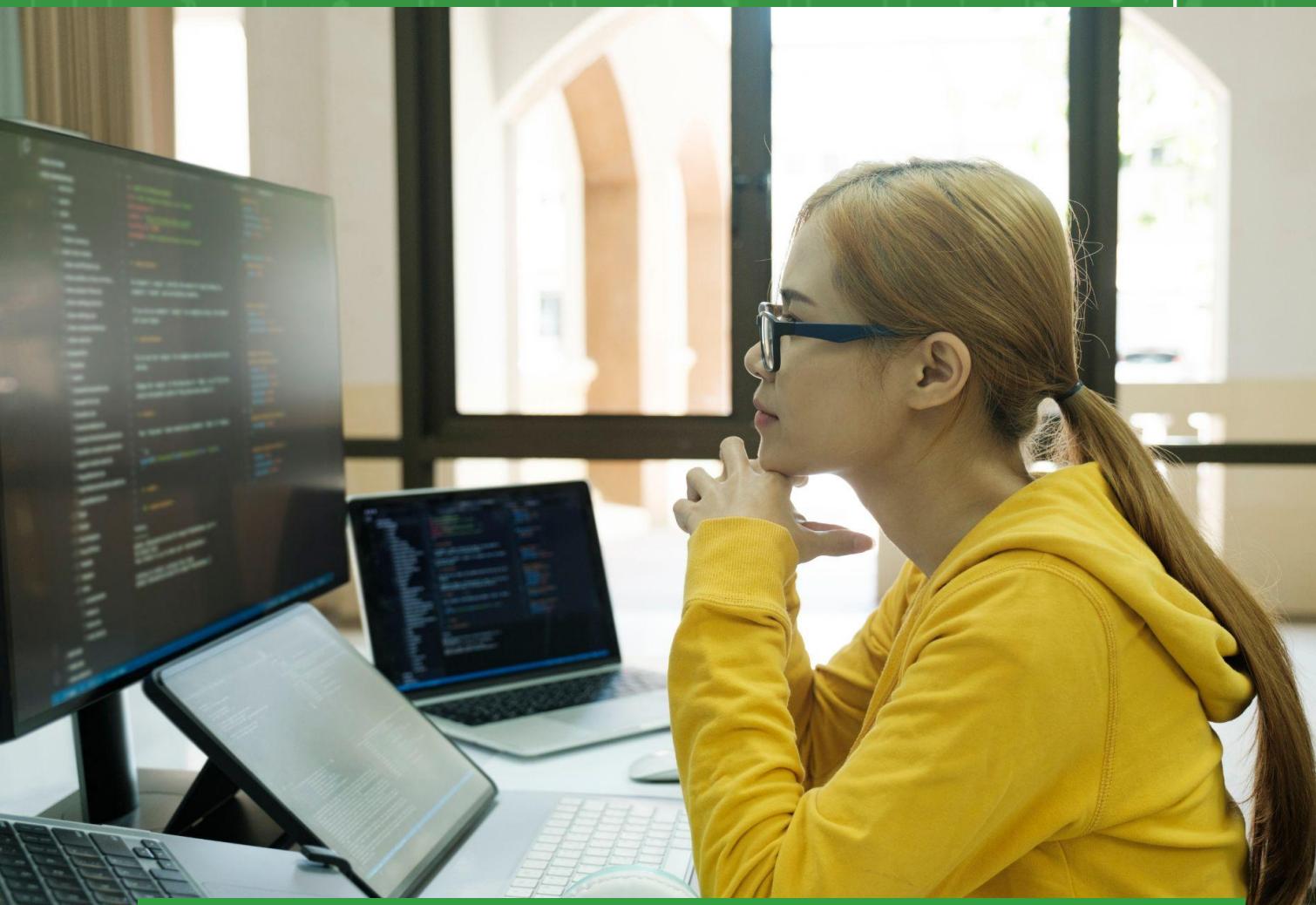


DESENVOLVIMENTO WEB I

CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET



Créditos

Equipe IFSC - Santa Catarina

Professor

Herval Daminelli

Equipe Designa Instrucional

Coordenação de Projeto

Cíntia Costa

Ester Konig

Coordenação Design Instrucional

Emily Mercuri

Design Instrucional

Joyce Paola Mangrich

Design Gráfico

Ayrin Barboza

Revisão ortográfica

Priscila Verçosa

Este trabalho está licenciado
sob CC BY-NC 4.0 



Ficha catalográfica

Apresentação

Nesta Unidade Curricular, você conhecerá conceitos e técnicas relacionados ao desenvolvimento de aplicações para a internet. Iniciaremos a primeira semana de estudos desta UC apresentando uma visão geral dos aspectos relativos ao Desenvolvimento de Aplicações Web.

Em seguida, apresentaremos os estudos abordando sobre os elementos básicos da linguagem HTML (camada de estrutura) para a marcação de páginas web.

Na sequência, abordaremos os componentes básicos das Folhas de Estilo em Cascata (CSS), que definem todo o aspecto visual e a camada de apresentação dos elementos em uma página web. Esses tópicos compõem o conteúdo compreendido entre a semana 11 e a semana 16.

Por último, para compor a construção da camada de comportamento de uma página web, abordaremos, entre as semanas 17 e 20, os conceitos da linguagem de Programação JavaScript, possibilitando uma ampla variedade de interações do usuário com nossa aplicação, para torná-la verdadeiramente uma aplicação web dinâmica.

Esse conjunto de conhecimentos é fundamental para o estudante, pois forma a base necessária para que ele consiga atuar profissionalmente com o mínimo de conhecimento na área de Desenvolvimento Web.

O Desenvolvimento Web é um dos setores mais promissores da Tecnologia da Informação, já que existe uma tendência, na atualidade, de se criar e desenvolver sistemas e aplicações que sejam executados diretamente pelo navegador do computador do cliente da aplicação, em oposição às chamadas aplicações desktop. Nesse cenário, a atividade do desenvolvedor web torna-se cada vez mais relevante, abrindo novas perspectivas e oportunidades para a atuação desse profissional em relação ao mundo do trabalho no qual ele está inserido.

Objetivo

- Compreender e utilizar conceitos fundamentais relacionados à construção da interface gráfica de uma aplicação para a web.

Sumário



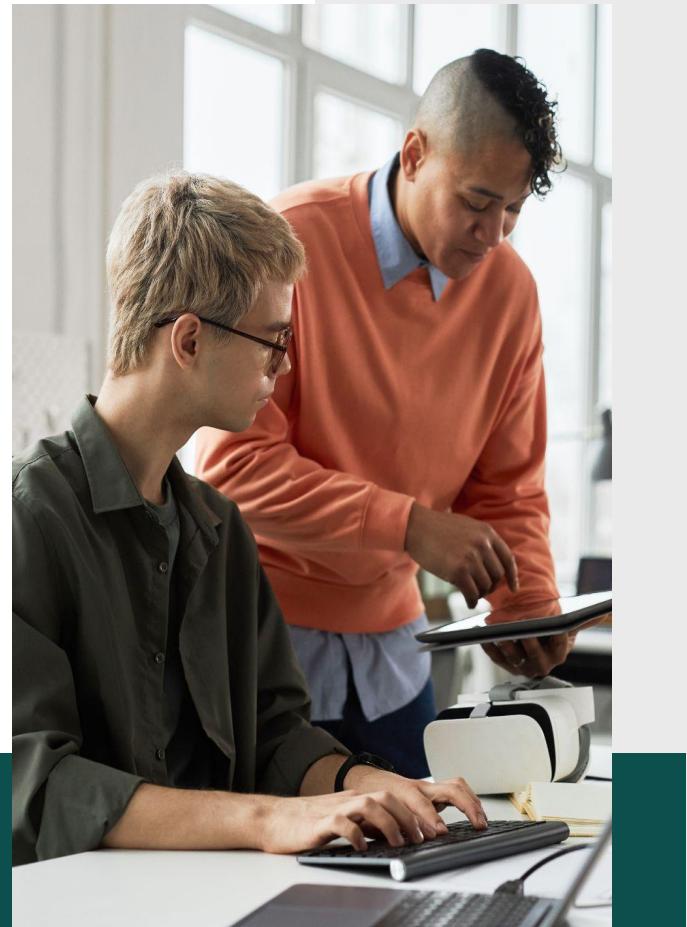
| | |
|--------------------------------------------------------------------------------|-----|
| 1. Fundamentos da programação web | 3 |
| 2. Introdução à linguagem HTML | 9 |
| 3. Estrutura básica de um documento web | 13 |
| 4. Marcação de texto em uma página web | 17 |
| 5. Marcação de listas em HTML5 | 20 |
| 6. Marcação de links na linguagem HTML5 | 23 |
| 7. Marcação de imagens em HTML5 | 26 |
| 8. Marcação de tabelas na linguagem HTML5 | 28 |
| 9. Criação de formulários em uma página web (etapa 1) | 33 |
| 10. Criação de formulários em uma página web (etapa 2) | 39 |
| 11. Introdução às Folhas de Estilo em Cascata – CSS | 44 |
| 12. Formatação de texto básica com CSS | 51 |
| 13. Formatação de tabelas | 55 |
| 14. Usando imagens de fundo com CSS | 59 |
| 15. Formatação de formulários (etapa 1) | 64 |
| 16. Formatação de formulários com CSS (parte 2) | 71 |
| 17. Introdução à linguagem de programação JavaScript | 78 |
| 18. Constantes, variáveis, operadores e tipos de dados na linguagem JavaScript | 80 |
| 19. Formas de inserção do JavaScript em um documento web | 88 |
| 20. Funções de usuário na linguagem JavaScript | 106 |
| Finalizando | 105 |
| Referências | 107 |

1. Fundamentos da Programação Web

A Programação Web é um conjunto de processos e técnicas que permitem a construção de aplicações web funcionais, também conhecidas como páginas da internet. Essas páginas compõem o que chamamos de interface gráfica de uma aplicação web, que, em última análise, permite a interação do usuário com os componentes da aplicação.

Por meio dessa interação, o usuário de uma aplicação web pode fornecer informações necessárias à execução de todo o processo. Além disso, o usuário ou cliente de uma aplicação web também pode receber respostas de solicitações que foram feitas nessa mesma interface.

Uma aplicação web completa tem alguns requisitos básicos para que possa funcionar. Esses requisitos envolvem conceitos como:



1

Dados

infraestrutura de comunicação para troca de dados.

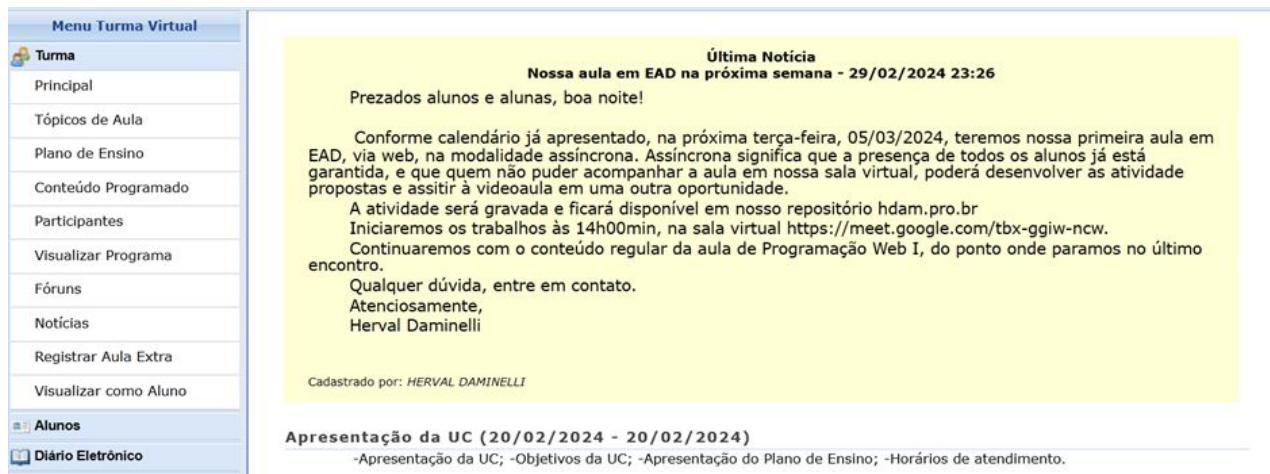
2

Interface gráfica

infraestrutura para utilização da interface gráfica de uma aplicação web.

Detalharemos esses conceitos mais adiante em nossos estudos. A seguir, você pode conferir a interface gráfica típica de uma aplicação web:

Exemplo de interface gráfica de uma aplicação web



The screenshot shows a left sidebar menu with sections like 'Turma' (Principal, Tópicos de Aula, etc.), 'Alunos' (Diário Eletrônico), and 'Aulas' (Apresentação da UC). The main content area displays a yellow-highlighted news item:

Última Notícia
Nossa aula em EAD na próxima semana - 29/02/2024 23:26
Prezados alunos e alunas, boa noite!

Conforme calendário já apresentado, na próxima terça-feira, 05/03/2024, teremos nossa primeira aula em EAD, via web, na modalidade assíncrona. Assíncrona significa que a presença de todos os alunos já está garantida, e que quem não puder acompanhar a aula em nossa sala virtual, poderá desenvolver as atividades propostas e assistir à videoaula em uma outra oportunidade.

A atividade será gravada e ficará disponível em nosso repositório hdam.pro.br
Iniciaremos os trabalhos às 14h00min, na sala virtual <https://meet.google.com/tbx-ggiw-ncw>.
Continuaremos com o conteúdo regular da aula de Programação Web I, do ponto onde paramos no último encontro.
Qualquer dúvida, entre em contato.
Atenciosamente,
Herval Daminelli

Cadastrado por: HERVAL DAMINELLI

Apresentação da UC (20/02/2024 - 20/02/2024)
-Apresentação da UC; -Objetivos da UC; -Apresentação do Plano de Ensino; -Horários de atendimento.

Fonte: Adaptado do Sigaa (2024).

Acompanhe, agora, como funciona um programa de computador.

Programa de computador

Um programa de computador também pode ser conhecido como um sistema computacional. De forma mais simplificada, podemos dizer que é uma sequência de ordens ou comandos, escrita de forma estruturada, que é executada pela máquina, com o objetivo de resolver determinado problema do mundo real.

**Podemos citar, como exemplos de
programa de computador, um editor de
texto, um jogo de cartas, um antivírus, um
navegador de internet etc.**

Esse conjunto de comandos é escrito com uma linguagem computacional adequada, que, quando interpretada pela máquina, devolve os resultados que resolvem o problema proposto.

Atualmente, existem várias linguagens computacionais disponíveis para codificar um conjunto de instruções que resolverão determinado problema, isto é, o programa de computador. A questão sobre qual linguagem de computador escolher está intimamente ligada ao tipo de problema a ser resolvido. Assim, quando falamos do desenvolvimento de uma aplicação web, devemos trabalhar com linguagens e tecnologias desenvolvidas especificamente para essa finalidade.



Quando falamos sobre linguagens de programação da atualidade, podemos citar os seguintes exemplos: Java, PHP, JavaScript, Python, C++.

Confira, a seguir, como funciona uma aplicação para web.

Aplicação para a web

Como vimos anteriormente, sempre que precisamos automatizar a resolução de um problema do mundo real usando um dispositivo computacional, devemos elaborar um programa de computador que resolva o problema proposto.

Em particular, no caso de um programa que é executado com a infraestrutura da internet, isto é, uma aplicação web, é preciso atender a dois requisitos. Veja no infográfico a seguir.

Requisitos

Conexão

O sistema computacional deve estar conectado à internet.

1.



Navegador

O sistema computacional deve ter um navegador de internet instalado.

2.



E por que existem esses dois requisitos? A definição de uma aplicação web supõe que o usuário da aplicação interaja com ela, isto é, forneça dados de entrada e receba respostas de suas requisições — resolução do problema — por meio de um navegador web.

Além disso, o processamento dos dados fornecidos à interface gráfica de uma aplicação web será feito em um outro local, ou seja, por um outro computador, que não é o mesmo utilizado pelo usuário da aplicação. E, para que haja essa troca de informações entre a máquina-cliente e a máquina servidora (arquitetura cliente-servidor), é necessário que ambas estejam conectadas entre si pela internet.

Arquitetura cliente-servidor

Quando nos referimos ao desenvolvimento de uma aplicação web, devemos perceber que todo o processo de elaboração e execução do código necessário para tal é separado em dois grandes blocos ou estágios.

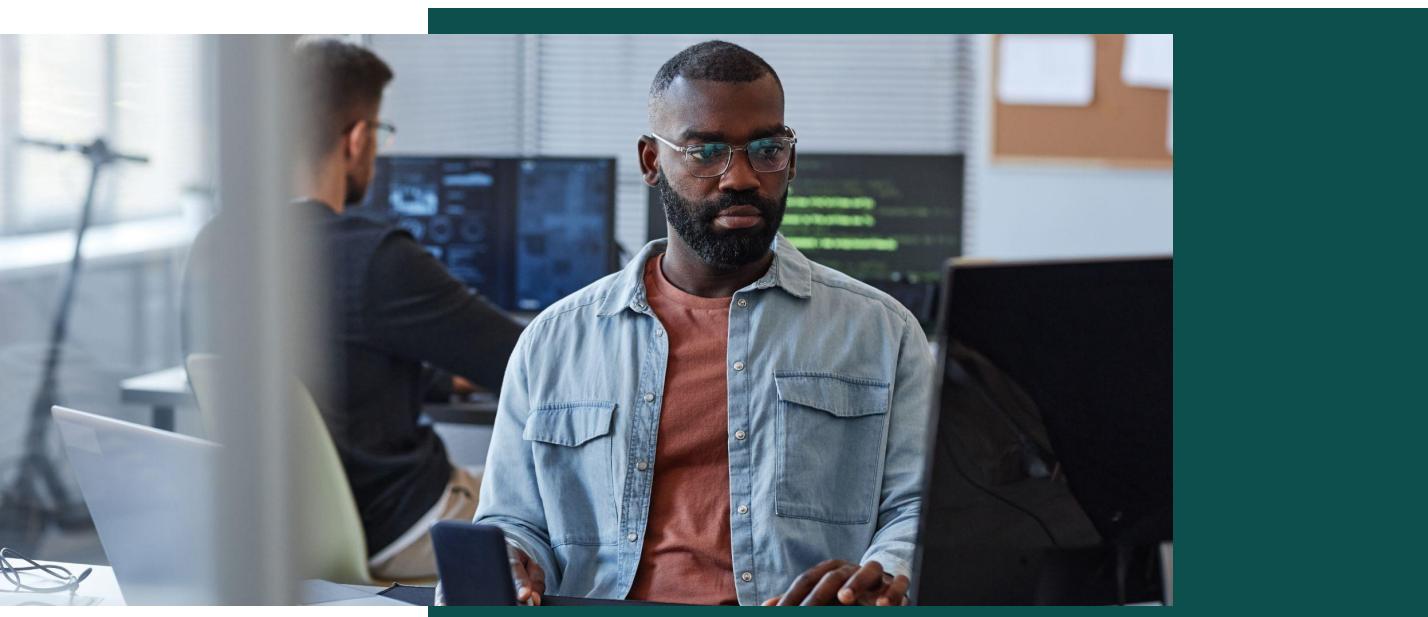
De um lado, precisamos desenvolver o código e utilizar tecnologias que serão executadas na máquina do usuário da aplicação, que interage diretamente com o navegador web.

Esse conjunto de códigos, produzido por linguagens e tecnologias próprias, é o que definimos como front-end da aplicação web, executado na máquina-cliente.

Por meio da manipulação da interface gráfica da aplicação web (páginas de internet), o usuário (ou cliente) fornece as informações necessárias para que tal aplicação possa funcionar.

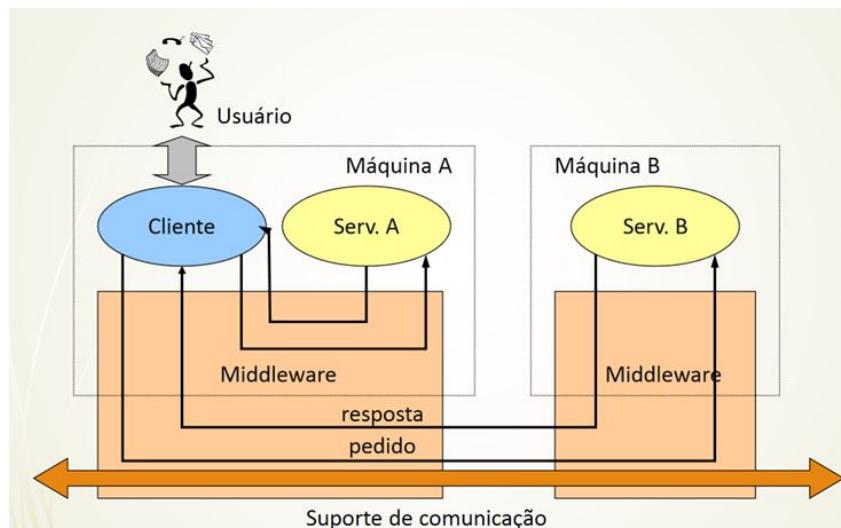
Em contrapartida, esses dados são transportados e executados, via web, em outro local, em que um conjunto de linguagens e tecnologias próprias, constantes de nossa aplicação, se encarregam de receber tais dados, processá-los e enviar o resultado de volta para o usuário.

