sucesso em sua criação: cabeçalho, menu, corpo e rodapé. Considerando essa estrutura, imagine-se no cenário em que um colega dentista solicitou a criação de um website para seu consultório. Esse website deverá ter uma página principal com algumas imagens e textos sobre os tratamentos realizados e uma segunda página de contato. Usando, principalmente, o modelo de caixas do CSS, crie uma página HTML que integre um arquivo CSS principal e um arquivo CSS específico para as propriedades de uma segunda página, que mostrará informações de contato. Utilize sua criatividade para criar tal página, utilizando CSS.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.



LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

O livro ensina a utilização de folhas de estilos com documentos XHTML, apresentando as propriedades mais utilizadas no contexto de CSS.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra, no parceiro Biblioteca Senac.

LAURA, A. G. **XHTML/CSS**: criação de páginas web. São Paulo: Senac, 2019.

Indicação 2

O livro apresenta indicações de tendências web, que podem ser adotadas na criação de websites e aplicações web com HTML e CSS.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra, no parceiro Biblioteca Senac.

FERRAZ, R. **Tendências da web. Série Universitária**. São Paulo: Senac, 2019.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática* e *Leitura Fundamental*, presentes neste Aprendizagem em Foco.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. A inclusão de folhas de estilo em cascata em arquivos HTML ocorre para modificar e estilizar os diferentes elementos dessa linguagem de marcação. Considerando os principais tipos de seletores, assinale a alternativa correta:
 - a. São exemplos de seletores para aplicação de regras do CSS, o seletor de elemento, seletor de classe e seletor de atributos.
 - b. São exemplos de seletores para aplicação de regras do CSS, o seletor de elemento, seletor de classe e seletor de universal, sendo que somente o seletor universal precisa de modificações específicas no HTML.
 - c. São exemplos de seletores para aplicação de regras do CSS, o seletor de elemento, seletor de classe e seletor universal, sendo que somente o seletor de classe necessita de modificações no HTML.
 - d. Para que um seletor possa aplicar uma regra de CSS em um documento HTML, não é necessária qualquer modificação. Basta colocar os arquivos *.html* e *.css* em um mesmo diretório.
 - e. O seletor de grupo e universal são os mesmos, possuindo apenas nomes diferentes.

2. Uma regra do CSS é composta por alguns elementos sintáticos para compor os estilos que serão aplicados em um documento HTML. Considerando tais elementos das regras do CSS, assinale a alternativa correta:
- a. Um bloco de declaração é definido pela abertura e fechamento de chaves (“{ ... }”) e determina o início e fim de uma propriedade do CSS.
 - b. Uma regra de CSS é definida pelo conjunto de seletor e valor de propriedade.
 - c. A propriedade é a regra que será aplicada em conformidade com o valor que recebe, também chamado de valor da propriedade.
 - d. Uma regra CSS pode ser composta por uma ou mais propriedades, desde que estejam em blocos de definições diferentes.
 - e. Cada um dos pares propriedade-propriedade valor não devem ser separados por vírgula “,” quando estiverem no mesmo bloco.



GABARITO

Questão 1 – Resposta C

Resolução: O seletor de classe e de identificador são os que necessitam de modificação nos elementos HTML, por meio da indicação de um atributo de nome *class* e *id*, respectivamente. O seletor universal e de grupo são diferentes. O primeiro aplicará as regras do CSS para todos os elementos do documento HTML, enquanto o de grupo só aplicará as regras para os elementos indicados no seletor. Não existe um seletor de atributos. Para que um arquivo .css seja aplicado em um arquivo .html, deve-se importar o arquivo .css no cabeçalho da página HTML.

Questão 2 – Resposta C

Resolução: Uma regra CSS é composta pelo seletor seguido pelo corpo da declaração, delimitado pela abertura e fechamento de chaves (“{ ... }”), as propriedades (que podem ser mais de uma) e seus respectivos valores de propriedades. Cada um dos pares propriedade-propriedade valor são separadas por vírgula “,” e resultarão em uma modificação do estilo de um determinado elemento HTML.