A outra “ponta” dessa engrenagem complexa é definida como servidor da aplicação web. E, no servidor, o conjunto de códigos e linguagens utilizadas compõe o que conhecemos como **back-end** de uma aplicação web.



A seguir, temos uma ilustração que resume o que foi apresentado aqui:

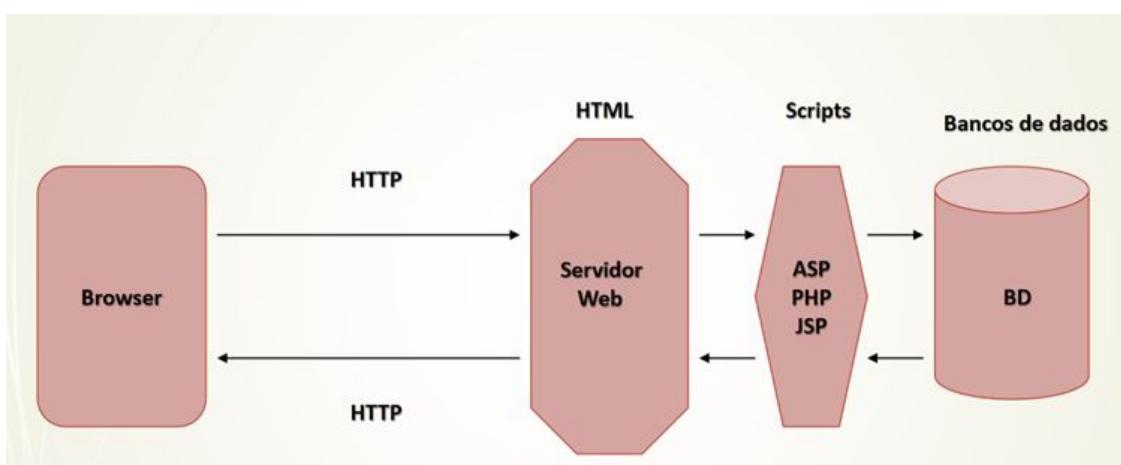
Exemplo de arquitetura cliente-servidor



Fonte: Elaborada pelo autor (2024).

Temos, também, outra forma de representar a estrutura cliente-servidor:

Exemplo de estrutura cliente-servidor



Fonte: Elaborada pelo autor (2024).

Para finalizar este tópico, vale mencionar que esta Unidade Curricular tratará sobre o desenvolvimento de um código que pode ser manipulado e executado pela máquina-cliente, com a aplicação de linguagens e tecnologias específicas para esse fim. O restante do código necessário para que tenhamos uma aplicação web completa será visto em outras Unidades Curriculares.

Confira, a seguir, uma introdução à linguagem HTML.

2. Introdução à linguagem HTML

A linguagem HTML, na sua versão 5, é utilizada para a criação de páginas web e, consequentemente, para criar a **interface gráfica** de uma aplicação web.

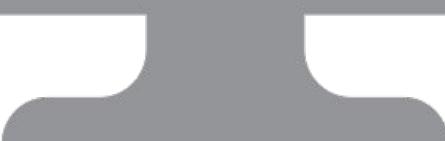
Mas o que é HTML? HTML é o acrônimo do termo em inglês HyperText Markup Language, ou seja, linguagem de marcação de hipertexto.

Hipertexto é uma forma de escrita não linear, não sequencial, que permite que o usuário acesse diretamente o conteúdo de um documento web sem precisar seguir uma ordem de navegação, isto é, sem necessitar começar pela página 1, depois seguir para a página 2 e, assim, sucessivamente.

O elemento que permite esse tipo de navegação (hipertexto) é chamado de link. Você poderá conhecer mais sobre ele em tópicos posteriores. Vamos lá?

Cyberpedia

Interface gráfica: é um conceito que se refere à forma como um usuário interage com uma aplicação computacional e, de forma simples, é representada por uma “tela” do programa, contendo elementos visuais como botões, imagens, menus de opções, entre outros.



Navegadores web

Um navegador de internet é um software que permite a exibição e o tratamento do conteúdo de uma página de internet. Portanto, é um software fundamental, pois sem ele não é possível utilizar um sistema para a web.

Atualmente, existem vários navegadores no mercado. Dentre eles, podemos citar Google Chrome, Firefox Mozilla, Microsoft Edge etc.

Ferramentas de desenvolvimento web

Você sabia que é possível criar páginas web em qualquer editor de texto? Isso acontece porque o arquivo gerado pela linguagem HTML é sempre um arquivo de texto.



No entanto, para facilitar e agilizar o processo de desenvolvimento da interface gráfica de um programa web, ou seja, as páginas web renderizadas pelo navegador, utilizadas pelo cliente e que compõem o front-end da aplicação, foram criadas ferramentas, ou editores de código, próprias para esse fim.

Existem vários softwares que atendem a essa necessidade. Um dos mais utilizados nessa fase inicial do desenvolvimento de uma aplicação web é o Visual Studio Code, que, além de ser gratuito, é um excelente editor de código-fonte leve.



O Visual Studio Code (VS Code) é um editor de código-fonte desenvolvido pela Microsoft. É gratuito, de código aberto e altamente extensível, oferecendo suporte a várias linguagens de programação e frameworks. Para baixar e instalar o Visual Studio Code, clique no botão.

[Acesse](#)

Depois de conhecer os navegadores e as ferramentas de desenvolvimento web, acompanhe como funciona a estrutura da linguagem HTML.

Estrutura da linguagem HTML

Uma página web é composta por elementos da linguagem HTML, chamados de **marcadores** ou **tags**. Quando renderizados pelo navegador, esses elementos indicam como a informação constante naquele marcador deve ser estruturada na página web.

Marcadores e tags

01 Texto

Se quisermos que o navegador exiba um texto qualquer, devemos utilizar o marcador criado na linguagem para esse fim.

02 Imagem

Se quisermos apresentar uma imagem na página web, devemos indicar essa situação ao navegador, utilizando um marcador criado especificamente para a representação de imagens.

De modo geral, a linguagem HTML oferece marcadores (ou tags) para que o navegador exiba conteúdo dos mais diferentes tipos, como áudio, vídeo, tabelas, controles de formulário, listas, links e tantos outros mais.

Anatomia de uma tag na linguagem HTML5

Vamos conhecer um marcador na linguagem HTML5 na prática? Analise a imagem a seguir:



Fonte: Elaborada pelo autor (2024).

As tags HTML, ainda, podem ser de dois tipos:

- **completas:** como na ilustração anterior;
- **vazias:** quando não apresentam conteúdo e nem a parte final.

Uma tag HTML começa com o símbolo <, depois vem o nome da tag (neste caso, o nome é a tag a, indicando um link) e o sinal de >. Se a tag for completa, ela continua com algum conteúdo neste ponto. Se ela for vazia, ela é encerrada por aqui. Sendo completa, a tag exige o fechamento (final). O final de uma tag é dado pelo símbolo <, a barra normal /, o nome da tag novamente e, finalmente, o sinal de >.

Alguns marcadores necessitam de uma informação adicional, dentro do início da tag, para que o navegador consiga renderizar corretamente a informação que se necessita. Essa informação adicional é dada na forma de um par, chamado de par atributo-valor. Todo atributo deve ter um nome, seguido de um sinal de igual, e,

após o sinal, entre aspas ou apóstrofos, um determinado valor de atributo. O valor depende da utilização de cada atributo.

A estrutura básica de um documento web será apresentada na próxima semana.
Não deixe de acompanhar!

3. Estrutura básica de um documento web

A estrutura básica de uma página HTML é dada pelo seguinte esquema:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8" >
    <title>  </title>
</head>

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

A **primeira linha** do arquivo HTML, que irá representar a primeira página de nossa aplicação web, indica ao navegador qual versão da linguagem estamos usando para criar esse arquivo. Aqui, o termo `<!doctype html>` indica que a versão utilizada é a 5.0.

Segundo, a segunda linha indica ao navegador que neste ponto começa, de fato, a página web. A tag `<html>` indica o início da página web. O atributo e o seu valor, que vemos dentro desse marcador, indicam ao navegador que o conteúdo de texto da

página como um todo estará no idioma português do Brasil. Outros códigos de outros idiomas podem ser usados no lugar dele.

Depois, vemos que o documento apresenta a tag <head> e, mais abaixo, a tag <body>. Confira no infográfico a seguir:

Tags

<head>

A tag <head> divide o documento em uma seção especial, em que colocamos informações adicionais para que o navegador exiba corretamente o conteúdo da página.



<title>

A tag <title>, dentro desta seção, exibe o título da página na aba do navegador.



<html>

Finalmente, dizemos ao navegador que a página termina ao fazermos o fechamento da tag </html>.



<meta>

Dentro desta seção, temos uma tag chamada <meta>, cujo atributo permite ao navegador exibir muitos símbolos especiais, que não estão presentes no teclado.



<body>

A última seção desse documento é dada por este marcador. É aqui que, de fato, colocamos todo o conteúdo a ser exibido pelo navegador, como texto, imagens, áudio, vídeo, tabelas, controle de formulários etc., como veremos mais adiante.



Adiante, você vai poder conhecer a marcação de texto em uma página web.

4. Marcação de texto em uma página web

Em geral, existem várias tags que utilizamos para a marcação de texto nessa linguagem. As mais comuns são:



`<p>`

A tag `<p>`, completa, para marcar texto de importância normal.



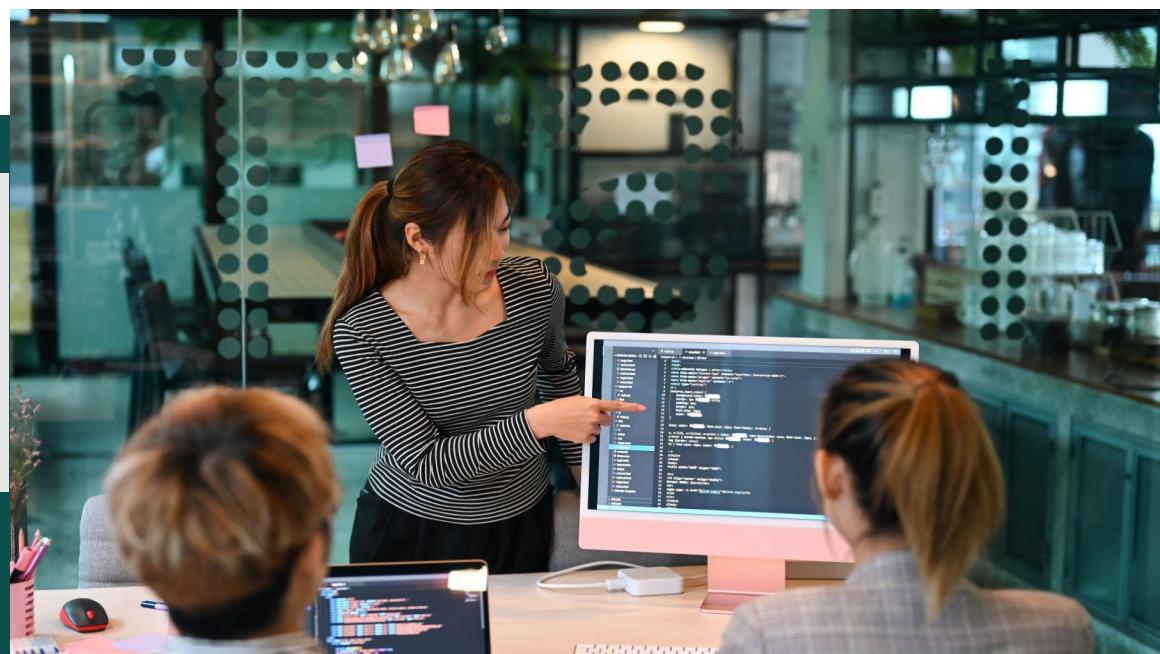
`<h1>, <h2>, <h3>, <h4>, <h5> e <h6>`

As tags de cabeçalho `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` e `<h6>` indicam ao navegador que ele deve ser apresentado de forma diferente do conteúdo regular, pois é o conteúdo mais importante da página.



``

A tag ``, completa, para marcar o texto que deverá ser estilizado e apresentado de modo diferente do texto regular.



Todo e qualquer texto renderizado em uma tag de cabeçalho aparece com a fonte negritada no navegador.

Confira o código de uma página web simples mostrando a formatação básica de texto em HTML5.

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Minha primeira página web </title>
</head>

<body>
    <h1> Este é o título da página web - conteúdo muito
importante </h1>
    <h2> Este é um subtítulo da página web - conteúdo ainda
relevante, mas de importância menor que o do título </h2>
    <h3> Este é um outro subtítulo que dividiu o anterior -
de menor importância </h3>

    <p> Este é um parágrafo regular de texto. Lorem ipsum
dolor sit amet, consectetur adipisicing elit. Molestias
rerum nihil similique deleniti optio error esse maxime
non! Dolores alias hic voluptates, nesciunt est dolore
vitae quidem quae cumque mollitia. Lorem ipsum dolor sit
amet consectetur adipisicing elit. Vero incident quan
neque asperiores alias ex earum dignissimos corporis.
</p>
    <p> Este é um parágrafo regular de texto. Lorem ipsum
dolor sit amet, consectetur adipisicing elit. Molestias
rerum nihil similique deleniti optio error esse maxime
non! Dolores alias hic voluptates, nesciunt est dolore
vitae quidem quae cumque mollitia. </p>
</body>
</html>
```

A seguir, temos um print de como esse código é renderizado na forma de página web no navegador:

Código renderizado



Este é o título da página web - conteúdo muito importante

Título <h1>

Este é um subtítulo da página web - conteúdo ainda relevante, mas de importância menor que o do título

Subtítulo <h2>

Este é um outro subtítulo que dividiu o anterior - de menor importância

Subtítulo <h3>

Este é um parágrafo regular de texto. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Molestias rerum nihil similique deleniti optio error esse maxime non! Dolores alias hic voluptates, nesciunt est dolore vitae quidem quae cumque mollitia. Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem molestias cumque nesciunt laudantium necessitatibus ullam vel molestiae eveniet maxime voluptas. Vero incidentum quam neque asperiores alias ex earum dignissimos corporis.

Parágrafo regular <p>

Este é um parágrafo regular de texto. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Molestias rerum nihil similique deleniti optio error esse maxime non! Dolores alias hic voluptates, nesciunt est dolore vitae quidem quae cumque mollitia. Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem molestias cumque nesciunt laudantium necessitatibus ullam vel molestiae eveniet maxime voluptas. Vero incidentum quam neque asperiores alias ex earum dignissimos corporis.

Fonte: Elaborada pelo autor (2024).

Acompanhe, agora, como funciona a marcação de listas em HTML5.

5. Marcação de listas em HTML5

Para que o navegador exiba na interface de nossa aplicação web uma informação na forma de uma relação de itens, podemos utilizar os **marcadores de lista**.

Basicamente, é possível ter listas em que a ordem sequencial de cada item é importante. Também, podemos ter listas em que a especificação de cada item de informação não precisa obedecer a uma ordem predeterminada.

Por exemplo, no primeiro caso, podemos citar os passos necessários para a troca de uma lâmpada queimada no teto da cozinha de nossa casa. No segundo caso, podemos citar a lista de material escolar a ser adquirida. Veja mais detalhes a seguir.

Listas numeradas

As listas numeradas são aquelas em que a ordem de cada item importa. A tag-mãe, que inicia a marcação na página web, é a tag ``. Dentro dessa tag, indicamos cada item da lista pelo marcador ``. O navegador pode usar símbolos chamados de numeradores (algarismos arábicos, algarismos romanos etc.) antes de cada item da lista.

A seguir, confira o código HTML de uma lista, representando um conjunto de passos mínimo para trocar uma lâmpada do teto da cozinha de nossa casa:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Listas numeradas </title>
</head>

<body>
    <p> Troca de lâmpada </p>

    <ol>
        <li> Desligar a energia elétrica </li>
        <li> Providenciar uma lâmpada nova </li>
        <li> Posicionar uma escada abaixo da lâmpada </li>
        <li> Subir na escada com a lâmpada nova e
            providenciar a troca </li>
        <li> Descer da escada e religar a energia elétrica
    </li>
    </ol>
</body>
</html>
```

E como funcionam as listas não numeradas? Acompanhe adiante.

Listas não numeradas

As listas não numeradas são aquelas em que a ordem de cada item não precisa obedecer a uma determinada sequência. A tag-mãe, que inicia a marcação na página web, é a tag ``. Dentro dessa tag, indicamos cada item da lista pelo marcador ``. O navegador pode usar símbolos chamados de marcadores (círculo, circunferência e quadrado, entre outros) antes de cada item da lista.

Confira, agora, o código HTML usado para mostrar uma pequena lista de material escolar a ser adquirido antes do início do período escolar:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Listas não-numeradas </title>
</head>

<body>
    <p> Relação de material escolar </p>

    <ul>
        <li> Livros </li>
        <li> Cadernos </li>
        <li> Mochila </li>
        <li> Lancheira </li>
        <li> Tablet </li>
    </ul>
</body>
</html>
```

Temos, ainda, a marcação de links na linguagem HTML5, como você pode conferir no tópico seguinte.

6. Marcação de links na linguagem HTML5

A capacidade que os navegadores oferecem de o usuário navegar por várias páginas, sem obedecer a uma ordem sequencial é dada por uma das tags mais importantes e mais antigas da linguagem: a tag de link, simbolizada por `<a>`.

Por meio de um link no documento web, o usuário pode solicitar ao navegador que o conduza a qualquer outra página disponibilizada em nossa aplicação.

A tag que simboliza um link é completa, tendo abertura e fechamento. Além disso, é obrigatório, dentro da abertura da tag, o uso do atributo `href`. Para que serve esse atributo? Ele indica ao navegador quem será (e onde estará) a nova página ou o novo recurso a ser buscado pelo navegador.

Observe um pequeno trecho de código que dá indicações ao navegador:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Links </title>
</head>

<body>
    <h1> Links em um documento web </h1>

    <!-- Usando a tag <a> -->
    <a href="cadastro-cliente.html" target="_blank"> Clique
    aqui para acessar a página de cadastro do cliente </a>
</body>
</html>
```

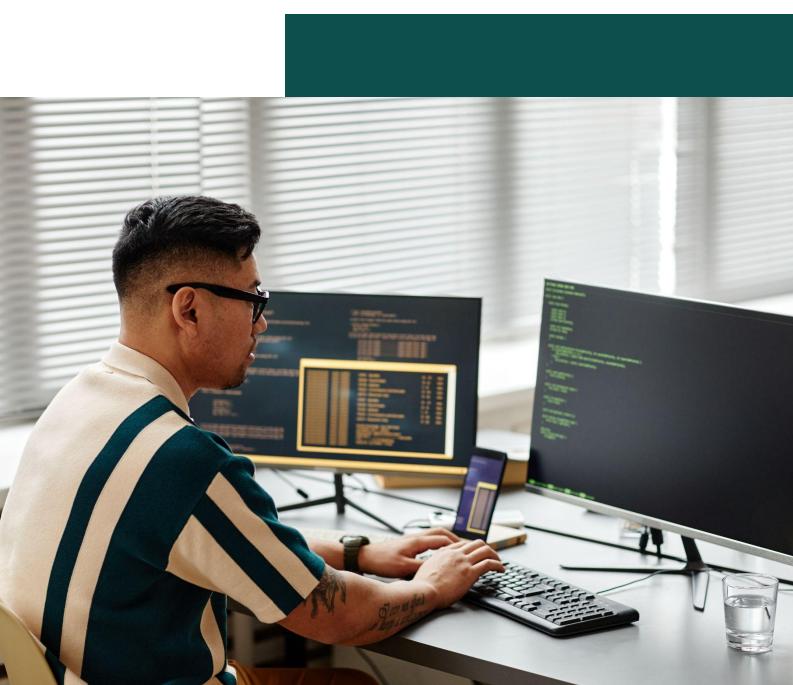
Em relação ao código anterior, observe que o atributo **href** diz ao navegador, para quando o link for clicado, buscar e mostrar ao usuário uma nova página web com o nome “cadastro-cliente.html”. Em nosso caso, essa página não existe de fato, é fictícia e mostrada aqui somente como exemplo.

Note, também, a presença de um atributo (opcional dentro da tag) com o nome de **target**, cujo valor é **_blank**. Se esse atributo estiver presente, o navegador exibirá a nova página em uma nova aba. Caso ele seja omitido, o navegador substitui a página anterior e mostra o novo documento na mesma aba.

Observe, no trecho de código anterior, o uso de um marcador um tanto “estranho”. Ele começa com o símbolo <!-- e termina com o formato -->.

Trata-se da tag de comentário interno ao código. Sempre que o navegador encontrar essa tag, toda informação contida dentro dela será desprezada e não renderizada pelo browser (navegador).

Então, se não é mostrado na página web, para que serve um comentário? Ele é útil para o próprio desenvolvedor fazer pequenas observações de trechos de código que são muito importantes. Essas observações podem se revelar muito úteis se, digamos, depois de um tempo relativamente longo, o documento tiver de ser revisitado, para manutenção ou modificação. Sem essas anotações, pode ser difícil lembrar o que essa parte específica do código está fazendo.



A documentação interna é considerada uma boa prática de codificação e deve ser utilizada com qualquer linguagem para desenvolvimento web que estivermos utilizando.

Para saber como funciona a marcação de imagens em HTML5, acompanhe o conteúdo da próxima semana.

7. Marcação de imagens em HTML5

O uso de imagens na composição da interface gráfica de uma aplicação web é fundamental, devido à facilidade de transmitir ao receptor uma determinada mensagem por meio desse recurso.

Sendo assim, vale a pena dar uma olhada mais de perto nos elementos que a linguagem HTML5 oferece para o tratamento de imagens em uma página web.

A principal tag é a . Ela é uma tag vazia, portanto, não tem fechamento. Os dois atributos obrigatórios da tag são:

Tag

01

src

Atributo que indica qual o nome do arquivo de imagem que será apresentado pelo navegador e qual a localização desse arquivo dentro da estrutura de pastas que compõem nosso sistema.

02

alt

Atributo relacionado ao quesito de acessibilidade de uma página web, que obriga o navegador a exibir uma informação alternativa no local da imagem, quando ela não puder ser renderizada corretamente.

Além desses atributos, o marcador de imagem pode apresentar outros de menor importância, que não serão abordados aqui.

Observe um pequeno pedaço de código utilizado para mostrar algumas imagens na interface de nossa aplicação:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Imagens com HTML5 </title>
</head>

<body>
    <h1> Manipulando imagens em uma página web </h1>

     <br>

    <img src=./imagens/icones.png" alt="Imagen com a relação de ícones usados em nossa aplicação web">
</body>
</html>
```

Vale destacar que, embora a tag `` seja a principal utilizada para a renderização de imagens, ela não é a única.



Existem outros dois marcadores que podem ser, também, utilizados nesse contexto. São eles: `<figcaption>` e `<figure>`. Você encontra mais informações sobre esse elemento clicando no botão. [Acesse](#)

Agora que você já conhece a marcação de imagens em HTML5, convido-lhe a saber como funciona a marcação de tabelas na linguagem HTML.

8. Marcação de tabelas na linguagem HTML5

As tabelas são componentes de uma página web, utilizados para mostrar dados no formato tabular. Uma tabela é composta de várias linhas e colunas, mas é possível criar com apenas uma linha e uma coluna também, e o cruzamento (intersecção) de uma linha com uma coluna é denominado “célula”.

Na linguagem HTML5, a tag-mãe de uma tabela, isto é, a tag que engloba todos os elementos relacionados a uma tabela, é a tag <table>. Essa tag define o início de uma tabela, mas, por si só, não mostra nada na página web.

Após indicarmos ao navegador que estamos iniciando a estruturação de uma tabela, devemos, em seguida, indicar quais linhas compõem a tabela. Cada linha é marcada pela tag <tr>, e a renderização de uma linha (ou célula) é sempre feita, pelo navegador, da esquerda para a direita.

Dentro de cada linha <tr>, devemos ter uma tag que indica qual a célula em que será inserido algum conteúdo. Para células, temos dois tipos de tags, que diferem entre si de acordo com o tipo de dado que a célula irá conter.

01

<td>

A célula de dado regular é marcada pela tag `<td>`. Dentro dessa célula, colocamos o dado propriamente dito, que está sendo exibido pela tabela.

02

<th>

A célula de cabeçalho da tabela é especial, sendo destinada a exibir o título (identificação) dos dados da referida linha ou coluna. Marcada como `<th>`.

Note que o formato do conteúdo apresentado dentro de uma célula `<td>` ou `<th>` tem algumas diferenças entre si. Se for somente texto, o conteúdo de uma célula `<td>` não sofre nenhuma formatação especial (por isso, conteúdo regular). Se for texto que está sendo inserido dentro de uma célula `<th>` (célula de cabeçalho), ele será renderizado pelo navegador com algumas características específicas, como é mostrado no infográfico a seguir:

Formatação



Negrito

O texto de uma célula `<th>` sempre aparece marcado em negrito.



Alinhamento

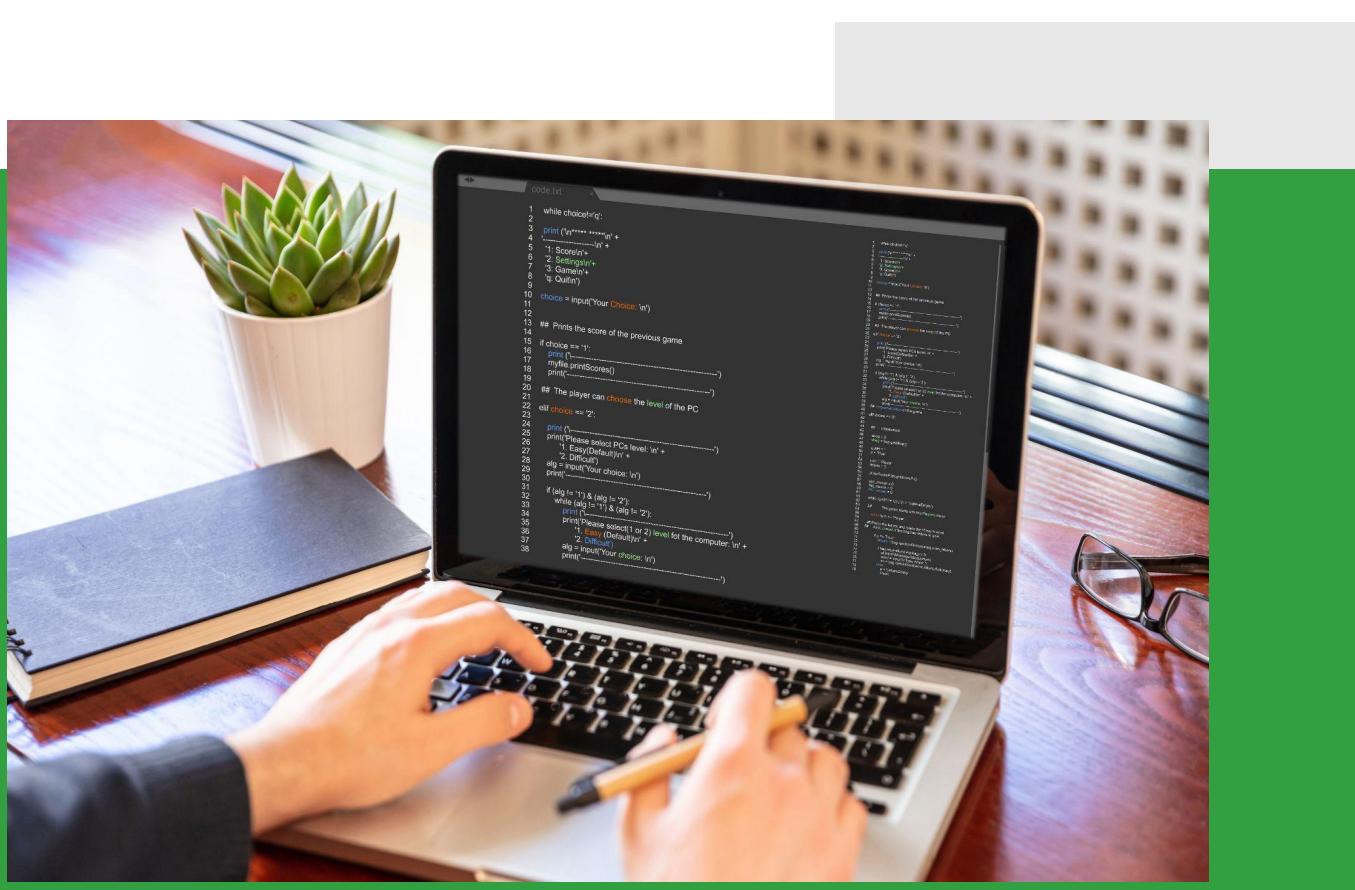
O texto dentro de uma célula `<th>` sempre aparece centralizado, na horizontal, em relação à largura da célula.

Também, cabe destacar que, além dos componentes apontados, uma tabela pode apresentar uma legenda. Essa legenda tem o objetivo, entre outros, de indicar qual a fonte de dados que estão sendo tabulados na página web, por exemplo.

Se uma tabela apresentar uma legenda, o texto dessa legenda deve ser marcado pela tag `<caption>`. A tag `<caption>` é o primeiro elemento que deve aparecer imediatamente após a abertura da tag `<table>`, independentemente de a legenda aparecer antes ou depois da tabela na página da web.

Note que, dentro de uma célula `<td>`, podemos inserir não apenas texto. A célula `<td>` é bastante flexível, e pode comportar, dentro dela, informações como imagens, áudio, vídeo, links e, em última instância, é possível inclusive criar a estrutura de uma tabela completamente nova dentro dessa célula `<td>`.

Confira, a seguir, o código que apresenta a estrutura completa de uma pequena tabela, destinada a mostrar notas de avaliações de alunos fictícios:



</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Tabelas com HTML5 </title>
</head>

<body>
    <h1> Marcação simplificada de uma tabela na linguagem HTML5
    </h1>

    <table>
        <!-- se houver legenda na tabela, ela (caption) deve aparecer
        exatamente neste ponto do código -->
        <caption> Rendimento semestral de alunos - UC PRW1 </caption>

        <tr>
            <!-- aqui, as célula de cabeçalho (título de cada coluna)
        -->
            <th> Nome do aluno </th>
            <th> Nota da primeira avaliação </th>
            <th> Nota da segunda avaliação </th>
        </tr>

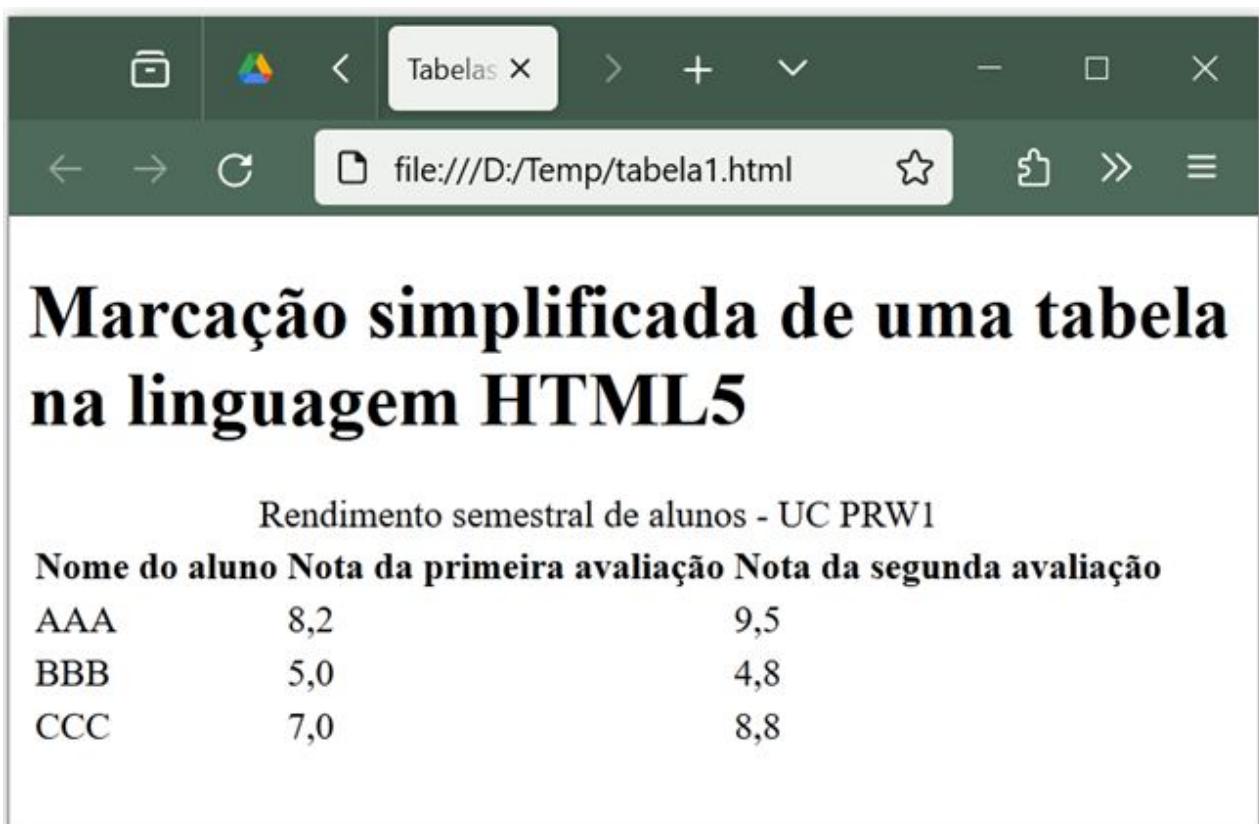
        <!--próxima linha da tabela-->
        <tr>
            <!--célula de dados regulares-->
        <td> AAA </td>
        <td> 8,2 </td>
        <td> 9,5 </td>
    </tr>

    <tr>
        <td> BBB </td>
        <td> 5,0 </td>
        <td> 4,8 </td>
    </tr>

    <tr>
        <td> CCC </td>
        <td> 7,0 </td>
        <td> 8,8 </td>
    </tr>
</table>
</body>
</html>
```

Já a imagem a seguir mostra como o código anterior foi renderizado em um navegador específico.

Código renderizado



The screenshot shows a web browser window with a dark green header bar. The title bar displays the URL "file:///D:/Temp/tabela1.html". The main content area contains a large title in bold black font: "Marcação simplificada de uma tabela na linguagem HTML5". Below the title, there is a subtitle: "Rendimento semestral de alunos - UC PRW1". A table is displayed with the following data:

| Nome do aluno | Nota da primeira avaliação | Nota da segunda avaliação |
|---------------|----------------------------|---------------------------|
| AAA | 8,2 | 9,5 |
| BBB | 5,0 | 4,8 |
| CCC | 7,0 | 8,8 |

Fonte: Elaborada pelo autor (2024).

Observe que, por padrão, o navegador sempre renderiza uma tabela, na página web, sem adicionar nenhuma linha de grade ao redor das células ou da própria tabela. Veremos posteriormente como solucionar esse problema.

Além disso, veja que, na figura anterior, o título de cada coluna (células <th>) apresenta o texto já em negrito e centralizado na horizontal, dentro de cada célula, como mencionamos anteriormente.

Existem outras tags adicionais que complementam a marcação de tabelas, embora as mais relevantes para o presente estudo estejam destacadas nos tópicos anteriores.



Se você desejar mais informações sobre esse tema, acesse os links seguintes:

- marcação de tabelas — primeira parte; [Acesse](#)
- marcação de tabelas — segunda parte; [Acesse](#)
- aprofundando-se no estudo da marcação de tabelas em HTML5. [Acesse](#)

Adiante, você poderá conhecer como funciona a criação de formulários em uma página web. Vamos lá?

9. Criação de formulários em uma página web (etapa 1)

Os formulários, em uma página web, são elementos destinados a coletar dados do usuário e enviar esses dados para algum local, para que possam ser processados.

Em linhas gerais, formulários são uma das maneiras mais comuns de se elaborar a interface gráfica de uma aplicação web. Por meio dessa interface — manipulada pelo navegador, na máquina-cliente —, o usuário interage com o sistema, permitindo que dados sejam recolhidos, enviados e processados por um software sendo executado na máquina servidora. Após esse processamento, os códigos executados na máquina servidora — e que compõem o back-end da aplicação web — elaboram

uma resposta, na forma de uma nova página web. Essa nova página web é enviada de volta ao cliente da aplicação, pelo navegador e, depois disso, todo o ciclo se reinicia.



Para mais informações sobre formulários na linguagem HTML5, visite os links abaixo:

[Acesse](#)

[Acesse](#)

[Acesse](#)

[Acesse](#)

[Acesse](#)

Acompanhe, agora, como funcionam os métodos de envio de dados ao servidor.

Métodos de envio de dados ao servidor

Ao enviar os dados de um formulário, o navegador pode empregar várias formas de transporte desses dados para o servidor, incluindo:



Método get

Os dados fornecidos ao formulário e transportados à máquina servidora para serem processados são codificados como texto comum e, nesse transporte, podem ser facilmente interceptados e visualizados.



Método post

Os dados, antes de serem enviados para processamento, são “embaralhados” — **criptografados** — na máquina-cliente, de forma a dificultar sua descoberta se forem interceptados durante a etapa de transporte.

Note que, além dessa diferença entre os dois métodos de envio de dados, existem algumas outras. Porém, essa, que está relacionada ao uso de **criptografia**, por parte do navegador, é, sem dúvida, uma das mais importantes, haja vista que diz respeito à segurança de dados sensíveis na manipulação de uma aplicação web.

Ao implementar-se a interface gráfica de uma aplicação web envolvendo formulários, o método post, salvo dito contrário, deveria ser sempre o preferido para utilização. Veremos, mais adiante, como indicar ao navegador o uso desse método dentro da página web.

Na sequência, acompanhe os principais controles para formulários.

Principais controles de formulários

O elemento-chave para marcação de formulários é a tag `<form>`, que é uma tag completa. Essa tag, obrigatoriamente, marca o início e o final de todo formulário.



Cyberpedia

Criptografia: é uma técnica antiga, que permite converter textos simples, legíveis por humanos, em textos incompreensíveis, cifrados, que escondem a informação original. É utilizada para preservar a confidencialidade de informações.

A tag `<form>` tem diversos atributos importantes, entre eles, o atributo `method`, que especifica qual método o navegador irá utilizar para transportar esses dados até o servidor, conforme vimos anteriormente. O valor do atributo `method` pode ser `post` ou `get`. Também, a tag `form` deve especificar, por meio de um atributo adequado, qual software, no servidor, irá receber esses dados e processá-los, devolvendo uma resposta ao usuário. Esses elementos serão vistos em um outro momento de nosso curso.

É importante destacar que o navegador diferencia, por meio de tags apropriadas, o tipo de dado que está sendo solicitado ao usuário. Veremos esse aspecto mais detalhadamente a seguir.

Identificadores de campo de dados

Basicamente, sempre que nossa interface web solicitar alguma informação, ela deve indicar claramente ao usuário qual o tipo de informação que ele deve fornecer em cada campo correspondente.

Para isso, utilizamos a tag `<label>`, que admite texto como seu conteúdo e serve como identificador de cada campo. A tag `<label>` é uma tag completa, composta por uma abertura e um fechamento.

Controle para fornecimento de texto em formulários

O texto é o tipo de dado mais comum no preenchimento de um formulário. Para isso, existe a tag `<input>`, com o atributo `type="text"`, que permite que qualquer caractere seja digitado no campo de formulário identificado por esse marcador.

Toda tag `<input>` é vazia, portanto, não tem fechamento.

Controle para fornecimento de números em formulários

Sempre haverá situações em que nossa aplicação web precisará requisitar ao usuário, durante o preenchimento de um formulário, algum dado numérico, por exemplo, seu saldo bancário, o valor de compra de um produto, sua idade etc.

Para essa finalidade, existe a tag `<input>`, com o atributo `type="number"`, que pode ser utilizada.

Cabe mencionar que, por padrão, ao utilizarmos a tag input para a inserção de dados numéricos, o navegador permitirá o fornecimento apenas de números inteiros.

Se quisermos utilizar números com casas decimais, devemos fazer uso do atributo `step`, que especifica o número de casas decimais máximo a ser fornecido. Também, em um campo numérico de formulário, é possível definir um valor mínimo e um valor máximo para o campo. Qualquer número fora dessa faixa não será aceito pelo navegador. Observe um exemplo de linha de código utilizando a tag `input` com os referidos atributos:

`</> Código na área`

```
<input type="number" max="150" min="0" step="0.1">
```

De acordo com essa linha, estamos especificando ao navegador que o campo aceita somente valores numéricos, com, no máximo, uma casa decimal — step="0.1" — dentro da faixa de valores 0 e 150, inclusive.

Controle para fornecimento de datas em formulários

É possível criarmos, em um formulário, um campo específico para o fornecimento de datas. Ao utilizar esse campo, o usuário pode digitar manualmente uma data. Ou, se preferir, pode escolher uma data, por meio de um pequeno calendário que o navegador apresenta, automaticamente, ao clicar sobre o campo.



A tag para preenchimento de datas é a tag `<input>`, com o atributo `type="date"`.

Esta é a primeira parte dos componentes de um formulário em uma página web. No próximo tópico, continuaremos com o estudo de outros elementos de formulários.

10. Criação de formulários em uma página web (etapa 2)

Conforme mencionamos anteriormente, neste tópico daremos continuidade ao estudo de outros controles importantes, na linguagem HTML, para a criação de formulários em uma página web. Vamos a eles!

Controle para seleção de um único valor, dentre vários valores — radio

Quando criamos um formulário para a interface gráfica de uma aplicação web, será necessário, em várias situações, oferecermos ao usuário um conjunto de alternativas, dentre as quais ele poderá **optar por somente uma**. Por exemplo, se quiséssemos saber qual a sua titulação máxima, poderíamos criar um controle de formulário que apresentasse as seguintes opções:

- ensino médio
- curso superior completo
- mestrado
- doutorado

Nesse cenário, o usuário poderia escolher apenas um único item, relacionado ao seu mais alto nível de formação acadêmica.

Para atingirmos esse objetivo, a linguagem HTML5 fornece a tag `<input>`, com o atributo `type="radio"`. Também, existe um atributo chamado `value`, que é utilizado para enviar alguma informação à máquina servidora, de acordo com o item escolhido pelo usuário.