TEMA 3

Linguagem JavaScript: do básico ao avançado

Autoria: Anderson da Silva Marcolino

Leitura crítica: Gabriela Silveira



DIRETO AO PONTO

A linguagem de programação ECMAScript, ou comumente conhecida como JavaScript, é uma linguagem que enriquece as possibilidades das páginas web e de seus conteúdos, trazendo comportamentos dinâmicos. Essa linguagem foi criada nos anos 1990 e somente em 1997 tornou-se um padrão ECMA. Após esta padronização, foi renomeada para ECMAScript. Contudo, como havia sido conhecida como JavaScript, essa denominação permanece até hoje, segundo Silva (2020).

JavaScript é uma linguagem interpretada, ou seja, necessita de um programa chamado interpretador, que lê o código e depois de sua interpretação (linha a linha e com análise léxica, sintática e semântica) e a executa por demanda. Esse interpretador é parte integrante dos navegadores. Contudo, como há navegadores com mecanismos e núcleos desenvolvidos com tecnologias diferentes, pode ocorrer incompatibilidade no suporte a determinadas funcionalidades do JavaScript, principalmente, as correspondentes as versões mais recentes (W3C, 2020).

Logo, além de se aprender a sintaxe da linguagem, a compatibilidade de suas funcionalidades deve ser verificada. Isso pode ser feito com pesquisas em mecanismos de busca sobre a compatibilidade de determinados códigos, ou a consulta direta em matérias da W3C.

A Figura 1 exibe uma página HTML com integração de um script programado com JavaScript.

Figura 1 – Função para encontrar número máximo

The screenshot shows a browser window with two panes. The left pane displays the source code of a HTML file. The right pane shows the resulting page with the output of the script.

```
<!DOCTYPE html>
<html>
<body>

<p>Encontre o maior número.</p>
<p id="conteudo"></p>

<script>
function findMax() {
    var i;
    var max = -Infinity;
    for(i = 0; i < arguments.length; i++) {
        if (arguments[i] > max) {
            max = arguments[i];
        }
    }
    return max;
}
document.getElementById("conteudo").innerHTML = findMax(4, 5, 6);
</script>

</body>
</html>
```

Encontre o maior número.
6

Fonte: elaborada pelo autor.

O primeiro ponto a ser identificado é como um código em JavaScript pode ser integrado à uma página HTML. Isso pode ocorrer por meio do elemento **script** inserido no corpo de uma página HTML, em seu cabeçalho ou por meio da integração, utilizando o elemento link do HTML.

Na Figura 1, o código é inserido diretamente por meio do elemento **script**. Apesar de ser uma prática comum, o ideal é que se crie um arquivo com extensão *.js* e este seja integrado ao documento HTML, visto que essa prática permite e facilita o reuso.

O segundo ponto a ser observado é como tal código em JavaScript é chamado pelo HTML. Isso ocorre pelo meio da função acessada no documento, apresentado na última linha do elemento **script**: `document.getElementById("conteudo").innerHTML`. Note que o valor do atributo *id* do elemento parágrafo (*p*) é o mesmo do passado como parâmetro para a função de busca do elemento por *id*. Tal integração pode ocorrer também por eventos HTML, que nada mais são do que atributos especiais, como o *onclick* de um botão ou o *onload* de uma página. Sem a integração do documento HTML com o código JavaScript,

a interpretação do código e sua execução podem não ocorrer no momento esperado, resultando em elementos e conteúdos não esperados nas páginas em que o código JavaScript integra.

Em um terceiro ponto, podemos analisar a sintaxe do código em JavaScript. Assim como as demais linguagens de programação, a linguagem apresenta estruturas condicionais, laços de repetição, como o *for*, presente na Figura 1, tipos primitivos de variáveis (cadeia de caracteres, números, booleanos) e tipos especiais como objetos e vetores.