Controle para seleção de múltiplos itens — checkbox

Ao contrário do elemento anterior, quando estamos criando um formulário para a interface gráfica de uma aplicação web, pode haver a necessidade, em várias situações, de oferecermos ao usuário um conjunto de alternativas, dentre as quais ele poderá optar **por várias delas simultaneamente**. Por exemplo, se quiséssemos saber quais as linguagens de desenvolvimento web que o usuário domina, poderíamos criar um controle de formulário que apresentasse as seguintes opções:

- PHP;
- HTML;
- CSS;
- JavaScript.

Neste caso, esse controle de formulário permite que o usuário selecione **mais de uma opção ao mesmo tempo**, possivelmente, marcando todas elas, se for o caso.

A tag que usamos para esse tipo de construção é a tag `<input>`, com o atributo `type="checkbox"`. Também, existe o atributo `value`, que envia as informações selecionadas pelo usuário, no formulário, à máquina servidora, para que estas informações sejam processadas posteriormente.

Controle para seleção de um único item — select

Semelhante ao comportamento de elementos do tipo botão de rádio, conforme visto anteriormente, também temos a tag `<select>`. De forma simplificada, ela permite que o usuário selecione apenas uma única opção, dentre várias à sua escolha.

Esta tag é marcada com `<select>`. Por sua vez, para indicarmos cada item dentro da seleção, que pode ser escolhido pelo usuário, usamos a tag `<option>`. Ambas

são tags completas, com abertura e fechamento. A tag `<option>` sempre deve aparecer dentro de `<select>`.

Controle para submissão de dados ao servidor

Depois de preencher todos os campos de dados de um formulário, o usuário deve sinalizar ao navegador que é hora de enviar esses dados para algum local, para serem processados e tratados posteriormente.



Temos vários controles com essa finalidade, na forma de botões de envio de dados. O mais utilizado é o botão dado pela tag `<button>`. É uma tag completa, com abertura e fechamento, cujo conteúdo é o próprio texto do botão. Ao ser pressionado, no formulário, esse botão obriga o navegador a coletar todos os campos de dados e enviá-los para processamento adequado. Na nossa estrutura cliente-servidor, esse local para onde os dados são submetidos é a máquina servidora, que executará código apropriado. Esse código se encarregará de processar esses dados e devolver uma resposta à requisição disparada pelo usuário.

Após revisarmos os principais elementos de formulários até agora, é hora de conferir como podemos agrupar todos esses elementos em um arquivo HTML para construir um formulário web funcional. Observe o código seguinte:

</> Código na área

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Formulários </title>
</head>

<body>
    <h1> Principais controles de formulário na linguagem HTML5
    </h1>

    <form method="post">
        <label> Seu nome: </label>
        <input type="text" > <br> <br>

        <label> Saldo atual de seu cartão de crédito: </label>
        <input type="number" step="0.01" > <br> <br>

        <label> Data de validade de seu cartão: </label>
        <input type="date" > <br> <br>

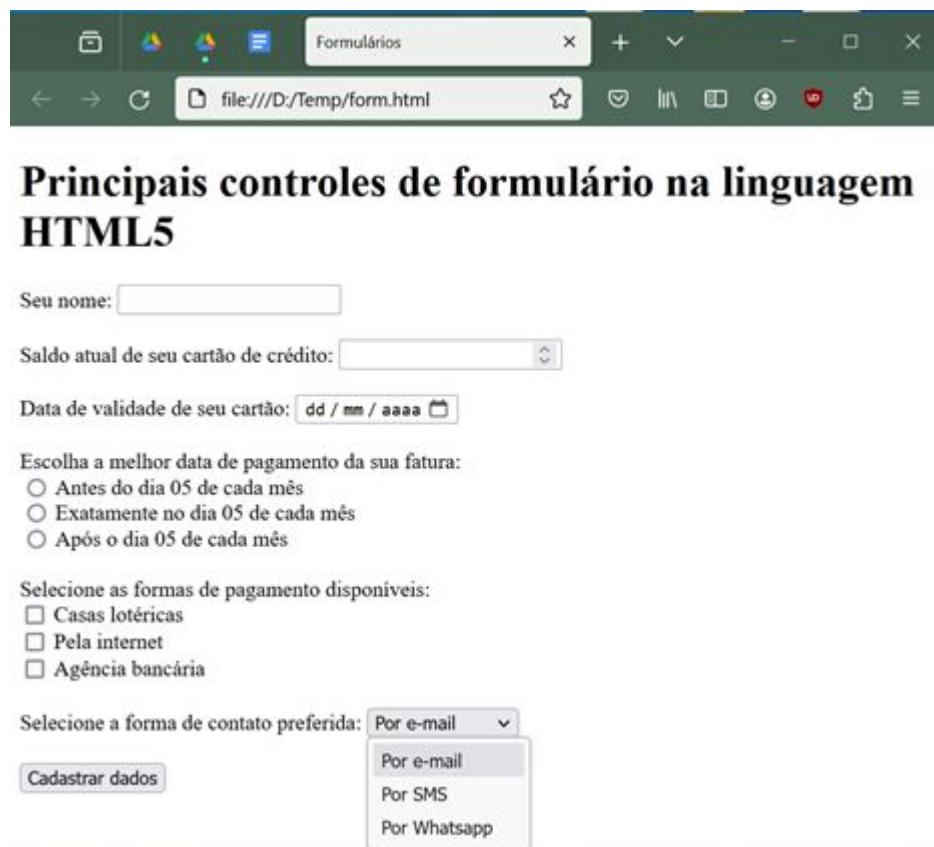
        <label> Escolha a melhor data de pagamento da sua fatura:
        </label> <br>
        <input type="radio" value="0" > Antes do dia 05 de cada mês
        <br>
        <input type="radio" value="1" > Exatamente no dia 05 de cada
        mês <br>
        <input type="radio" value="2" > Após o dia 05 de cada mês <br>
        <br>
        <label> Selecione as formas de pagamento disponíveis: </label>
        <br>
        <input type="checkbox" value="0" > Casas lotéricas <br>
        <input type="checkbox" value="1" > Pela internet <br>
        <input type="checkbox" value="2" > Agência bancária <br> <br>

        <label> Selecione a forma de contato preferida: </label>
        <select>
            <option> Por e-mail </option>
            <option> Por SMS </option>
            <option> Por Whatsapp </option>
        </select> <br> <br>

        <button> Cadastrar dados </button>
    </form>
</body>
</html>
```

Por fim, podemos renderizar o código anterior em um navegador. O resultado é dado pela figura a seguir:

Código renderizado



The screenshot shows a web browser window titled "Formulários" with the URL "file:///D:/Temp/form.html". The page content is as follows:

Principais controles de formulário na linguagem HTML5

Seu nome:

Saldo atual de seu cartão de crédito:

Data de validade de seu cartão: dd / mm / aaaa

Escolha a melhor data de pagamento da sua fatura:

Antes do dia 05 de cada mês
 Exatamente no dia 05 de cada mês
 Após o dia 05 de cada mês

Selecione as formas de pagamento disponíveis:

Casas lotéricas
 Pela internet
 Agência bancária

Selecione a forma de contato preferida:

Fonte: Elaborada pelo autor (2024).

Depois de praticar esse desafio, você já estará pronto para prosseguir com nosso estudo. Porém, como você pode verificar nas páginas renderizadas anteriormente, elas não oferecem nenhum atrativo visual, isto é, não há cores nas páginas, não há sequer texto centralizado, não há nenhuma formatação.

Na próxima etapa, faremos justamente isso: aprenderemos como fazer o navegador aplicar estilos aos elementos de nossa página web, por meio de uma linguagem conhecida como CSS, criada especialmente para modificar o aspecto visual dos elementos de uma página web.



11. Introdução às Folhas de Estilo em Cascata - CSS

O termo CSS, ou Folhas de Estilo em Cascata, é uma abreviatura do termo em inglês Cascading Style Sheet.

De forma simplificada, Folha de Estilos em Cascata é uma técnica que visa à aplicação de formatação e de estilos a elementos de uma página web, como formatação de texto, fontes, cores, posicionamento de elementos, animações etc.

Finalidade das CSS

As **Folhas de Estilo em Cascata**, mais conhecidas como CSS, podem ser entendidas como uma linguagem de estilos, que apresenta as seguintes finalidades:

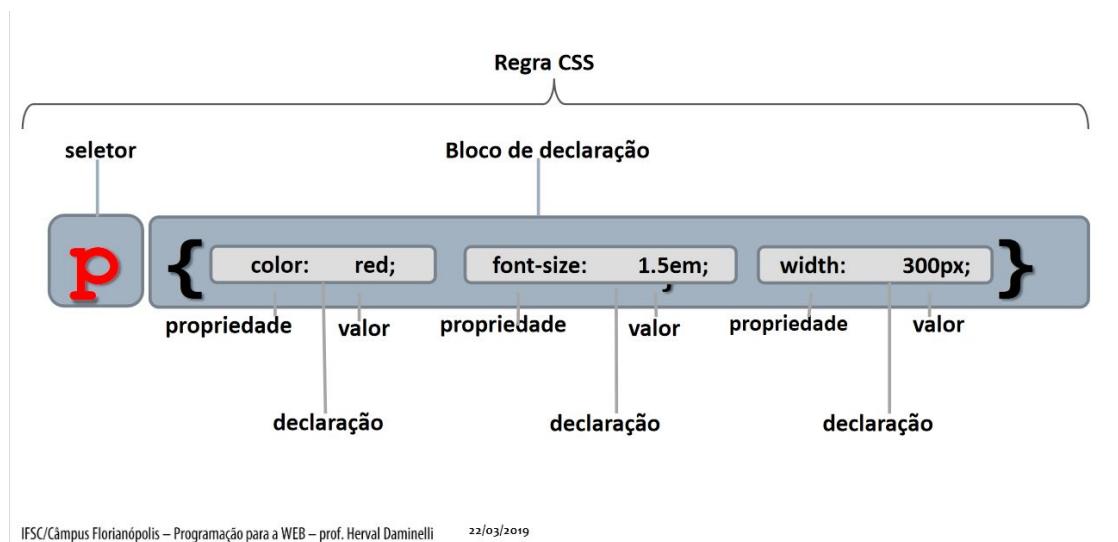
- Retirar do documento HTML toda e qualquer declaração que tenha por objetivo estilizar os elementos da página web.
- Separar os aspectos de formatação dos elementos de uma página web (CSS) daqueles relacionados ao conteúdo estrutural da página (HTML).

Vamos conhecer a regra CSS?

Regra CSS

Entende-se por unidade básica de Folhas de Estilos como a menor porção de código capaz de estilizar (formatar) determinado elemento HTML de um documento web. Acompanhe, no esquema a seguir, os principais componentes de uma regra CSS.

Estrutura de uma regra CSS



IFSC/Câmpus Florianópolis – Programação para a WEB – prof. Herval Daminelli 22/03/2019

Fonte: Elaborada pelo autor (2024).

O conjunto de várias regras CSS, agrupadas em um arquivo específico, compõem o que chamamos de Folha de Estilos. Conheça outras informações sobre as regras CSS a seguir.

Definição dos elementos de uma regra CSS

De acordo com a figura **Estrutura de uma regra CSS**, podemos destacar os seguintes elementos componentes de uma regra CSS:

Seletor

Define qual tag HTML na página web está sendo estilizada.

Chaves {}

Definem o início e o fim de uma regra CSS.

Propriedade

Define a característica que está sendo estilizada no elemento da página web.

Valor

Define o valor, por exemplo, uma cor, uma família de fonte, um comprimento de um elemento, um tipo de espessura etc. para a propriedade que sendo estilizada.

Declaração

Um par composto por uma propriedade e um valor.

Bloco de declaração

Um conjunto de várias declarações.

Regra CSS

A junção de um seletor com um bloco de declaração.

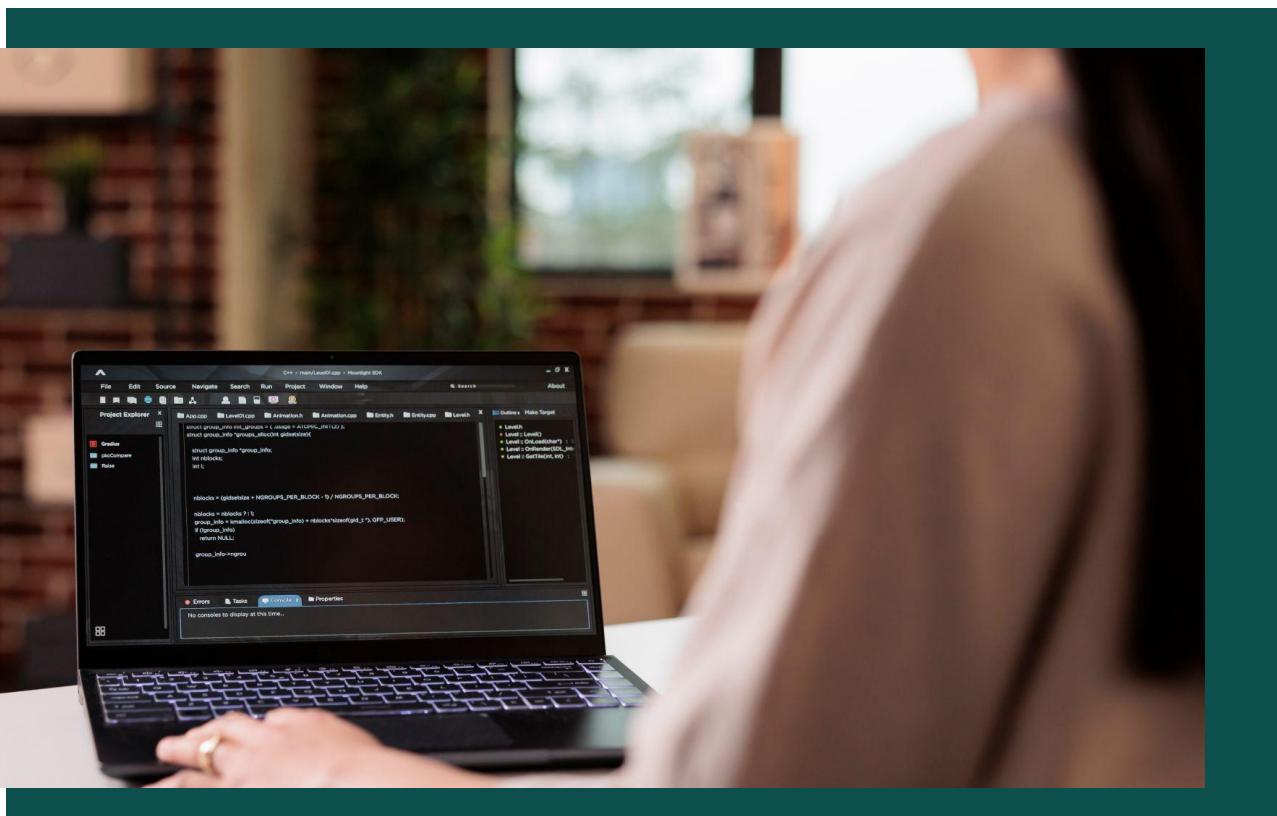
Observe dois aspectos importantíssimos ao definirmos uma regra CSS:

**Toda declaração é encerrada com um ponto-e-vírgula
(;); um bloco de declaração deve, sempre, estar
inserido dentro de um par de chaves {}.**

A compreensão dos elementos básicos do CSS, como o seletor, as chaves, a propriedade e o valor, é fundamental para a criação eficaz de estilos em uma página web. Ao entender como esses componentes se relacionam, torna-se possível definir regras CSS de forma clara e concisa, garantindo uma apresentação visual coerente e atrativa para os usuários.

Comentários em CSS

Já tratamos, em uma etapa anterior, sobre a importância de se documentar internamente qualquer código em desenvolvimento. Vimos que a utilização de comentário para essa finalidade é considerada uma boa prática de desenvolvimento.



Neste sentido, da mesma forma que comentamos declarações HTML com a tag `<!-- -->`, também podemos fazê-lo dentro de um arquivo CSS, por meio dos símbolos `/*` e `*/`. Qualquer trecho de texto escrito dentro desses símbolos é desconsiderado pelo navegador e tratado como documentação interna de nossa aplicação. Veremos a aplicação dessa técnica mais adiante em nosso estudo.

Seletores do tipo classe e id

Você pôde compreender, ao longo de nosso estudo, que frequentemente desejamos que o navegador formate em algumas situações, apenas determinado elemento da página web, e deixe os demais elementos do mesmo tipo intocados.

Por exemplo: não raro, podemos nos deparar com a situação em que temos cinco parágrafos de texto do documento HTML, mas queremos trocar a cor da fonte somente do primeiro parágrafo, sem que esta mudança afete os demais. Para isso, precisamos indicar ao navegador qual parágrafo deve ser formatado, de forma a separá-lo dos demais. Para isso usamos os seletores **class** ou **id**.

Cyberpedia

Class e **id**: são atributos que podem ser introduzidos dentro de qualquer tag na página web, com o propósito de estilizar elementos.

A grande diferença entre esses dois seletores é que o nome dado a um atributo class pode ser repetido, isto é, pode aparecer mais de uma vez na página web, ao passo que o atributo id é único e não deve se repetir em nenhuma tag HTML.

Veja, agora, como estilizar uma página web por meio da inserção de código CSS.

Vinculação de uma folha de estilos a um documento web

Em linhas gerais, podemos definir três formas de inserção de código CSS em uma página web, para que ela seja estilizada. São elas:

- CSS inline;
- CSS incorporado;
- CSS externo.

Os formatos inline e incorporado não aproveitam todas as vantagens que as folhas de estilo podem trazer à formatação de uma página web e, portanto, não são muito utilizados. Desta forma, não serão contemplados no presente estudo.

Dedicaremos especial atenção ao CSS externo, que aproveita todos os benefícios e potencialidades que essa técnica de formatação pode trazer no desenvolvimento de uma aplicação web.

Por meio desse tipo de vinculação, é perfeitamente possível termos um único arquivo externo CSS sendo utilizado para formatar várias páginas web.

CSS externo

A vinculação de uma folha de estilos externa à página web se dá por meio da tag `<link>`. Essa tag é vazia, portanto, não apresenta fechamento. A tag `<link>` tem as seguintes características:

- Só pode ser inserida na seção `<head>` do documento.
- Deve apresentar os atributos `rel` e `href`.
- O valor do atributo `rel` é sempre o termo “stylesheet”. Esse valor indica ao navegador que o conteúdo do arquivo externo é uma folha de estilos.
- O valor do atributo `href` indica ao navegador o nome do arquivo de estilos externo, que deve ter sempre a extensão `.css`; e o local em que o navegador irá encontrar o arquivo de estilos (a pasta ou diretório, na estrutura de pastas de nossa aplicação web, em que o arquivo CSS está armazenado).

A seguir, você confere um trecho de código subdividido em dois arquivos diferentes (arquivo1.html e formata-paragrafo.css): um arquivo HTML, que representa a página web, e um arquivo externo css, que contém a folha de estilos. Note o uso dos seletores class e id na folha de estilos, e o correspondente atributo no arquivo HTML.

</> Código na área

arquivo1.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Usando CSS </title>
    <!-- chamada do arquivo externo CSS -->
    <link rel="stylesheet" href="formata-paragrafo.css" >
</head>

<body>
    <h1> Usando CSS externo para formatação da fonte de
diversos parágrafos </h1>
    <p class="troca-cor-fonte" > Utilização de atributo
class </p>
    <p class="troca-cor-fonte" > Utilização de atributo
class </p>
    <p> Parágrafo sem class ou id: não sofreu formatação
</p>
    <p> Parágrafo sem class ou id: não sofreu formatação
</p>

    <p id="troca-tipo-fonte" > Utilização de atributo id
</p>
</body>
</html>
```

Repare que, no CSS, o seletor **class** é sempre representado pelo símbolo ponto-final (**.**). O seletor **id** é sempre marcado com o símbolo da cerquilha (**#**). Observe, também, como inserimos o arquivo externo css na página web, por meio da tag **<link>**, usando os atributos **rel** e **href**:

</> Código na área

arquivo formata-paragrafo.css

```
/*usando seletor do tipo class para formatar somente os
parágrafos desejados*/
p.troca-cor-fonte {
    color: green; /*propriedade CSS para trocar cor da
    fonte de um texto dentro do parágrafo que contém a class
    troca-cor-fonte*/
}

p#troca-tipo-fonte {
    font-family: candara; /*propriedade CSS para trocar o
    tipo de fonte do texto de um parágrafo que contenha o
    atributo id troca-tipo-fonte*/
}
```

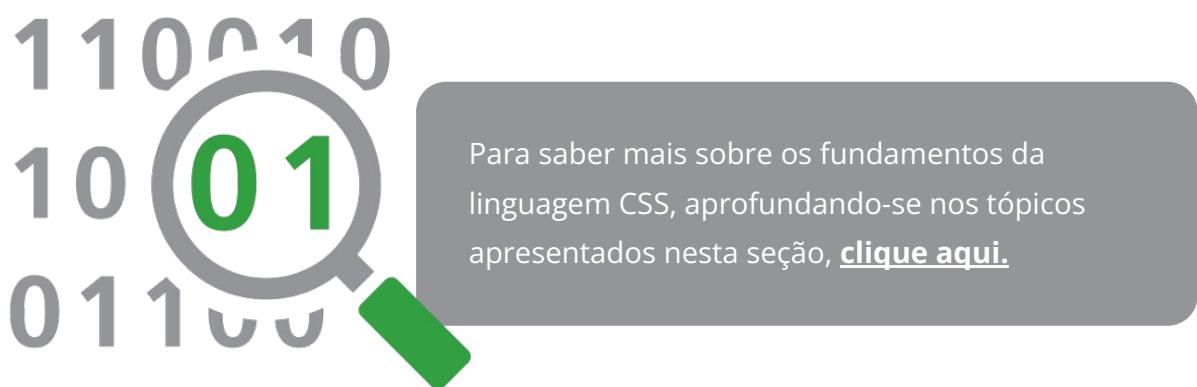
A seguir, observe o resultado da formatação desses parágrafos na página web e qual aspecto visual ela apresenta quando exibida em um navegador:

Renderização de uma página web com CSS externo



Fonte: Elaborada pelo autor (2024).

Neste ponto de nosso estudo, surge uma dúvida: o que acontece quando temos elementos diferentes que precisam receber a mesma formatação? Será que é possível, em uma folha de estilos, aplicarmos a mesma formatação para vários seletores diferentes, ao mesmo tempo? Se sim, como a regra CSS deveria ser escrita para atingirmos esse objetivo?

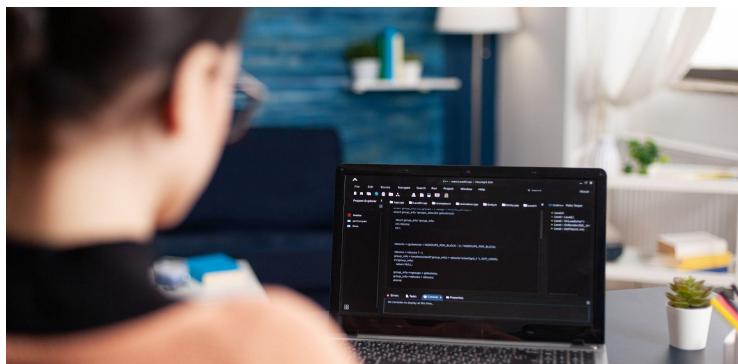


Acompanhe, a seguir, quais são as unidades de medidas que podem ser utilizadas em CSS.

Unidades de medida em CSS

Frequentemente, utilizaremos, em uma folha de estilos, propriedades CSS que exigirão a definição de um número e de uma medida para esse número. Por exemplo: definir o tamanho de uma fonte, definir a largura de um parágrafo, definir um espaço separando a linha de cima da linha de baixo de um texto etc.

Nesta situação, devemos utilizar uma unidade de medida válida no CSS. Existem várias, dentre as quais, podemos citar:



- pixel (px);
- em;
- rem;
- centímetro (cm);
- milímetro (mm);
- percentual (%);

Dentre as unidades de medida mencionadas, uma das mais utilizadas é **pixel**. Ela equivale ao tamanho de um ponto luminoso usado pela placa gráfica do dispositivo computacional para formar uma imagem. A seguir, veja o exemplo de uma regra CSS, usando uma unidade de medida:

</> Código na área

```
p {  
    width: 350px;  
}
```

Observe que, no código anterior, indicamos ao navegador que todos os parágrafos do documento web deverão ter uma largura igual a 350 pixels.



Quer algumas informações mais detalhadas sobre unidades de medida em CSS?

[Confira aqui.](#)

A respeito da formatação de texto com CSS, você pode conferir a seguir.

12. Formatação de texto básica com CSS

Para aplicação de estilos básicos a um elemento que contenha texto na página web, podemos mencionar as seguintes propriedades CSS:



font-size: define o tamanho da fonte a ser utilizada pelo navegador, ao exibir o texto sendo formatado.



font-weight: aplica negrito ao texto formatado.



font-family: permite indicar qual o tipo de fonte utilizado pelo navegador para exibir o texto em questão.



color: define a cor do texto a ser aplicada. O valor da cor é o nome dela, em inglês.

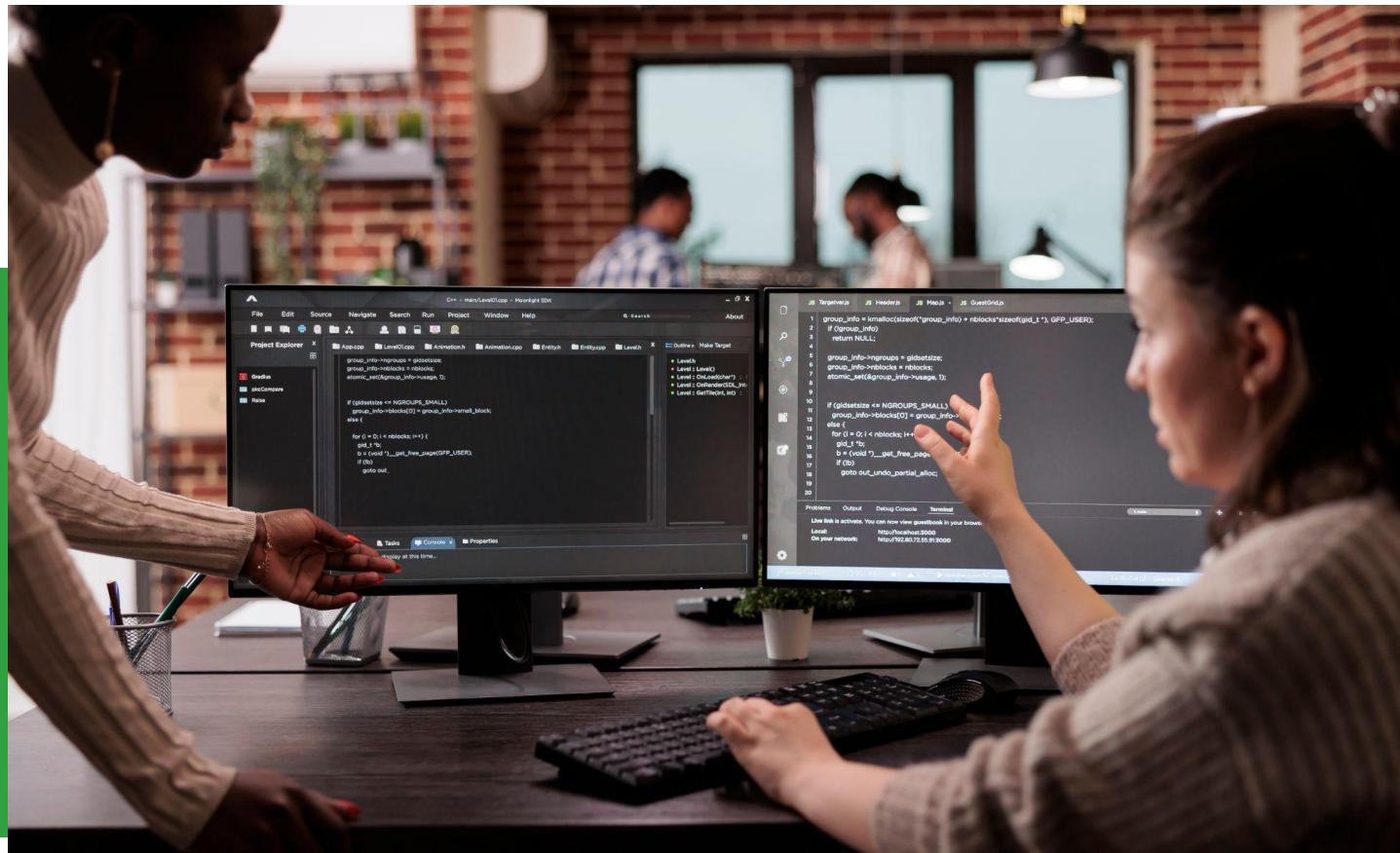


text-decoration: permite o sublinhado do texto.



font-style: permite a aplicação de itálico ao texto.

Existem diversas outras propriedades das folhas de estilos relacionadas à formatação de textos. Pela impossibilidade de, no presente estudo, abordarmos todas elas, destacamos somente estas, que são as mais relevantes para formatação em CSS.



A seguir, você pode observar a codificação de uma folha de estilos para formatar parágrafos de uma página web:

</> Código na área

arquivo texto.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Formatação de texto </title>
    <link rel="stylesheet" href="formata-texto.css" >
</head>

<body>
    <h1> Formatação básica de texto com CSS </h1>

    <p class="texto1"> Lorem ipsum dolor sit, amet consectetur adipisicing elit. Blanditiis temporibus facilis laboriosam reprehenderit? Deleniti cumque, ullam, eligendi amet explicabo dolorum expedita non natus voluptas neque ratione accusantium dolor perspiciatis sed. Lorem ipsum dolor sit amet, consectetur adipisicing elit. At debitis laborum a ducimus! Eos rem ducimus perspiciatis, cum blanditiis quia ratione nobis! Optio commodi quisquam autem amet porro ipsa sequi? </p>

    <p class="texto2"> Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eveniet quaerat dolorum facere expedita, ratione reiciendis ab autem incidunt, nulla corporis veritatis eligendi accusamus odio ipsum fugiat eos quasi itaque ipsam! Lorem ipsum dolor sit amet consectetur adipisicing elit. Quisquam impedit, possimus delectus facilis vel dolor ullam cumque eaque accusamus modi quibusdam animi est sint, maiores earum aperiam iure. Facilis, unde. </p>
</body>
</html>
```

Agora, importando uma fonte externa ao sistema:

</> Código na área

arquivo formata-texto.css

```
/*importando uma fonte externa ao sistema*/
@import
url('https://fonts.googleapis.com/css2?family=Roboto+Slab&display=swap');

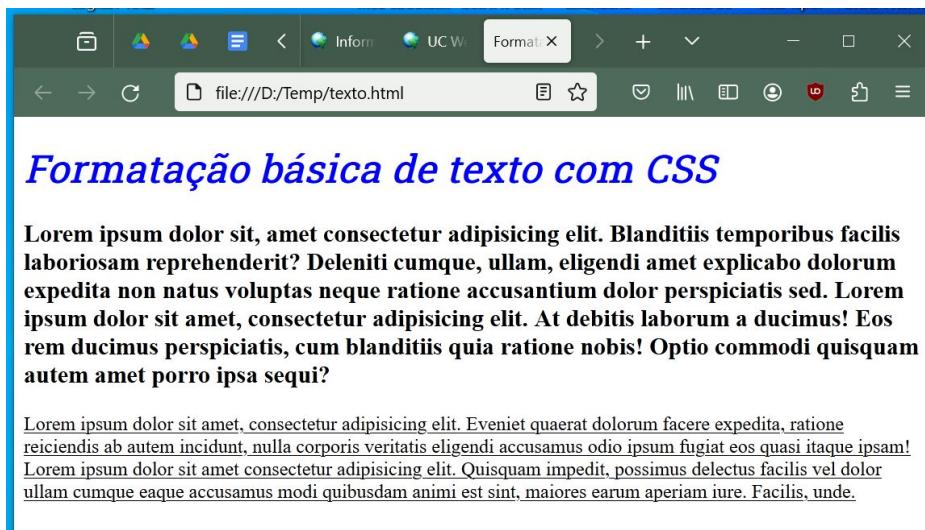
h1 {
    font-family: "Roboto Slab"; /*tipo da fonte para o
cabeçalho h1*/
    color: blue;             /* cor da fonte*/
    font-style: italic;       /* itálico */
}

p.texto1 {
    font-size: 20px;          /*tamanho da fonte para o
primeiro parágrafo*/
    font-weight: bold;         /*negrito para o texto do
primeiro parágrafo*/
}

p.texto2 {
    text-decoration: underline; /*texto sublinhado para o
segundo parágrafo*/
}
```

A imagem a seguir mostra a renderização de uma página web com estilos para formatação de texto.

Estilos para formatação de texto



Fonte: Elaborada pelo autor (2024).

Uma vez compreendido como realizar essa aplicação, é o momento de você entender como fazer a formatação de tabelas a partir do CSS.

13. Formatação de tabelas

Como vimos anteriormente sobre tabelas com HTML5, por padrão, ao renderizar uma tabela, o navegador não exibe nenhuma linha de borda, quer seja ao redor da própria tabela, quer seja ao redor de cada célula.

Por isso, para que os dados tabulados possam ser lidos de forma eficiente pelo usuário, é comum utilizarmos folhas de estilo em cascata e algumas propriedades básicas do CSS para a aplicação de alguns estilos que favoreçam tal leitura.

Começamos pela propriedade border, que exibe um retângulo ao redor de qualquer elemento na página web. A propriedade border necessita que especifiquemos alguns valores, como:



- a espessura da borda;
- o estilo da borda (solid, dashed, double, groove etc.); e
- a cor da borda.

Também, podemos definir uma altura e uma largura para o tamanho da tabela, com as propriedades **height** e **width**, respectivamente.

Se você quiser colocar uma cor de fundo (ou para cada célula, ou para a tabela como um todo), pode usar a propriedade background-color e especificar uma cor qualquer.

Finalmente, podemos manipular a posição do texto em cada célula, alinhando-o ao centro, com a propriedade **text-align: center**.

A seguir, você visualiza o código de dois arquivos em separado. O primeiro arquivo HTML mostra a marcação de uma tabela básica.

</> Código na área

arquivo tabela2.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Tabelas com HTML5 e CSS3 </title>
    <link rel="stylesheet" href="formata-tabela.css" >
</head>

<body>
    <h1> Tabela na linguagem HTML5 e sua formatação básica
com CSS </h1>

    <table>
        <caption> Rendimento semestral de alunos - UC PRW1
</caption>

        <tr>
            <th> Nome do aluno </th>
            <th> Nota da primeira avaliação </th>
            <th> Nota da segunda avaliação </th>
        </tr>

        <tr>
            <td> AAA </td>
            <td> 8,2 </td>
            <td> 9,5 </td>
        </tr>
        <tr>
            <td> BBB </td>
            <td> 5,0 </td>
            <td> 4,8 </td>
        </tr>

        <tr>
            <td> CCC </td>
            <td> 7,0 </td>
            <td> 8,8 </td>
        </tr>
    </table>
</body>
</html>
```

Este segundo código, em CSS, mostra a aplicação de propriedades básicas, conforme descrito anteriormente, para formatação da referida tabela:

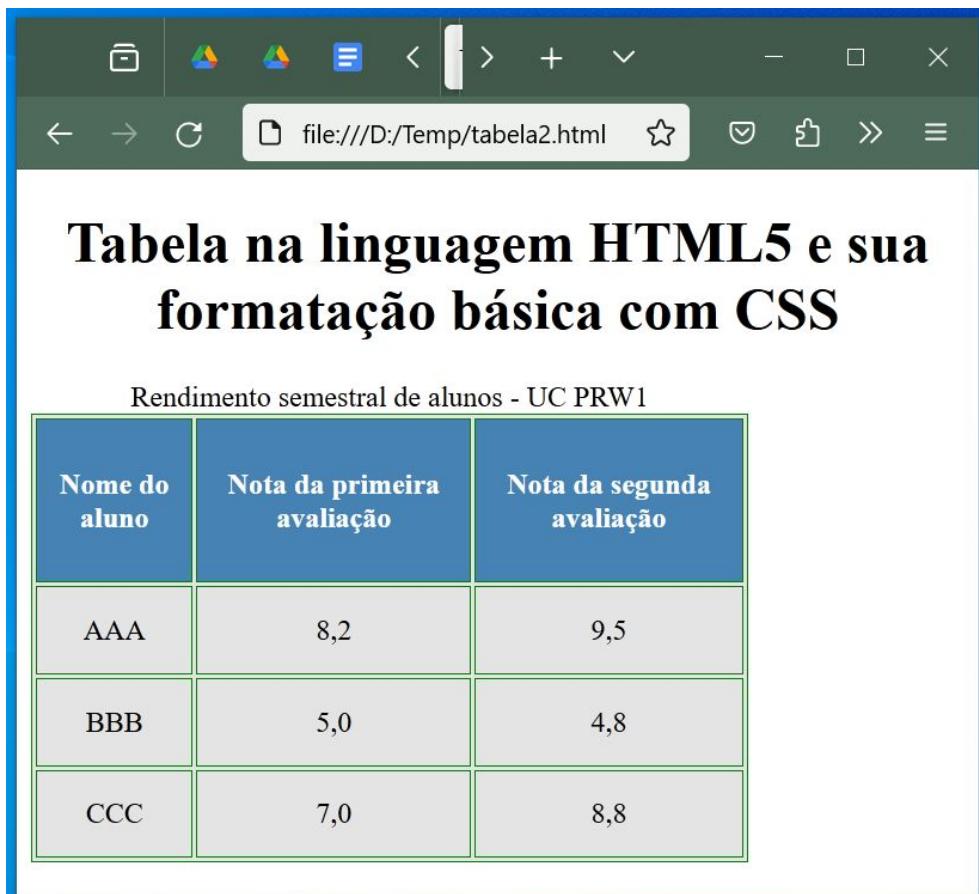
</> Código na área

arquivo formata-tabela.css

```
h1 {  
    text-align: center; /*alinhamento do texto do cabeçalho  
h1 na horizontal*/  
}  
  
table {  
    border: 1px solid green;           /*definindo uma borda ao  
redor da tabela*/  
    width: 400px;                   /*definindo a largura da  
tabela*/  
    height: 250px;                 /*definindo a altura da  
tabela*/  
    background-color: #e4e4e4;        /*definindo uma cor de  
fundo para a tabela - usando o sistema hexadecimal para  
definir uma cor*/  
}  
  
th, td {  
    border: 1px solid green;         /*definindo uma borda,  
simultaneamente, ao redor de cada célula td e th*/  
}  
  
th {  
    background-color: rgb(70,130,180); /*definindo a cor de  
fundo das células de cabeçalho da tabela. Aqui, usamos  
um outro método para definir a cor, conhecido como  
método rgb - vermelho - verde - azul*/  
    color: white;                /*mudando a cor da fonte*/  
}  
  
td {  
    text-align: center;   /*por padrão, o texto dentro de  
uma célula de dados td é sempre alinhado à esquerda.  
Aqui, solicitamos ao navegador o texto centralizado*/  
}
```

Na sequência, apresentamos uma figura mostrando como o navegador renderizará a tabela anterior, após a aplicação da folha de estilos constante do arquivo formata-tabela.css:

Aspecto visual de uma tabela com aplicação de estilos em CSS



A screenshot of a web browser window displaying a table. The browser's title bar shows "file:///D:/Temp/tabela2.html". The main content area has a dark background with white text. At the top, there is a heading: "Tabela na linguagem HTML5 e sua formatação básica com CSS". Below the heading, a caption reads "Rendimento semestral de alunos - UC PRW1". The table has three columns: "Nome do aluno", "Nota da primeira avaliação", and "Nota da segunda avaliação". It contains three rows of data: AAA with notes 8,2 and 9,5; BBB with notes 5,0 and 4,8; and CCC with notes 7,0 and 8,8.

| Nome do aluno | Nota da primeira avaliação | Nota da segunda avaliação |
|---------------|----------------------------|---------------------------|
| AAA | 8,2 | 9,5 |
| BBB | 5,0 | 4,8 |
| CCC | 7,0 | 8,8 |

Fonte: Elaborada pelo autor (2024).

E como podemos utilizar imagens de fundo? Acompanhe a seguir!

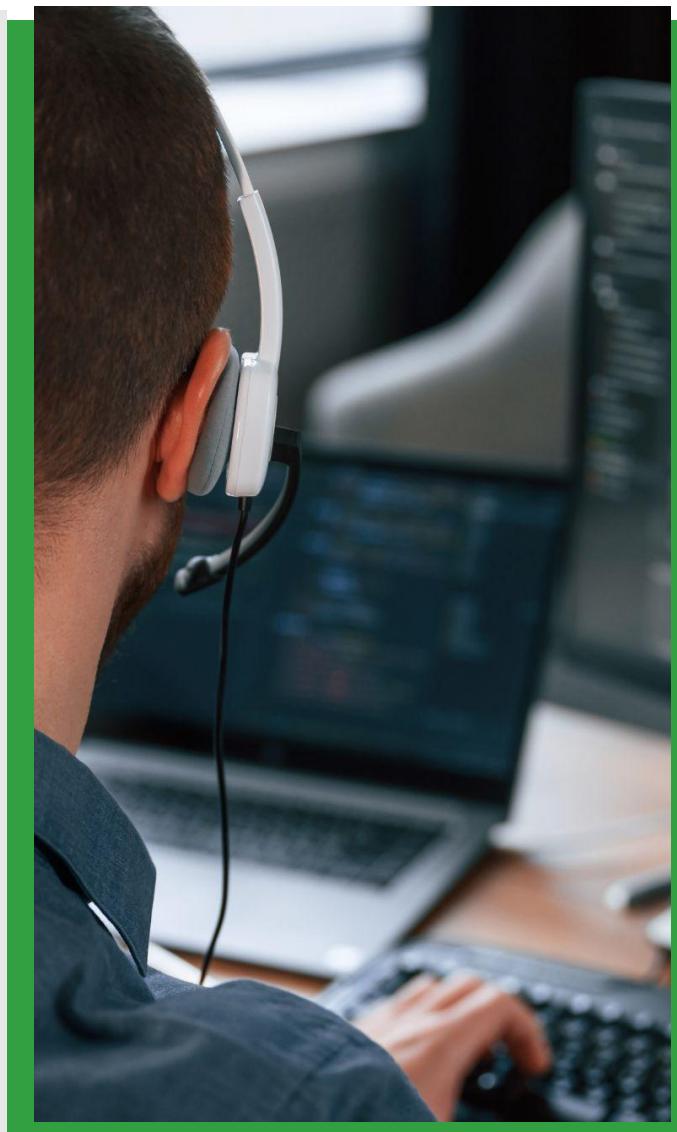
14. Usando imagens de fundo com CSS

As folhas de estilo em cascata permitem que o navegador aplique uma imagem de fundo a qualquer elemento que possa ser visualizado na página web.