No exemplo da Figura 1, temos uma variável *i* que recebe o valor zero no código da condição avaliada no *for*. É nesse momento que esta variável se torna do tipo primitivo número. Já a variável *max* recebe o valor *-Infinity*, ou seja, o valor máximo que uma variável do tipo número poderá atingir (64 bits). Isso ocorre porque o laço de repetição *for* será executado até que *max* seja o número de maior valor. Essa troca, para que *max* se torne o maior valor, ocorre no corpo da estrutura condicional *if*, que será verdadeira enquanto o valor do vetor *arguments*, que recebe os parâmetros da chamada da função *findMax()*, for maior que *max*, fazendo a substituição do mesmo pelo novo número disposto no vetor na posição do índice *i*, que é incrementado pelo *for*. Essa troca ocorre até que o *for* identifique que o *i* é maior que a quantidade de números do vetor (*arguments.length*), saindo da estrutura de repetição e retornando o valor de *max (return max)*.

O valor retornado é, então, inserido dentro do conteúdo do elemento de parágrafo, por meio do *id* identificado com o nome de *conteudo*.

Assim, podemos ter uma visão geral de como é o comportamento de JavaScript ao ser integrado a um documento HTML e como pode modificar o modelo de documento de objeto (DOM) que é, de fato, a renderização realizada pelos navegadores.

Referências bibliográficas

W3SCHOOLS. **JavaScript and HTML DOM Reference.** 2020. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 4 mar. 2020.

SILVA, M. S. **JavaScript-Guia do Programador:** guia completo das funcionalidades de linguagem JavaScript. São Paulo: Novatec, 2020.



PARA SABER MAIS

Você sabia que há métodos específicos para se trabalhar com vetores em JavaScript?

Um vetor ou *array* em JavaScript são códigos utilizados para armazenar múltiplos valores em uma única variável. Exemplo: var carros = [“Saab”, “Volvo”, “BMW”];. Assim, um vetor é uma variável especial, capaz de armazenar mais de um valor e que para acessar cada valor é necessário utilizar um índice (como visto na Figura 1).

No exemplo do parágrafo anterior, cria-se um vetor chamado *carros* que recebe três valores. É possível criar um vetor também por meio da palavra ***new***, ou seja, instanciando um objeto do tipo ***array***. Entretanto, assim como um objeto do tipo ***string***, o tempo de execução é maior e recomenda-se a declaração, como vista anteriormente.

Para acessar um valor de um vetor, pode-se utilizar um código como: `document.getElementById(“demo”).innerHTML = carros[0];`. Nesse exemplo, o valor retornado será “*Saab*”, que corresponde à primeira ***string*** inserida no vetor.

Para alterar um valor, basta utilizar o vetor e seu índice. Por exemplo: `carros[0] = “Opel”;`. Esse código altera o valor do nome do carro que estava na posição zero.

Para acessar um primeiro elemento de um vetor, o índice deve ser zero. Já o último, pode-se utilizar `carros[carros.length-1]`.

Para imprimir um vetor, podemos utilizar a estrutura condicional **for** (vide Figura 1) ou um método específico. É aqui que surge um dos diversos métodos que podem facilitar o trabalho com os vetores de JavaScript: o método ou função **forEach()**. Por exemplo, para imprimir os carros do vetor anterior, basta utilizar algo como apresenta do código a seguir:

```
var carros, texto;  
  
var carros = ["Saab", "Volvo", "BMW"];  
  
texto = "<ul>";  
  
carros.forEach(minhaFuncao);  
  
texto += "</ul>";  
  
function minhaFuncao(valor) {  
  
    texto += "<li>" + valor + "</li>";  
  
}  

```

Outro método muito utilizado é o **push()**. Esse método permite incluir um novo elemento ao final de um vetor. No exemplo do código anterior, podemos adicionar um novo carro utilizando: `carros.push("Gol")`.