Em relação ao suporte que os modernos navegadores oferecem, particularmente quase qualquer tipo de imagem pode ser utilizado. Desta forma, para a escrita deste tutorial, os formatos mais utilizados são as imagens do tipo **jpg** e **png**.

Observe que todo elemento HTML em uma página web tem uma determinada altura e uma largura, que são aplicadas, por padrão, pelo navegador, mas que podem ser alterados pelas respectivas propriedades CSS, como vimos anteriormente. Por isso, ao inserirmos uma imagem de fundo, o navegador pode adotar diferentes critérios de estilização:

- se o tamanho da imagem de fundo for maior que a altura e a largura definidas para o elemento, a imagem de fundo se apresentará truncada, e partes dela ficarão ocultas;
- se a imagem de fundo tem um tamanho menor que o fundo da tag a ser coberta, o navegador irá repetir a imagem, tanto na horizontal, quanto na vertical, se houver espaço.



Verifique, a seguir, quais as propriedades CSS mais utilizadas quando tratamos da aplicação de imagens de fundo em um documento web:

- **Background-image:** url (uma-imagem-local ou uma-imagem-remota). Essa propriedade permite especificar qual imagem o navegador utilizará como fundo do elemento.
- **Background-repeat:** no-repeat. A imagem não será repetida no fundo do elemento, se ela tiver um tamanho menor que a altura e largura da tag html. Neste caso, o navegador sempre posiciona o início da imagem no canto superior esquerdo da caixa que rodeia o elemento HTML.
- **Background-repeat:** repeat-x. A imagem será repetida, no fundo do elemento, somente na horizontal.
- **Background-repeat:** repeat-y. A imagem será repetida somente na vertical (eixo y).
- **Background-position:** 100 px 200 px. Permite especificar a posição exata (com medida em pixels, em mm, em rem etc.) na direção horizontal e vertical, em que o navegador começará a mostrar a imagem de fundo. Neste exemplo dado, o navegador medirá uma distância de 100 pixels a partir do lado esquerdo do elemento na página web e uma distância de 200 pixels a partir do topo do elemento. Na interseção dessas duas distâncias é que o navegador começará a desenhar a imagem de fundo.
- **Background-size:** cover. Com esta propriedade, o navegador tenta redimensionar automaticamente a imagem para que ela cubra exatamente o tamanho do elemento na página web. Dependendo da imagem de fundo e da altura e largura da tag que estiver sendo estilizada, essa propriedade pode deformar a imagem de fundo, fazendo com que ela apareça muito “espremida” ou muito “esticada”.

Na sequência, você pode observar o código utilizado para aplicarmos imagem de fundo a uma tabela e ao cabeçalho da página, vistos em exemplos anteriores:

</> Código na área

arquivo tabela3.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Tabelas com HTML5 e CSS3 </title>
    <link rel="stylesheet" href="formata-tabela.css" >
</head>

<body>
    <h1> Aplicação de imagens de fundo com CSS3 </h1>

    <table>
        <caption> Rendimento semestral de alunos - UC PRW1
        </caption>

        <tr>
            <th> Nome do aluno </th>
            <th> Nota da primeira avaliação </th>
            <th> Nota da segunda avaliação </th>
        </tr>

        <tr>
            <td> AAA </td>
            <td> 8,2 </td>
            <td> 9,5 </td>
        </tr>

        <tr>
            <td> BBB </td>
            <td> 5,0 </td>
            <td> 4,8 </td>
        </tr>

        <tr>
            <td> CCC </td>
            <td> 7,0 </td>
            <td> 8,8 </td>
        </tr>
    </table>
</body>
</html>
```

Sabemos que a página HTML acima, renderizada pelo navegador, apresenta a estrutura da tabela sem nenhuma aplicação de estilo.

Portanto, se desejarmos que o navegador altere a formatação básica da tabela, como imagens de fundo, bordas, formatação de texto, estilos de fontes, cores e espaçamentos entre elementos, devemos criar uma folha de estilos em CSS, cujo código você confere a seguir:

</> Código na área

arquivo formata-tabela.css

```
h1 {  
    text-align: center;  
    background-image: url(imagem-local.jpg); /*aqui, estamos  
    indicando ao navegador para inserir uma imagem de fundo na  
    área do cabeçalho h1. Esta imagem está guardada, localmente,  
    em nossa aplicação web, junto com o arquivo html*/  
    border: 1px solid rgb(117, 175, 236); /*"caixa" ao redor do  
    conteúdo da tag h1*/  
}  
  
table {  
    border: 1px solid green;  
    width: 600px;  
    height: 450px;  
    background-image:  
        url(https://coisasdadoris.com.br/cdn/shop/products/papel-de-parede-fundo-do-mar.jpg); /*usando  
    link de uma imagem remota*/  
    background-size: cover; /*fazendo o navegador  
    ajustar, automaticamente, o tamanho da imagem ao fundo do  
    elemento*/  
    margin: auto; /*esta propriedade  
    centraliza a tabela na página web, horizontalmente*/  
}  
  
th, td {  
    border: 1px solid green;  
}  
  
th {  
    background-color: rgb(70, 130, 180);  
    color: white;  
}  
  
td {  
    text-align: center;  
}
```

Agora, confira as imagens de fundo sob a tabela e sob o cabeçalho **h1**.

Imagen de fondo em uma tabela com CSS



The screenshot shows a Microsoft Edge browser window with the title "Tabelas com HTML5 e CSS3". The address bar displays "file:///D:/Temp/tabela3.html". The main content is a table titled "Rendimento semestral de alunos - UC PRW1". The table has three columns: "Nome do aluno", "Nota da primeira avaliação", and "Nota da segunda avaliação". The rows represent students AAA, BBB, and CCC. Each cell contains a small sea creature icon (jellyfish, starfish, coral) and a numerical grade. The background of the entire table is a light blue color.

| Nome do aluno | Nota da primeira avaliação | Nota da segunda avaliação |
|---------------|----------------------------|---------------------------|
| AAA | 8,2 | 9,5 |
| BBB | 5,0 | 4,8 |
| CCC | 7,0 | 8,8 |

Fonte: Elaborada pelo autor (2024).

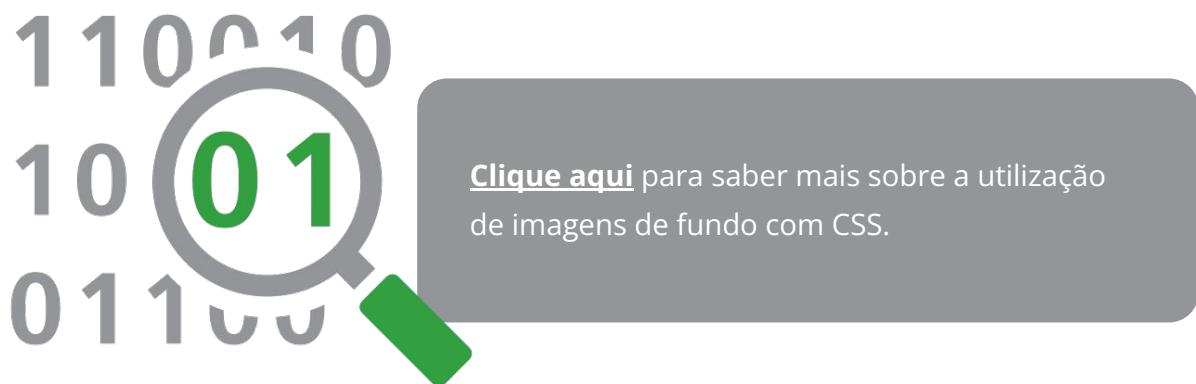
Quando se trata de inserirmos imagens de fundo por meio da propriedade `background-image` do CSS, devemos ter em mente duas situações:

- A imagem de fundo que estamos inserindo está armazenada em uma pasta ou diretório onde estão todos os arquivos de nossa aplicação web. Neste caso, dizemos que estamos inserindo uma **imagem local**.
- A imagem de fundo que estamos inserindo está sendo retirada de um computador em um endereço qualquer na internet (uma url qualquer). Neste caso, dizemos que a imagem de fundo sendo inserida é uma **imagem remota**.

No código anterior, por exemplo, a imagem de fundo usada para estilizar o cabeçalho **h1** da página web é uma imagem local, enquanto a imagem de fundo aplicada à tabela é uma imagem remota.

Imagens, geralmente, têm direito autoral, isto é, estão sob propriedade intelectual de uma pessoa, entidade, empresa, organização etc. Portanto, antes de utilizar qualquer imagem em uma aplicação web, certifique-se de que elas estão devidamente liberadas ou autorizadas para uso.

Lembrando, as imagens podem ser integradas em CSS por meio da propriedade "background-image", permitindo a personalização visual de elementos HTML.



Essa técnica oferece flexibilidade para adicionar detalhes visuais, como padrões, texturas ou até mesmo logotipos, sem comprometer a estrutura do documento HTML.

15. Formatação de formulários (etapa 1)

Você percebeu que, em seções anteriores, criamos toda a estrutura de um formulário, mas que foi apresentada pelo navegador sem nenhuma estilização? Pois

bem, é hora de cuidarmos de alguns detalhes que permitirão a alteração de diversos aspectos visuais dos componentes de formulários, por meio das folhas de estilo em cascata.

Formatação de labels

Quando utilizamos a tag <label> antes de cada caixa de informação, para indicar ao usuário o tipo de dado a ser inserido, o navegador, por padrão, renderiza cada caixa logo após o término do texto do respectivo label.



Esse comportamento conduz a um desalinhamento desses elementos – **label** e caixas de dados do tipo <**input**> – dando a impressão de uma interface gráfica descuidada. Para resolver esse problema, adotamos os seguintes passos:

- definimos uma largura para os labels afetados, com a propriedade CSS width;
- transformamos o elemento label para elemento do tipo nível de bloco, por meio da propriedade display, com o valor inline-block;
- alinhamos o texto dentro de cada label com a propriedade text-align, à direita;
- finalmente, separamos um label do seu anterior, dando um espaço maior entre eles, com a propriedade margin-bottom e uma medida qualquer, em

pixels, que você julgar adequada. Esta propriedade serve para separar um bloco label-caixa do bloco que está abaixo dele, dando um espaço maior ao conjunto, tornando o leiaute mais atrativo.

Também, é interessante colocarmos uma cor de fundo ou uma imagem de fundo em toda a área compreendida pelo formulário – por meio da propriedade background do CSS.

Para manter um senso de equilíbrio, podemos alinhar todo o conjunto no centro da página web, por meio da propriedade margin, com o valor auto.

Por padrão, o navegador classifica os diversos elementos de uma página web em elementos inline e elementos nível de bloco.

Qual a diferença entre esses dois grupos de componentes, do ponto de vista da aplicação das propriedades CSS?

- Os classificados como inline não obedecem a algumas propriedades CSS, como centralização, usando margens automáticas.
- Também, se um elemento for do tipo **inline**, ele permitirá que o navegador renderize outro componente **inline** do seu lado direito, desde que haja espaço suficiente para isso. Essa situação pode levar a posicionamento de conteúdo em locais indesejados.
- Elementos inline típicos de um formulário: **labels**, caixas do tipo **input**, botões submit - para envio de dados ao servidor - etc.
- Componentes classificados como nível de bloco têm a vantagem de obedecer a todas as propriedades CSS, mas impedem que qualquer outro marcador seja renderizado do lado direito dele. Essa situação também pode conduzir a leiautes visualmente não atrativos.

- Quando necessário, podemos ordenar ao navegador que converta qualquer elemento inline para nível de bloco, ou vice-e-versa, por meio da propriedade `display: block`, ou `display: inline`, ou, ainda, `display: inline-block`.

O valor `inline-block`, declarado para a propriedade `display`, torna o elemento “híbrido”. O que isso significa? Significa que, ao mesmo tempo, ele adquire comportamento de um componente `inline` e nível de bloco, permitindo, por exemplo, que determinada tag obedeça à centralização automática, e se posicione ao lado de outra tag existente. Você irá notar que utilizaremos, mais adiante, essa técnica para posicionarmos os labels ao lado das caixas de texto, em nosso formulário.



Vamos conferir, na prática, como podemos aplicar o CSS para alteração do estilo visual de diversos componentes de formulário. Veja a seguir:

</> Código na área

arquivo form.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Formulários com CSS </title>
    <link rel="stylesheet" href="formata-formulario.css">
</head>

<body>
    <h1> Aplicação de estilos a elementos de formulário com CSS </h1>

    <form method="post">
        <label class="alinhado"> Seu nome: </label>
        <input type="text" > <br>

        <label class="alinhado"> Saldo atual de seu cartão de crédito: </label>
        <input type="number" step="0.01" > <br>

        <label class="alinhado"> Data de validade de seu cartão: </label>
        <input type="date" > <br> <br>

        <label> Escolha a melhor data de pagamento da sua fatura: </label> <br>
        <input type="radio" value="0" > Antes do dia 05 de cada mês <br>
        <input type="radio" value="1" > Exatamente no dia 05 de cada mês <br>
        <input type="radio" value="2" > Após o dia 05 de cada mês <br> <br>

        <label> Selecione as formas de pagamento disponíveis: </label> <br>
        <input type="checkbox" value="0" > Casas lotéricas <br>
        <input type="checkbox" value="1" > Pela internet <br>
        <input type="checkbox" value="2" > Agência bancária <br>
        <br>

        <label> Selecione a forma de contato preferida: </label>
        <select>
            <option> Por e-mail </option>
            <option> Por SMS </option>
            <option> Por Whatsapp </option>
        </select> <br> <br>

        <button> Cadastrar dados </button>
    </form>
</body>
</html>
```

A seguir, você pode visualizar o código do arquivo CSS utilizado para formatar os labels do formulário anterior:

</> Código na área

arquivo formata-formulario.css

```
/*importando fonte personalizada externa*/
@import
url('https://fonts.googleapis.com/css2?family=Roboto+Con
densed:ital@0;1&display=swap');

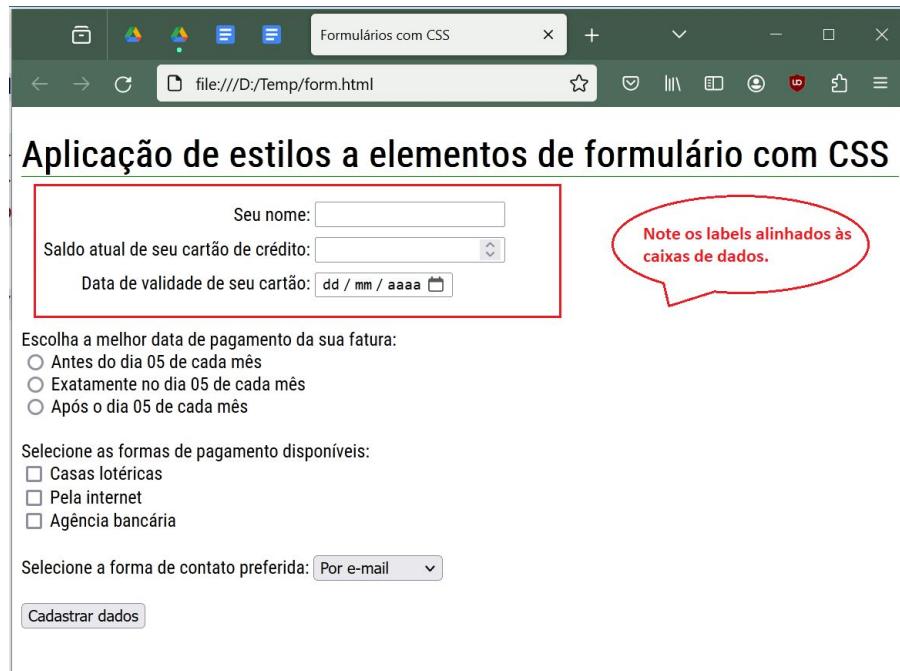
body {
    font-family: "Roboto Condensed"; /*declarando a fonte
personalizada para toda a página*/
}

h1 {
    border-bottom: 1px solid green; /*acrescentando uma
borda na parte inferior do título*/
}

/*fazendo o alinhamento dos labels às caixas de dados.
Repare que, no arquivo HTML, queremos alinhar somente os
labels que estão na frente das caixas input. Os outros
não devem ser afetados. Para isso, criamos a classe
"alinha" no HTML. Aqui, no CSS, mandamos o navegador
alinhar somente os labels com a class="alinha"*/
label.alinha {
    display: inline-block; /*modifica o
comportamento dos labels, para permitir alinhamento*/
    width: 250px; /*largura de cada
label*/
    text-align: right; /*alinha o texto do
label à direita*/
    margin-bottom: 10px; /*afasta uma caixa da
outra, na direção vertical*/
}
```

Muito complicado para entender? Para ajudar, confira, a seguir, o resultado, na página web, da renderização dos dois arquivos criados anteriormente:

Exemplo da renderização de um formulário com formatação em CSS



The screenshot shows a web browser window with the title "Formulários com CSS". The page content is titled "Aplicação de estilos a elementos de formulário com CSS". It contains the following elements:

- A red-bordered input group containing three fields:
 - Seu nome:
 - Saldo atual de seu cartão de crédito:
 - Data de validade de seu cartão: dd / mm / aaaa
- A red callout bubble pointing to the input group with the text: "Note os labels alinhados às caixas de dados."
- Text: "Escolha a melhor data de pagamento da sua fatura:"
- Three radio buttons:
 - Antes do dia 05 de cada mês
 - Exatamente no dia 05 de cada mês
 - Após o dia 05 de cada mês
- Text: "Selecione as formas de pagamento disponíveis:"
- Three checkboxes:
 - Casas lotéricas
 - Pela internet
 - Agência bancária
- Text: "Selecione a forma de contato preferida:
- A button:

Fonte: Elaborada pelo autor (2024).

Agora que apresentamos o básico da formatação de componentes label em um formulário, que tal darmos mais um passo e avançarmos na aplicação de estilos mais sofisticados? É o que faremos a seguir.

16. Formatação de formulários com CSS (etapa 2)

Dando sequência ao uso de estilos em nosso formulário-modelo, aplicaremos as seguintes estilizações:

- borda ao redor de todo o formulário;
- cor de fundo para a região do formulário;
- centralização do formulário na página web, na horizontal;
- área sombreada ao redor do formulário;
- centralização do botão submit;
- alteração do estilo da fonte do botão submit;
- alteração da largura e altura do botão submit;
- afastamento do botão submit em relação aos elementos que estão acima dele.

Detalharemos, na sequência, como aplicar cada um desses estilos. Acompanhe!

Borda ao redor do formulário

Neste ponto do nosso estudo, essa tarefa não deve mais ser um problema. Basta usarmos a propriedade CSS **border** e definirmos valores para espessura da borda, estilo e cor da borda, para o seletor **form**. No arquivo CSS, teremos uma linha de código semelhante a esta:

```
border: 1px solid green;
```

Outra coisa interessante de se ressaltar é que podemos arredondar os quatro cantos de uma borda, ao redor de qualquer elemento. Basta usarmos a propriedade a seguir:

```
border-radius: 10px;
```

Quanto maior for o valor da medida, em pixels, mais arredondado se torna o canto de cada borda.

Cor de fundo para o formulário

Da mesma forma, para aplicarmos uma cor de fundo para o formulário, basta utilizarmos a propriedade background-color, assim:

```
background-color: #fefefe;
```

Note que, na linha anterior, indicamos ao navegador uma outra forma de nos referirmos a cores, usando o sistema de numeração hexadecimal.

Observe, também, que seria muito fácil solicitarmos ao navegador a aplicação de uma imagem de fundo ao formulário, ao invés de uma cor. Como? Usando a propriedade a seguir:

```
background-color: url(nome-de-um-arquivo-de-imagem.png) ;
```



Para saber mais sobre as várias formas de utilizarmos cores em CSS, consulte [aqui](#)

Acompanhe, agora, como funciona o posicionamento dos elementos em uma página web.

Centralização de toda a caixa do formulário

Quando tratamos do posicionamento de elementos em uma página web, dois modelos se destacam:

- Todos os componentes da página web estão alinhados à esquerda da página. Este é o modo padrão de alinhamento do navegador.
- Todo o conjunto está centralizado na página web. Devemos utilizar uma propriedade específica para atingir esse objetivo.

A forma mais simples de centralizar a caixa do formulário é por meio da declaração margin, utilizando o valor dado pela palavra-chave auto, assim:

```
margin: auto;
```

A propriedade margin controla o espaço que o navegador insere ao redor dos quatro lados de uma caixa. Esses quatro lados podem ser controlados individualmente. Assim, para o form, declararmos a seguinte propriedade:

```
margin: 50px 60px 70px 80px;
```

Estaríamos indicando ao navegador para separar o elemento que está acima do formulário com uma distância de 50 px; o elemento que está à direita, com uma

distância de 60 px; o elemento que está abaixo do formulário, com uma distância de 70 px; e, finalmente, o elemento que está à esquerda do formulário, com uma distância de 80 px.

Quando utilizamos somente o valor auto para a propriedade margin, obtemos o efeito da centralização automática do componente na página web.