Para remover um último elemento de um vetor, pode-se utilizar o método **pop()**. Já o método **shift()** remove o primeiro elemento de um vetor e troca todos os demais para uma posição a menos. Já o método **unshift()**, inclui um elemento na primeira posição do vetor

e troca os demais valores uma posição à frente. Finalmente, para inserir um elemento em uma determinada posição de um vetor, pode-se utilizar o método ***splice()***. Por exemplo: var frutas = [“Banana”, “Laranja”, “Maçã”, “Manga”];

```
frutas.splice(2, 0, “Limão”, “Kiwi”);.
```

O primeiro parâmetro (número 2) passado no método ***splice()*** define a posição onde os novos elementos devem ser inseridos; o segundo parâmetro (zero) define quantos elementos devem ser removidos, no caso, nenhum elemento será removido. O resto dos parâmetros definem os novos elementos que serão inseridos. O método ***splice*** pode ser utilizado para remover elementos. Por exemplo, utilizando frutas.splice(0,1); o primeiro elemento do vetor é removido.

Como método contrário ao ***splice***, temos o método ***concat()*** que concatena dois ou mais vetores: var arr1 = [“Cecilie”, “Lone”];

```
var arr2 = [“Emil”, “Tobias”, “Linus”];
```

```
var arr3 = [“Robin”, “Morgan”];
```

```
var meusFilhos = arr1.concat(arr2, arr3);
```

No exemplo anterior, três vetores são concatenados, resultando em um vetor de sete elementos, indo do zero ao seis.

Caso você queira criar um vetor sem valores e adicionar seus valores posteriormente, por meio do ***push()***, pode-se declarar um vetor vazio: var veículos = [].

Pode-se utilizar também o ***instanceof*** para reconhecer se uma variável é ou não um vetor: carros instanceof Array; // retornará tru-verdadeiro.

Como prática de estudo, indica-se a utilização desses métodos em diversos projetos de websites! Bons estudos!

Referências bibliográficas

W3SCHOOLS. **JavaScript and HTML DOM Reference.** 2020. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 4 mar. 2021.

TEORIA EM PRÁTICA

JavaScript é uma linguagem de programação, o que permite enriquecer consideravelmente um website ou aplicação web. Para isso, sua integração deve ser clara e bem definida. Considerando a necessidade de exibir conteúdo com base na interação do usuário, podemos utilizar plenamente, funções integradas aos eventos HTML. Nesse contexto, crie uma página HTML que apresentará um campo para inserção de um nome e senha de usuário. Compare o nome com a senha e, se ambos forem diferentes, apresente uma mensagem em um elemento textual a sua escolha, no corpo de uma página HTML.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Essa dissertação de mestrado, intitulada Frameworks e Bibliotecas Javascript, apresenta uma visão de geração de frameworks e

bibliotecas que podem ser utilizadas para facilitar a vida dos desenvolvedores web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra campo de busca.

DUARTE, F. B. **Frameworks e Bibliotecas Javascript**. Porto: ISEP – Instituto Superior de Engenharia do Porto. Dissertação de Mestrado, 2015.

Indicação 2

Na obra *Apostila desenvolvimento web com HTML, CSS e JavaScript* é possível estudar maiores detalhes das tecnologias web, em especial a integração de JavaScript com as folhas de estilo e páginas em HTML.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra campo de busca.

K19 TREINAMENTOS. Apostila desenvolvimento web com HTML, CSS e javascript – TADS. Apostila, 2015..

QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco

e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. A linguagem de marcação de hipertexto (HTML), bem como a linguagem de folha de estilos (CSS), não é considerada linguagem de programação. Considerando que JavaScript ou ECMA Script é conhecida como uma linguagem de *scripting*, assinale a alternativa correta:
 - a. A linguagem JavaScript é uma linguagem de programação e é conhecida como uma linguagem de *scripting*, pois é interpretada pelo navegador.
 - b. A linguagem JavaScript não é uma linguagem de programação, tal como HTML e CSS, e é conhecida como uma linguagem de *scripting*, pois é interpretada pelo navegador.
 - c. A linguagem JavaScript é uma linguagem de programação e é conhecida como uma linguagem de *scripting*, pois é interpretada pelo compilador.
 - d. A linguagem JavaScript é uma linguagem de programação e é conhecida como uma linguagem de marcação, pois é interpretada pelo navegador.
 - e. A linguagem JavaScript é uma linguagem de programação e é conhecida como uma linguagem de *scripting*, pois é interpretada pela *Java Virtual Machine*.
2. Para a integração de um código desenvolvido em JavaScript, é necessária a utilização de algum evento HTML. Considerando tais eventos HTML, assinale a alternativa correta:
 - a. *Onchange* é um evento que é acionado quando uma determinada folha de estilo é modificada.

- b. *Onclick* é um evento acionado quando um usuário clica sobre um determinado elemento HTML que possui tal atributo de evento.
- c. *Onmouseover* é um evento acionado quando o ponteiro do mouse é retirado da área de um elemento HTML.
- d. *Onload* é um evento acionado ao carregar o arquivo CSS de uma página.
- e. *Onmouseout* é um evento acionado quando o ponteiro do mouse é colocado sob a área de um elemento HTML.



GABARITO

Questão 1 – Resposta A

Resolução: A linguagem JavaScript é uma linguagem de programação e é conhecida como uma linguagem de *scripting*, pois é interpretada pelo navegador, mais especificamente pelo interpretador que está inserido no navegador. Apesar do nome recordar a linguagem Java, não tem qualquer relação com a *Java Virtual Machine*, tão pouco pode ser considerada uma linguagem de marcação, como HTML ou de folha de estilos, como CSS.

Questão 2 – Resposta B

Resolução: *Onchange* é um evento acionado quando um documento HTML é modificado. Já o *onclick*, é acionado quando um determinado elemento é clicado. *Onmouseover* e *Onmouseout* correspondem ao colocar e retirar o ponteiro do mouse de um determinado elemento HTML. Ao fazer isso, o usuário acionará o código JavaScript indicado em tais atributos de eventos. Finalmente, o *onload* é um evento acionado ao carregar um documento HTML.



TEMA 4

Implementação de Páginas Web com HTML, CSS e JavaScript

Autoria: Anderson da Silva Marcolino

Leitura crítica: Gabriela Silveira



DIRETO AO PONTO

Para criarmos aplicações web e website robustos, a integração de diferentes tecnologias é necessária. As mais básicas são HTML, CSS e JavaScript.

Enquanto a linguagem de marcação de hipertexto (HTML) cria a estrutura básica com diferentes elementos. As folhas de estilo em cascata criam uma camada visual para um ou mais elementos do documento HTML. Finalmente, JavaScript cria interações mais atrativas e modificam o comportamento e conteúdo dos elementos HTML.

O primeiro passo para garantir a criação de um projeto consistente, e que seja padronizado e reconhecido pelo time de desenvolvedores, é criar uma estrutura básica para este projeto. Essa estrutura, composta por uma pasta raiz e subpastas que concentram arquivos de mesma extensão, como a subpasta *css*, que armazenará as folhas de estilos (*.css*), a subpasta *js* que armazenará arquivos JavaScript (*.js*) e a subpasta *html* que armazenará as demais páginas do website ou da aplicação web.

É importante mencionar que ao integrar um arcabouço ou *framework* ao projeto, a estrutura de pastas pode variar. Contudo, essa padronização básica deve nortear a organização do mesmo.

Criada a estrutura, a etapa seguinte é criar a página *index.html*, que é a página inicial do nosso website. É esta página que será apresentada e buscada pelo navegador, ao acessarmos um determinado site por meio de seu endereço. Adicionalmente, é nesta página que devemos integrar o arquivo *.css* da folha de estilos, principal, de nosso projeto. Podemos nomeá-lo como desejar, mas é fortemente indicado utilizar o nome *style.css*.

Por último, e não menos importante, devemos criar nosso arquivo JavaScript, que podemos chamar de *javascript.js*. Cada um desses arquivos será mantido em suas subpastas e não possui uma quantidade limite para os projetos.