Sombreamento de uma caixa com CSS

É perfeitamente possível adicionarmos um sombreamento ao redor de qualquer tag na página web. Assim, se quisermos que o navegador renderize a caixa contendo o formulário e aplique uma sombra ao redor dela, podemos utilizar a seguinte propriedade:

```
box-shadow: 5px 6px 7px rgb(220,220,220);
```

O primeiro valor (5 px), medido em pixels, indica a largura da sombra no lado direito da borda do formulário. O segundo valor (6 px), indica a largura da sombra no lado inferior da borda do formulário. O terceiro valor (7 px), indica o grau de desfoque de uma sombra. Se esse valor é zero, a sombra se apresenta sólida. Quanto maior esse número, mais “borrada” ela se torna.

Centralização do botão submit

O botão submit representa um controle de formulário que é classificado como elemento inline. Portanto, do ponto de vista da aplicação de propriedades CSS, ele

tem o mesmo comportamento dos elementos label já tratados anteriormente. Portanto, antes de podermos aplicar a centralização, devemos convertê-lo em um elemento nível de bloco. Caso contrário, a centralização não funcionará. Isso é feito conforme vemos a seguir:

```
display: block;
```

Ao usarmos a propriedade display – já vista para os elementos label – o navegador converte o botão para um elemento de bloco, permitindo que ele obedeça a qualquer regra CSS que quisermos aplicar.

Após isso, basta usarmos a propriedade margin, com valor automático, e o botão estará centralizado.

```
margin: 30px auto;
```

Observe que o primeiro valor (30 px) da propriedade margin se refere à margem superior. Portanto, estamos dizendo ao navegador para colocar um espaço de 30 pixels entre o botão e o controle de formulário que está acima dele. O valor auto se refere, simultaneamente, à margem direita e esquerda, permitindo a centralização deste elemento na horizontal.

Alteração do estilo da fonte do botão submit

Se declararmos uma fonte personalizada para a página inteira, é de se esperar que todos os elementos que contêm texto estariam formatados com a referida fonte, certo? Por exemplo, a regra:

```
body { font-family: candara; }
```

Isso nos levaria a concluir que qualquer texto dentro da página web estaria sendo estilizado com a fonte Candara. Na prática, não é isso que acontece. Existem alguns controles de formulário que não herdam, automaticamente, a fonte declarada na página web. E o botão submit é um deles. Para que o texto do botão seja formatado corretamente, devemos declarar explicitamente a propriedade a seguir, para o botão:

```
button { font-family: uma-fonte-qualquer; }
```

Obrigamos, assim, o navegador a aplicar o estilo desejado. Também, note que, se o tamanho do texto do botão estiver muito pequeno, podemos aumentá-lo por meio da propriedade **font-size**. É possível deixarmos o texto do botão em negrito, usando a propriedade a seguir:

```
button {font-weight: bold;}
```

Alteração das dimensões do botão

Finalmente, se as dimensões do botão não estiverem adequadas em relação ao tamanho dos demais elementos, podemos facilmente alterá-las por meio do uso das propriedades a seguir:

```
width: 200px;  
height: 40px;
```

No exemplo anterior, estamos dizendo ao navegador para definir a largura do botão em exatamente 200 pixels e sua altura igual a 40 pixels. A seguir, você pode visualizar a folha de estilos externa completa utilizada para formatar o arquivo de formulário (form.html), já criado nas seções anteriores:

</> Código na área

formata-formulario.css

```
/*importando fonte personalizada externa*/
@import
url('https://fonts.googleapis.com/css2?family=Roboto+Condensed:ital@0;1&display=swap');

body {
    font-family: "Roboto Condensed"; /*declarando a fonte
personalizada para toda a página*/
}
h1 {
    border-bottom: 1px solid green; /*acrescentando uma borda
na parte inferior do título*/
    text-align: center; /*alinhando texto na
horizontal, centralizado*/
}

/*fazendo o alinhamento dos labels às caixas de dados.
Repare que, no arquivo HTML, queremos alinhar somente os
labels que estão na frente das caixas input. Os outros não
devem ser afetados. Para isso, criamos a classe "alinha" no
HTML. Aqui, no CSS, mandamos o navegador alinhar somente os
labels com a class="alinha"*/
label.alinha {
    display: inline-block; /*modifica o
comportamento dos labels, para permitir alinhamento*/
    width: 250px; /*largura de cada label*/
    text-align: right; /*alinha o texto do label
à direita*/
    margin-bottom: 10px; /*afasta uma caixa da
outra, na direção vertical*/
}

/*formatação do formulário*/

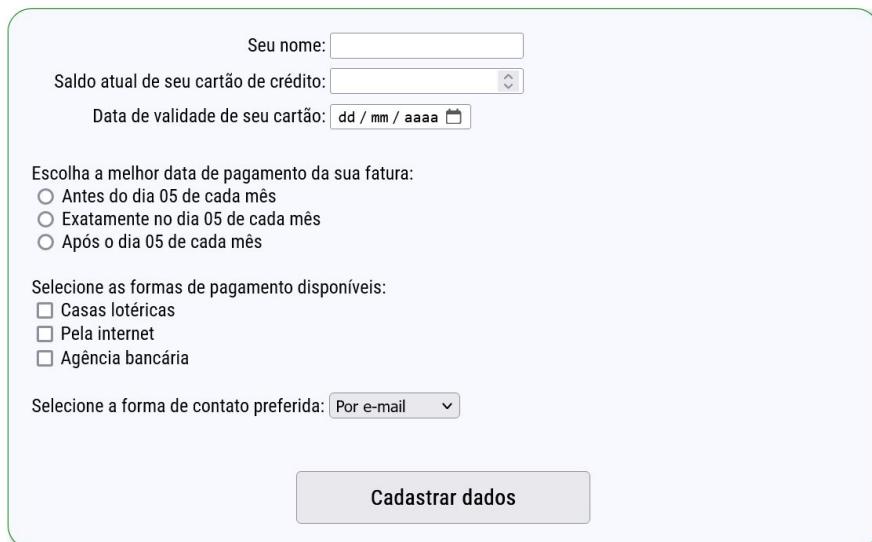
form {
    width: 700px; /*largura do formulário*/
    border: 1px solid green; /*borda ao redor do
formulário*/
    box-shadow: 5px 5px 5px rgb(47, 79, 79); /*cor da sombra ao
redor da borda*/
    border-radius: 20px; /*tamanho do
arredondamento das bordas*/
    margin: 30px auto; /*centralização
automática. O formulário empurra o cabeçalho h1 a uma
distância de 30px para cima*/
    background-color: #F8F8FF; /*cor de fundo
para o formulário, em hexadecimal*/
    padding: 20px; /*propriedade
que coloca um espaço de preenchimento DENTRO da caixa do
formulário, para que o texto não pareça grudado nas bordas*/
}
```

```
button {
    display: block;                                /*permite
    centralizar o botão*/
    margin: 25px auto;                            /*centraliza o botão
    e o afasta do elemento acima em 25 pixels*/
    width: 250px;                                 /*define a largura
    do botão*/
    height: 45px;                                /*define a altura do
    botão*/
    font-family: "Roboto Condensed";           /*define a fonte
    personalizada para o botão*/
    font-weight: bold;                            /*negrito*/
    font-size: 18px;                             /*tamanho da fonte*/
    margin-bottom: 0px;                           /*reduz a distância
    entre a borda inferior do botão e a borda inferior do
    formulário*/
}
```

Depois de tudo isso, como será a exibição da página web no navegador? Confira o resultado:

Renderização de um formulário completamente estilizado com CSS

Aplicação de estilos a elementos de formulário com CSS



Seu nome:

Saldo atual de seu cartão de crédito:

Data de validade de seu cartão: dd / mm / aaaa 

Escolha a melhor data de pagamento da sua fatura:

Antes do dia 05 de cada mês
 Exatamente no dia 05 de cada mês
 Após o dia 05 de cada mês

Selecione as formas de pagamento disponíveis:

Casas lotéricas
 Pela internet
 Agência bancária

Selecione a forma de contato preferida: Por e-mail

Cadastrar dados

Fonte: Elaborada pelo autor (2024).

Caro estudante, já evoluímos bastante em nossa caminhada rumo à construção de uma aplicação web completa. Porém, o que vimos até aqui ainda não é o suficiente. Por quê? Imagine a questão a seguir:



Preciso que o navegador do usuário esconda uma determinada área de menu em minha página web, quando o mouse não estiver posicionado sobre esta área. Como fazer?

Bem, a resposta à pergunta é simples: somente com o que vimos até agora (HTML e CSS), isso não é possível.

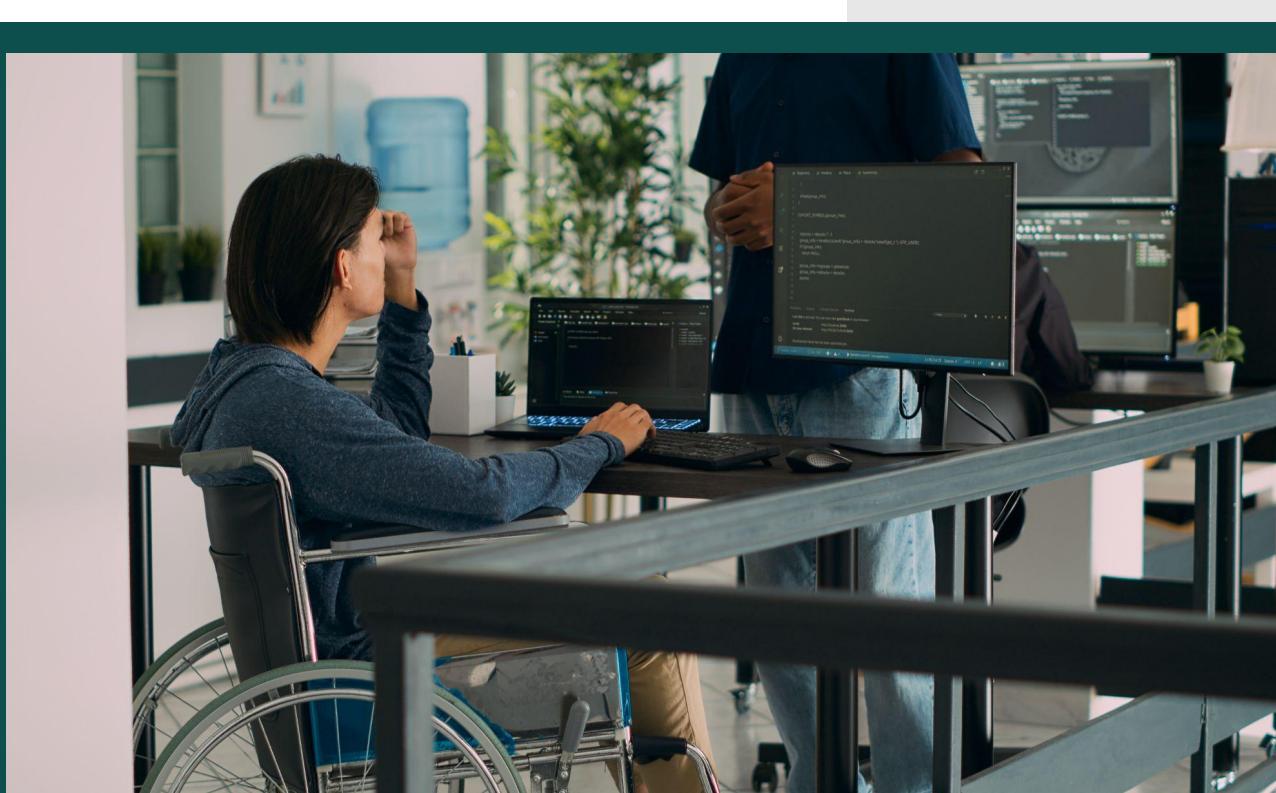
Para que nossa aplicação web possa reagir a esse tipo de evento, devemos introduzir uma nova linguagem, que será o tópico principal da próxima sessão. Então, vamos lá!

17. Introdução à linguagem de programação JavaScript

A linguagem de programação JavaScript adiciona uma camada de comportamento à nossa aplicação web. Mas o que isso quer dizer? Bem, queremos dizer que o navegador, por meio de comandos dessa linguagem, é capaz de detectar e reagir a eventos – disparados pelo usuário ou pelo próprio sistema – e dar uma resposta adequada a esses eventos. Isso é o que chamamos de comportamento do software em nosso sistema.

Embora, nos primórdios de sua criação, a linguagem JavaScript tenha sido idealizada como uma linguagem que só executaria comandos no navegador do cliente da aplicação, isso já não é mais inteiramente verdade. Na atualidade, a linguagem JavaScript também pode ser utilizada como linguagem de programação que executa comandos no servidor de nossa aplicação web.

Exemplos típicos de eventos que podem ocorrer em um documento web: abertura de uma página, fechamento de uma página, carregamento de uma imagem, clique em um botão de formulário, clique em um botão do mouse, aperto de uma tecla no teclado etc.



A linguagem de programação JavaScript é classificada como uma linguagem executada do lado do cliente, na máquina do usuário – cliente-side. Portanto, todo o conjunto de comandos é inserido, de forma integrada, junto do próprio código HTML e CSS. Esse código é executado pelo navegador do usuário.



Para saber mais sobre a linguagem JavaScript, consulte os seguintes links:

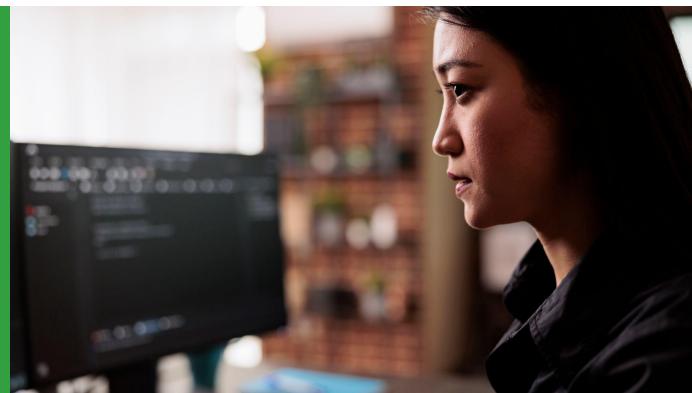
- [Tutorial completo.](#)
- [PDF de resumo.](#)
- [Introdução à linguagem JavaScript por meio de videoaulas - na tabela de conteúdos, videoaulas a partir da data de 25/02/2022.](#)

E que tal nos aprofundarmos um pouco mais nos elementos básicos dessa linguagem? Então, vamos adiante.

18. Constantes, variáveis, operadores e tipos de dados na linguagem JavaScript

Conforme você viu na UC Lógica de Programação, sempre que precisamos criar um programa de computador para automatizar determinada tarefa, precisamos utilizar uma linguagem de programação. E uma linguagem de programação é definida por alguns elementos, tais como:

- constantes
- variáveis
- operadores
- tipos de dados
- identificadores



Vamos conferir mais informações sobre as variáveis e os identificadores?

Variáveis e identificadores

À medida em que criamos um determinado programa de computador, ele precisa armazenar importantes informações. Essas informações podem ser fornecidas pelo meio externo – por exemplo, pelo usuário da aplicação – ou essas informações vão sendo geradas pelo próprio programa, no decorrer de sua execução. De qualquer maneira, o computador necessita de um modo de armazenar esses dados. Isso em JavaScript é feito por meio do uso de variáveis.

Cyberpedia

Variável: em JavaScript, uma variável é um identificador (nome de variável, nome de constante, nome de função, nome de objeto etc.) que utilizamos para que o código guarde determinado valor de uma informação.

Existem algumas regras a que devemos obedecer ao nomear uma variável: nomes de variáveis podem começar com letras ou com os símbolos \$ ou _. Elas só podem conter letras, números, \$ e _. Elas não podem começar com um número. Podem conter caracteres acentuados? Sim, embora seu uso seja desaconselhado.

Também, é importante lembrar que a linguagem JavaScript diferencia letras maiúsculas de minúsculas em um identificador. O que isso significa? Observe as linhas a seguir:

```
let media;  
let Media;
```

Para o JavaScript, embora as duas variáveis tenham o mesmo nome, elas são **DIFERENTES** entre si. Isto é, são duas variáveis completamente separadas, pois uma começa com **m** e a outra com **M**. Muita atenção a isso!

Além disso, antes do nome de uma variável, na primeira vez em que ela é referenciada, usamos o identificador let, assim:

```
let idade;
```

Embora o JavaScript não obrigue terminar cada linha de comando com o ponto-e-vírgula, é considerada boa prática de programação adotarmos este padrão.

Note, também, que, como o próprio nome sugere, uma variável pode armazenar, no decorrer do programa, valores diferentes de uma informação. Isto é, se criarmos uma variável para armazenar o nome de um aluno, ela pode ter, em determinado trecho do código, o valor, digamos, “João de Almeida”. Já em outro ponto do código, essa mesma variável poderia armazenar a informação “Maria da Conceição”.

Outro aspecto em relação a nomes de identificadores é quando o identificador é composto de várias palavras. Cuidado! Não podemos usar espaço em branco no identificador. Isso estaria errado:

```
let usuario não tem carro;
```

Nesses casos, podemos usar o formato conhecido como Camel Case, que consiste em escrever o primeiro nome todo em minúsculo, unido aos demais nomes. A partir

do segundo nome em diante, somente a letra inicial de cada termos é grafada em maiúsculo.

Outra forma é substituirmos cada espaço em branco pelo símbolo sublinhado.

Ao contrário das variáveis, as constantes em JavaScript são percebidas como elementos destinados a armazenar uma informação cujo valor, uma vez atribuído, não pode mais ser modificado. Ou seja, se criamos uma constante para armazenar a média final do aluno em **PRW1**, esse valor da média, uma vez atribuído, não pode mais ser modificado dentro da constante até o final da execução do código.

Em linhas gerais, as regras para nomearmos uma constante são as mesmas das regras para as variáveis. Uma constante sempre é criada declarando-se a palavra reservada **const** antes do seu nome, assim:

```
const meuCPF;
```

Operadores

No decorrer da criação de código em JavaScript, utilizamos muitas variáveis e constantes para a manipulação de dados. Não raro, necessitamos executar diversas operações com os valores dos dados presentes nesses identificadores.

É aí que surgem os operadores. Operadores são símbolos especiais utilizados com variáveis, constantes e outros identificadores da linguagem, que têm um significado especial e obrigam o computador a executar determinada ação (operação) sobre esses identificadores. Os tipos de operadores mais comuns são:

■ Operadores matemáticos ou aritméticos

Realizam uma operação matemática com os dados armazenados nos identificadores. Exemplo: soma, subtração, divisão, multiplicação etc.

■ Operadores relacionais

Realizam uma comparação lógica entre os operandos. Exemplo: maior, menor, maior ou igual, menor ou igual, igual, diferente etc. Isso sempre resulta em um valor verdadeiro ou falso.

■ Operadores lógicos

Executam uma operação lógica entre os dados (E, OU ou NÃO). Essa operação sempre resulta em um valor lógico verdadeiro ou falso.

■ Operador de concatenação

Serve para juntar dois ou mais identificadores que armazenam texto. Em JavaScript, o termo “string” é sinônimo de texto. O operador utilizado aqui é o símbolo matemático da soma + (mais).

■ Operador de atribuição

Um dos mais utilizados na linguagem específica que o conteúdo do lado direito do símbolo será colocado (atribuído, inserido) dentro da variável (ou constante) do lado esquerdo da atribuição. Em JavaScript, o símbolo de atribuição é o sinal de igual (=). Exemplo: idade = 28.

Em síntese, os operadores, como símbolos especiais na linguagem, desempenham um papel crucial ao induzir o computador a executar operações específicas sobre variáveis, constantes e outros identificadores. Essa funcionalidade é essencial para a manipulação e o processamento eficaz de dados em diferentes contextos de programação.



[Clique aqui](#) para fazer uma consulta rápida sobre operadores em JavaScript.

Conheça, agora, os tipos de dados em JavaScript.

Tipos de dados na linguagem JavaScript

É comum, na maioria das linguagens de programação, que, ao declararmos (criarmos) um identificador (variável, constante, objeto, função etc.), necessitemos informar qual o tipo de dado que está sendo armazenado naquele identificador. Isto é, precisamos informar, no código, se o dado é numérico, se é texto, se é lógico (verdadeiro ou falso) etc.

Quando uma linguagem de programação apresenta esse tipo de exigência, ela pode ser classificada como uma linguagem fortemente tipada, por exemplo: Java, C++, Python, entre outras.

Não é o caso do JavaScript, ao menos não quando utilizamos o interpretador da linguagem em sua versão padrão, e não a versão estrita. Mas isso é assunto para um outro momento. Aqui, podemos tranquilamente criar uma variável ou constante e atribuirmos a elas o valor que quisermos, sem a necessidade de especificar um tipo

de dado. Linguagens com esse comportamento são classificadas como **linguagens de programação fracamente tipadas**, como, por exemplo, PHP, Ruby etc.



Ainda assim, para efeito de clareza e legibilidade de código, sempre é importante considerarmos que, como qualquer outra linguagem de programação, o JavaScript permite a classificação dos dados em tipos como texto, número, lógica etc. Porém, não exige que esses tipos sejam especificados no momento da criação de um identificador.