Com o conjunto de arquivos pronto, pode-se começar a trabalhar no projeto em si. Como já mencionado, precisamos estruturar e incluir os elementos HTML na *index.html*. Nessa etapa, é importante haver consistência entre os elementos, o nome das classes utilizadas como seletores das folhas de estilo, os atributos de eventos e suas funções existentes no JavaScript.

A consistência de tais nomes são fundamentais, pois serão, consequentemente, integrados às regras das folhas de estilo e aos seletores de elementos HTML declarados no JavaScript.

Outro fator importante é a integração dos arquivos aos documentos HTML. Por exemplo, ao integrarmos outras páginas, como o arquivo CSS e o arquivo JavaScript na Index, devemos acessar, por meio do caminho onde estão armazenados, corretamente. Para isso, podemos colocar como referências, no atributo *href* um ponto, seguido de uma barra, a subpasta e o nome do arquivo com sua extensão. Por exemplo: *./css/style.css*. Para referenciarmos a própria index no menu, por exemplo, no link *home*, basta indicar o próprio arquivo *index.html* no atributo *href*, visto que o documento HTML não se encontra em uma subpasta, ou seja, está na mesma pasta.

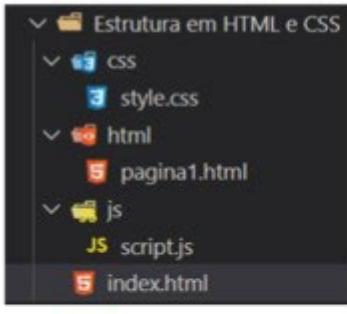
Já ao integrarmos a *index.html* em um documento HTML secundário qualquer, mantido na subpasta *html*, devemos retornar uma hierarquia de pastas. Enquanto o ponto e barra (“.”) permitem acessar uma subpasta, dois pontos e barra (“..”) permite navegar para uma pasta anterior. Por exemplo, para que um documento HTML chamado *pagina1.html* forneça acesso para o *index.html*, devemos indicar, no atributo *href*, dois pontos, barra e, então, o nome do arquivo *index.html*, ficando, portanto, o valor de *../index.html*.

No contexto da inclusão dos arquivos em nossos documentos HTML, devemos lembrar que a inclusão de um código JavaScript ocorre por meio do elemento *script*, que possui o atributo de origem do arquivo, em inglês, *source*. Esse atributo deve possuir valor similar ao *href* que permite a importação do CSS e outros documentos HTML. Por exemplo, para importar o arquivo *javascript.js* devemos usar um ponto, barra, nome da subpasta e nome do arquivo: *./js/javascript.js*.

Finalmente, podemos utilizar ainda, tanto para o valor do atributo *href*, quanto *src*, um link de um arquivo .css ou .js, entre outros, que estejam disponíveis na Internet, o que facilita o reuso de recursos de terceiros em nossos projetos.

A Figura 1 apresenta exemplos de utilização de arquivos e da estrutura básica de um projeto de um website, e apresenta: a) a estrutura de arquivos do projeto; b) excerto do arquivo *index.html*; c) excerto do arquivo *pagina1.html*; d) excerto do arquivo *style.css* e excerto do arquivo *javascript.js*.

Figura 1 – Estrutura e excerto de arquivos do projeto

a) Estrutura do projeto 	b) index.html <pre> 8 <link rel="stylesheet" href="css/style.css"/> 9 <title>Título da Barra de Títulos</title> 10 </head> 11 <body> 12 <header class="cabecalho"> 13 <h1>Cabeçalho do Site</h1> 14 </header> 15 <nav class="menu"> 16 17 Início 18 Tarefas </pre>
c) pagina1.html <pre> 8 <link rel="stylesheet" href="../css/style.css"/> 9 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"> 10 <title>Página 1</title> 11 </head> 12 <body> 13 <header class="cabecalho"> 14 <h1>Cabeçalho do Site</h1> 15 </header> 16 <nav class="menu"> 17 18 Início 19 Tarefas 20 21 </nav> 22 23 <div class="row"> 24 <div class="container"> 25 <div class="card"> 26 <h1 class="card-header">Lista de Tarefas</h1> 27 <div class="card-body"> 28 <div class="input-group mb-3"> 29 <input type="text" class="form-control" placeholder="Digite a tarefa" name="tarefa" autofocus> 30 <div class="input-group-append"> 31 <button class="btn btn-primary" type="button" id="botao">Cadastrar</button> 32 </div> </pre>	
d) style.css <pre> 20 .menu { 21 background-color: green; 22 overflow: hidden; 23 } </pre>	e) javascript.js <pre> 2 let btn = document.querySelector('#botao'); 3 let lista = document.querySelector('#lista'); 4 let card = document.querySelector('.card'); </pre>

Fonte: elaborada pelo autor.

Pode-se observar como os valores dos atributos *href* e *src* devem ser fornecidos, considerando a estrutura do projeto e um exemplo de seletor por classe do *css* e a declaração de variáveis, que armazenam a referência de elementos, como a do botão, identificado por meio de seu identificador e do cartão, por meio de sua classe.

Referências bibliográficas

W3SCHOOLS. **JavaScript and HTML DOM Reference**. 2020. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 4 mar. 2021.

SILVA, M. S. **JavaScript - Guia do Programador**: guia completo das funcionalidades de linguagem JavaScript. São Paulo: Novatec, 2020.

PARA SABER MAIS

Você sabia que é possível utilizar expressões regulares em JavaScript?

Uma expressão regular é uma sequência de caracteres que formam um padrão de busca. Quando você busca um determinado dado em seu texto, pode utilizar este padrão para descrever o que está procurando. Uma expressão regular pode ser um simples caractere ou um padrão mais complicado.

Pode-se utilizar uma expressão regular também para validação de dados, como, por exemplo, um CPF (Cadastro de Pessoa Física): ([0-9]{2}[\.]?[0-9]{3}[\.]?[0-9]{3}[\.]/?)[0-9]{4}[-]?[0-9]{2}) | ([0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{3}[\.]/?)[0-9]{2}).

A princípio, a leitura pode ser complexa, mas cada uma das partes permitirá verificar se os dígitos do CPF são válidos.

No contexto do JavaScript, podemos utilizar um exemplo mais simples:

```
/programar/i
```

Acima, temos uma expressão regular que utilizará o termo programar na busca. Sendo este termo modificado por *i* que indica que a busca diferencia maiúsculas e minúsculas.

Agora, para usar tal termo, devemos utilizar dois métodos: **string()** e **replace()**.

```
var str = "Amo programar!";
var n = str.search("programar");
```

O método de busca (**search()**) tenta localizar os caracteres passados por parâmetro e retorna a posição, caso encontrado. Se o valor não é retornado, o método retorna -1.

Para utilizarmos a expressão regular apresentada anteriormente, podemos substituir o parâmetro passado:

```
var n = str.search(/programar/i);
```

O métodos de substituição (**replace()**), trocará um termo buscado por outro passado como segundo parâmetro:

```
var str = " Amo programar!";
var res = str.replace("programar", "Cozinhar");
```

Esse método também aceita a passagem de uma expressão regular:

```
var res = str.replace(/programar/i, "Cozinhar");
```

Existem outros modificadores a serem utilizados com expressões regulares no lugar do *i*, como *g*, que realiza uma busca de todos os termos que sejam iguais ao da expressão regular; ou *m*, que realiza uma busca em múltiplas linhas.

Interessante, não? É importante mencionar que as expressões regulares são poderosas e podem ser utilizadas com outras linguagens e em outros contextos. Logo, são importantes e merecem ser estudadas!

Referências bibliográficas

W3SCHOOLS. **JavaScript and HTML DOM Reference**. 2020. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 4 mar. 2021.

TEORIA EM PRÁTICA

JavaScript é uma linguagem de programação que se integra aos documentos HTML, criando comportamento dinâmico aos diferentes elementos de tal documento. Considerando seu uso e integração, crie um documento HTML que apresente três modos de identificar um elemento do HTML na linguagem JavaScript, de modo a introduzir algum comportamento, como imprimir uma mensagem no console, exibir um alerta ou alterar o conteúdo diretamente, em um elemento HTML específico.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Esta dissertação de mestrado, intitulado *Frameworks e Bibliotecas Javascript*, apresenta uma visão da geração de frameworks e

bibliotecas, que podem ser utilizadas para facilitar a vida dos desenvolvedores web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra.

DUARTE, F. B. **Frameworks e Bibliotecas Javascript**. Dissertação de Mestrado. Porto: ISEP – Instituto Superior de Engenharia do Porto, 2015.

Indicação 2

Na *Apostila desenvolvimento web com HTML, CSS e JavaScript* é possível estudar maiores detalhes das tecnologias web, em especial a integração de JavaScript com as folhas de estilo e páginas em HTML.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra.

K19 TREINAMENTOS. **Apostila desenvolvimento web com HTML, CSS e javascript – TADS**. Apostila, 2015.

QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. A linguagem de marcação de hipertexto (HTML), a linguagem de folha de estilos (CSS) e a linguagem de programação JavaScript ou ECMA Script, são essenciais para a criação de um projeto básico de website. Sobre tais tecnologias, assinale a alternativa correta:

 - a. As folhas de estilo em cascata servem para estilizar os elementos HTML e elementos JavaScript.
 - b. A linguagem de marcação de hipertexto define a estrutura básica de elementos de uma página, não necessitando de folhas de estilos para definir interfaces atrativas.
 - c. A linguagem de marcação de hipertexto pode ser substituída totalmente por JavaScript.
 - d. A linguagem de folhas de estilo só pode ser utilizada em projetos que integrem códigos JavaScript.
 - e. A linguagem JavaScript é uma linguagem de programação que compõem documentos HTML, possibilitando integrar comportamentos dinâmicos, não obtidos pelo uso da linguagem de marcação apenas.
2. Para a criação de um website, devemos considerar um padrão mínimo na estrutura de website. Considerando tal estrutura, assinale a alternativa correta:

 - a. A escolha das pastas e subpastas de um projeto deverá ser realizada mediante acordo com o time de desenvolvedores e clientes.
 - b. Uma padronização indicada é a criação de subpastas que receberão e agruparão arquivos de mesma extensão.
 - c. Uma subpasta *html* é indicada para armazenar arquivos *.html*, incluindo o arquivo *index.html*.
 - d. Uma subpasta *css* deverá armazenar as folhas de estilo em cascata e todas as páginas *html*, que também integrem folhas de estilo.
 - e. A pasta *js* deverá armazenar arquivos JavaScript e também arquivos *.css*.



GABARITO

Questão 1 – Resposta E

Resolução: As folhas de estilo em cascata estilizam apenas os elementos definidos em HTML. Assim, podem ser utilizadas em projetos que não integram JavaScript. Ademais, o HTML por si só não provê meios de estilização de seus elementos, necessitando obrigatoriamente das folhas de estilo para isso. Nesse sentido, nota-se que a linguagem HTML não pode ser substituída totalmente por JavaScript, pois essa linguagem integra os documentos HTML de modo a resultar em comportamentos dinâmicos.

Questão 2 – Resposta B

Resolução: A escolha de uma padronização de pastas e subpastas deverá ocorrer entre o time de desenvolvedores somente, já que representam a parte interessada. Dentre tal padronização, uma indicada é a criação de subpastas que armazenarão arquivos que correspondam com a mesma extensão. Por exemplo, arquivos .css ficarão na pasta css, arquivos .js ficarão na pasta js. Já os arquivos html, com exceção do arquivo index.html, deverão ser mantidos na subpasta html. O index.html fica na pasta raiz por ser o arquivo principal e a primeira página que o navegador exibirá em um projeto de um website.



BONS ESTUDOS!