Outro fato importante é que, em qualquer linha de comando, se você quiser que o JavaScript entenda que você está manipulando texto ou string, você sempre deve colocar o valor do texto entre aspas ("") ou apóstrofos (''). Veja::

```
let nomeDoAluno = "Joana de Gusmão"; ou  
let nomeDoAluno = 'Joana de Gusmão';
```

Em JavaScript, o símbolo de comentário de várias linhas, dentro do código, inicia com /* e termina com */. Para comentários de uma só linha, use o símbolo //

Mostramos, até aqui, na teoria, diversos aspectos introdutórios desta que é, na atualidade, uma linguagem de programação indispensável para a criação de uma aplicação web completa. Agora, vamos juntar tudo isso em um código de exemplo que você confere a seguir:

</> Código na área

```
arquivo fundamentos-javascript.js
//criando e declarando variáveis em JavaScript
let 7Idade; //incorrecto. Nome de variável não pode
começar com dígito
let idade; //correto
let minha idade; //incorrecto. Sem espaço em branco no
nome da variável
let nomeDoAluno = "Joana da Silva"; //declara e atribui
o valor de texto à variável. Note a string entre aspas
let temCarro;
let nota1, nota2; //lista de declaração
let soma, divisao;

//criando constantes
const cpf;
const mediaFinal;
const idadeFixa = 35; //declara e atribui o valor à
constante

//usando operadores matemáticos
soma = nota1 + nota2;
divisao = nota1 / nota2;

//operadores relacionais
let resultado1, resultado2, resultado3;
resultado1 = nota1 > nota2; //o resultado guardado
dentro da variável resultado é sempre booleano
(verdadeiro ou falso)
resultado2 = nota2 == nota1; //operador de igualdade
resultado3 = nota2 != nota2; //operador de diferença
(diferente)
```

E agora, para finalizar este tópico, vamos deixar a seu critério a descoberta de uma informação importante, quando se trata de salvar código em JavaScript em um arquivo. Veja o que propomos a seguir:



Deu bug?

Um arquivo que armazena código na linguagem JavaScript deve sempre ser salvo com uma extensão. Olhando o quadro anterior, você saberia dizer qual é essa extensão?

Até agora, estudamos como escrever comandos da linguagem JavaScript que declaram variáveis, constantes, que usam operadores e tipos de dados. Mas, de fato, onde inserir todos esses comandos em uma página web? Responderemos essa sua dúvida no tópico a seguir.

19. Formas de inserção do JavaScript em um documento web

Há formas diferentes de se inserir código da linguagem JavaScript em uma página web. Dependendo do método adotado, podemos não aproveitar todas as vantagens que a linguagem oferece.

Com isso em mente, apenas mencionaremos o formato de inserção inline e incorporado, dedicando uma atenção maior ao formato de arquivo externo.

Em JavaScript inline, os comandos da linguagem são inseridos dentro da própria tag HTML que está sendo utilizada. Por exemplo:

```
<button onclick="alert('Olá, mundo!');"> Clique no botão para  
mostrar o alerta </button>
```

Note, na linha anterior, que os comandos da linguagem são colocados entre aspas, como valores de um atributo do elemento button, dentro da própria página web. O

atributo especial, neste caso, **onclick**, funciona como um comando da linguagem JavaScript. A desvantagem desse método é que o efeito desse comando está estritamente “amarrado” a um único elemento da página, o que limita muito a eficiência da linguagem. Apenas como curiosidade, esse trecho de código obrigará o navegador a exibir uma pequena caixa de alerta com o texto “Olá, mundo!” toda vez que o usuário disparar o evento de clique sobre o botão submit de um formulário.

Em JavaScript incorporado, os comandos da linguagem são inseridos dentro da própria página html utilizada. Novamente, temos o problema de que esse código serve somente para a página em questão, o que é inconveniente se necessitarmos utilizar os mesmos comandos da linguagem em várias páginas web diferentes. Veja:

</> Código na área

```
arquivo fundamentos-javascript.html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Inserção do JavaScript em um documento web
</title>
</head>

<body>
    <h1> Método de inserção do JavaScript incorporado </h1>
    <p> Desvantagem: este comando só serve para esta
    página. E se quisermos utilizá-lo em outras "n" páginas
    de nossa aplicação web? Devemos, em todas elas, escrever
    o mesmo código novamente. </p>
    <script>
        alert("Olá, mundo!");
    </script>
</body>
</html>
```

Note, na listagem anterior, a presença da tag especial **<script>**, utilizada para a inserção de comandos do JavaScript dentro da página web. Toda vez que o usuário

carregar essa página no navegador, será exibida a mesma caixa de alerta já referida anteriormente.

JavaScript externo: os comandos da linguagem são inseridos dentro de um arquivo externo, separado da página web, com a extensão **js**. Esse arquivo é vinculado à página HTML por meio do uso da tag especial `<script>`, geralmente localizada ao final do documento.

Observe o modelo a seguir:

</> Código na área

arquivo fundamentos-javascript-externo.js

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Inserção do JavaScript em um documento web
</title>
</head>
<body>
    <h1> Método de inserção do JavaScript externo </h1>
    <p> Vantagem: agora, toda e qualquer página web da minha aplicação pode reutilizar os mesmos comandos em JavaScript presentes no arquivo externo. Basta utilizar a tag script, que vincula o arquivo js a este documento web. </p>

    <script src="meu-arquivo-externo.js" ></script>

</body>
</html>
```

O método de vinculação citado anteriormente é, de longe, o mais eficiente, pois permite que toda e qualquer página web possa utilizar o mesmo conjunto de

comandos da linguagem JavaScript, sem a necessidade de duplicação de código em vários locais e arquivos diferentes de nossa aplicação.

Observe, novamente, a presença da tag <script>, colocada logo antes do fechamento da página, e o seu atributo *src*, contendo o nome e a localização do arquivo externo com os comandos da linguagem.

Note, também, que a tag <script>, embora seja completa, não apresenta nenhum conteúdo dentro dela. Ela é aberta e imediatamente fechada, sem sequer conter qualquer espaço em branco. A única informação colocada é o valor do atributo src.

Agora, toda vez que esta página for renderizada pelo navegador, ele sempre irá localizar o arquivo JavaScript externo e executar os comandos lá inseridos. No presente estudo, sempre utilizaremos essa forma de inserção de comandos da linguagem em nossa aplicação web.

Apenas para o caso de você estar se perguntando se é possível adicionarmos várias tags <script> ao final das páginas, cada uma delas incluindo um arquivo externo .js diferente, a resposta é sim.

Agora que você já conhece os métodos pelos quais os comandos JavaScript são executados dentro de uma página web, vamos para a última parte do estudo de nossa Unidade Curricular.

20. Funções de usuário na linguagem JavaScript

A linguagem JavaScript permite a utilização de uma estrutura própria de organização de código chamada de *função de usuário*.

Funções de usuário tratam de blocos de código que, geralmente, tendem a se repetir várias vezes em nossa aplicação e que são manipulados de forma especial, a fim de tornar a execução e manutenção do referido código mais eficientes.

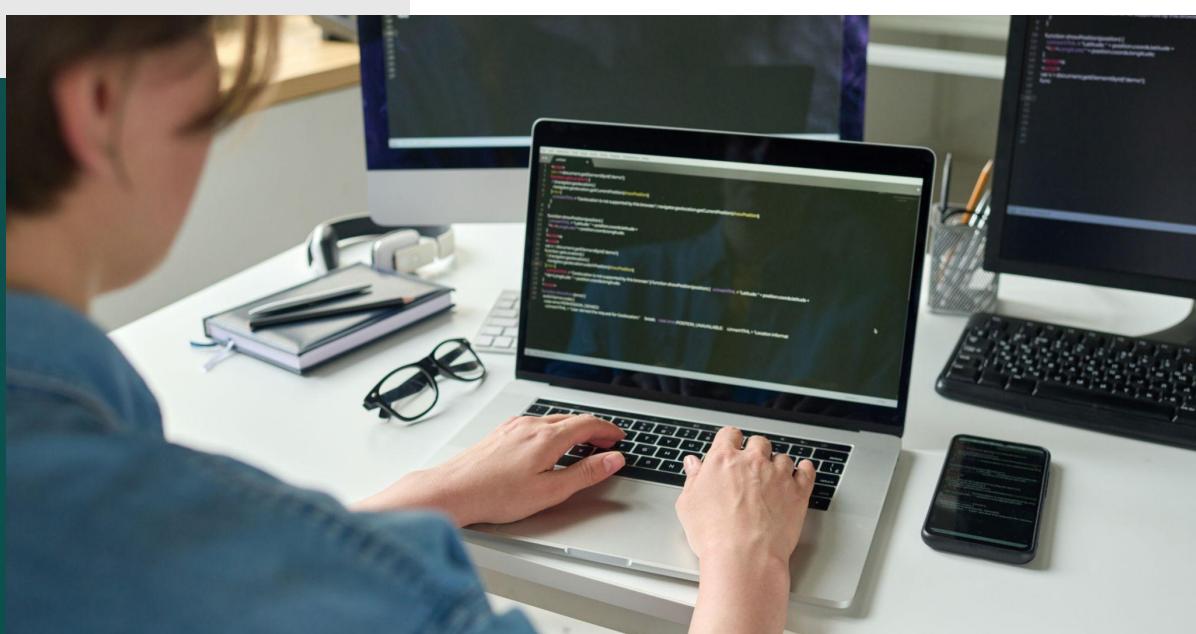


A técnica de separação do código que se repete, retirando-o do fluxo normal de execução dos comandos (área chamada de programa principal ou script principal) e colocando-o em uma sessão especial da aplicação, é referida, também, pelos termos *subrotina* ou *subprogramação*, conforme você já aprendeu nas aulas de Lógica de Programação.

Talvez você tenha se perguntado o que significa o termo *script*, no universo do desenvolvimento de código. Aqui, esse termo tem absolutamente o mesmo significado de programa de computador.

Em síntese, sempre que você perceber a repetição de várias linhas de comando dentro de sua aplicação, é um bom indício de que essas linhas de código devem estar inseridas em uma função de usuário.

Uma função de usuário é um bloco de código especial, criado pelo próprio desenvolvedor web, para tornar a resolução do problema mais eficiente. Porém, a própria linguagem já dispõe de inúmeras funções prontas, que você **INVOCÁ** toda vez que necessita executar uma tarefa especial dentro do código. Já vimos e utilizamos uma delas em páginas anteriores: a função `alert()` do JavaScript.



As funções de usuário podem receber diversas classificações na linguagem JavaScript e podem ser implementadas e utilizadas de maneiras bem diferentes. Veja alguns tipos de funções de usuário suportados pela linguagem:

- funções de usuário regulares;
- funções de usuário anônimas;
- literais de funções de usuário;
- funções de usuário tipo seta.

Das mencionadas anteriormente, neste estágio de nosso estudo, cobriremos apenas as funções regulares de usuário.

Anatomia de uma função regular de usuário em JavaScript

Toda função regular de usuário é composta por uma declaração, em um local específico do arquivo JavaScript em que está sendo definida. Por declaração, subentendem-se duas partes:

01

Declaração do cabeçalho da função.

02

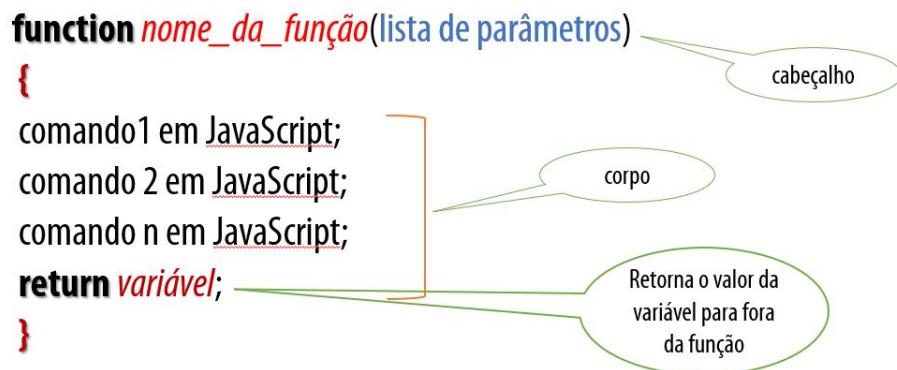
Declaração do corpo da função.

A declaração do cabeçalho inicia-se pela palavra reservada ***function***, seguida do nome da função (um identificador) e de um conjunto de parênteses – os parâmetros da função (adiante, você verá mais sobre eles).

Após isso, temos a declaração do corpo da função, que consiste em abrir uma chave {, escrever os comandos da linguagem JavaScript que serão executados pela função e, em seguida, fechar o corpo da função com o símbolo }. Observe o gráfico a seguir para ter uma compreensão mais clara de como declaramos uma função em JavaScript:

Declaração de uma função regular de usuário em JavaScript

Anatomia de uma função



Fonte: Elaborada pelo autor (2024).

Note que o nome de uma função em JavaScript deve obedecer às mesmas regras dadas para nomes de variáveis, como já vimos anteriormente. A lista de parâmetros contém as informações (variáveis), que serão recebidas da área externa à função (o script principal) e que são necessárias para que a função execute seu trabalho.

A lista de parâmetros deve sempre estar entre parênteses. Se houver a passagem de mais de um parâmetro, eles devem estar separados por vírgula. Observe que, se a função não necessitar de nenhuma informação do meio em que ela está sendo invocada, os parênteses ficam vazios. Dentro do corpo da função, temos os comandos que reexecutarão determinada tarefa.

Variáveis locais e globais e a lista de parâmetros

Por definição, em JavaScript, toda variável passada como parâmetro é tratada, dentro do corpo da função, como uma **variável local**.

Mas o que é uma variável local no âmbito de uma função? É uma variável que só existe enquanto o código da função estiver sendo executado. No momento em que o interpretador da linguagem terminar a execução da função, qualquer referência a uma variável local se torna inválida. De modo mais simples, assim que uma função termina, todas as variáveis locais passadas como parâmetros ou criadas dentro da função são extintas da memória RAM do computador. Se tentássemos acessar o valor de uma variável local fora da função, o programa detectaria um erro de execução.



Deu bug?

Neste ponto, surge um problema: o que devo fazer se precisar acessar o valor de uma variável local fora da função?

O JavaScript resolve isso de várias maneiras diferentes: uma delas é declarar todas as variáveis, tanto aquelas criadas fora da função como aquelas criadas dentro da função, como **variáveis globais**. Uma variável global pode ter seu valor reconhecido corretamente, não importando o “escopo” da variável, isto é, não importando o local do código onde ela foi criada - dentro ou fora da função. Esse procedimento parece resolver nosso problema anterior, mas não é o mais indicado, pois tende a gerar erros difíceis de serem solucionados em aplicações mais complexas.

Desta maneira, para contornar a situação, o JavaScript permite que qualquer variável criada como variável local, dentro de uma função, transfira seu valor para o ponto em que ela foi invocada (fora da função), atuando como se estivéssemos tratando com uma variável global. Isso é feito por meio do comando **return**, dentro do corpo da função.

CUIDADO! Qualquer comando após a linha do return não será executado dentro da função. Uma função termina no momento em que o JavaScript executa o comando return!

E como fazemos a “ligação” entre a área externa do nosso código (o script principal), em que a função será invocada para fazer seu trabalho, e a área da declaração da função? Estude com atenção o gráfico a seguir, para entender melhor como isso acontece:

Invocação e declaração de uma função regular de usuário em JavaScript

Invocação e declaração da função

```
//script principal - invocação da função  
let retorno = nome_da_função(lista de  
argumentos);  
//outros comandos;
```

```
//declaração da função  
function nome_da_função(lista de parâmetros)  
{  
    comando1 em JavaScript;  
    comando 2 em JavaScript;  
    comando n em JavaScript;  
    return variável;  
}
```

Fonte: Elaborada pelo autor (2024).

Assim que o JavaScript encontra o comando return, dentro da função, o valor é retornado (e a função termina) para o script principal e armazenado dentro da variável que está do lado esquerdo do operador de atribuição – em nosso gráfico, a variável de cor lilás chamada de retorno.

E se a lógica de nossa função diz que não há a necessidade de se retornar nenhum

dado para a área externa, em que ela foi invocada, como fazemos? Bem, neste caso, basta somente invocar o nome da função no script principal, sem termos nenhuma variável de retorno. Também, neste caso, não teríamos o comando return dentro da função. Para clarear, observe o gráfico a seguir:

Invocação e declaração de uma função regular de usuário em JavaScript, sem retorno

Invocação e declaração da função sem retorno

| | |
|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>//script principal - invocação da função nome_da_função(lista de argumentos); //outros comandos;</p> | <p>//declaração da função function nome_da_função(lista de parâmetros) { comando1 em JavaScript; comando 2 em JavaScript; comando n em JavaScript; return variável; }</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Fonte: Elaborada pelo autor (2024).

Depois de tudo isso, que tal conferir, na prática, como fica a codificação? Na listagem a seguir, ilustramos, por meio de linhas de comando da linguagem, o que foi abordado sobre funções até o momento, dando uma visão geral de como as partes do código em JavaScript estão interligadas – o script principal, a invocação da função e a declaração da função, juntamente com a página HTML.

Só para clarear, estamos supondo que temos duas notas parciais de um aluno qualquer de nosso curso e queremos que o JavaScript nos mostre qual a média final desse aluno logo que a página HTML for carregada no navegador. Essa operação (cálculo da média) deve ser executada por uma função regular de usuário em JavaScript.



Você irá notar que o código dentro de cada arquivo está abarrotado de comentários. Isso é necessário para você entender com clareza o que o JavaScript está fazendo. Mas isso pode prejudicar sua legibilidade. Como tarefa adicional, sugerimos que você abra esses dois arquivos em seu editor, remova todos os comentários, salve os arquivos modificados em uma pasta de seu computador – documentos, por exemplo – e analise novamente os comandos, agora sem excesso de comentários. Vamos ao programa!

</> Código na área

arquivo calcula-media.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="utf-8">
    <title> Integrando JavaScript em uma página web </title>
</head>
```

```
<body>
  <h1> Integrando JavaScript em uma página web </h1>
  <p> Exemplo prático mostrando como podemos fazer o
JavaScript calcular a média de duas notas de um aluno, no
momento em que a página é carregada no navegador. </p>
  <p> Para simplificar o exemplo, supomos que estas notas já
estão fixadas: a primeira terá o valor 6,5 e a segunda terá
o valor da nota 8,5. Porém, em uma aplicação web verdadeira,
as duas notas deverão ser fornecidas em campos de
formulário. Pela complexidade do assunto, estas técnicas do
JavaScript serão abordadas em Unidades Curriculares
posteriores, em nosso curso. Recarregue a página para ver
novamente o JavaScript em ação. </p>

  <!--Chamando o código externo do JavaScript a ser
executado-->
  <script src="calcula-media.js"></script>
</body>
</html>
```

No trecho de código a seguir, você confere como utilizamos todos os conceitos vistos anteriormente e relacionados ao uso de uma função da linguagem JavaScript, para fazer o navegador calcular a média aritmética simples de duas notas (6,5 e 8,5) de um aluno qualquer e exibir o resultado dessa média na página web:

</> Código na área

arquivo calcula-media.js

```
//declaração da função - área especial do nosso código em
//JavaScript - esta pode ser considerada uma área somente para
//a declaração das funções. Note, dentro dos parênteses, as
//duas variáveis nota1 e nota2, que armazenam as notas vindas
//de fora da função e que serão utilizadas para calcularmos a
//média, dentro da função. São os parâmetros da função
function calcularMedia(nota1, nota2)
{
    //definição dos comandos no corpo da função
    let media; //media, aqui, é uma variável local. Esta
    //variável é perdida logo depois que a função termina sua
    //execução
    media = (nota1 + nota2)/2;
    return media; //portanto, se quisermos que a função envie
    //o valor da média lá para o script principal, onde ela foi
    //invocada, devemos chamar o comando return antes de a função
    //finalizar
}

//área do script principal - as linhas de comando a seguir
//são executadas pelo JS logo que a página é carregada. O
//código dentro da função permanece inativo. A função só é
//executada no momento em que o JavaScript chega na linha em
//que invocamos a função
let mediaCalculada;
let nota1 = 6.5;
let nota2 = 8.5;
//invocar a função, e receber a média que ela irá calcular.
//Note a variável de retorno abaixo, para receber a média que
//voltará depois que a função fizer seu trabalho
//(mediaCalculada). Note, também, que, para a função poder
//executar sua tarefa, é necessário que passemos para dentro
//dela, os dois argumentos, dentro do parênteses, que
//representam o valor das duas notas
mediaCalculada = calcularMedia(nota1, nota2);

//agora, basta pedir que o JavaScript mostre, no navegador,
//o valor final da média calculada, usando uma caixa de alerta
//(uma caixa pop-up). Uma caixa de alerta é sempre um
//componente que mostra somente texto no navegador. Se
//quisermos que o JavaScript exiba, também, o valor dentro de
//uma variável, devemos fechar o texto entre aspas e
//concatenar a variável com o operador de concatenação +
alert("Com os valores das duas notas fornecidos, o
JavaScript calculou que a média final do aluno é igual a " +
mediaCalculada);
```

Finalizando

Quanta coisa aprendemos até aqui, não é mesmo? Lembra de quando começamos nosso estudo?

Iniciamos do básico, apresentando a você os conceitos relacionados ao desenvolvimento web e à necessidade fundamental de conhecermos uma linguagem importante, uma linguagem universal, que os navegadores utilizam para mostrar o conteúdo de qualquer página web – a linguagem HTML.

Em seguida, você aprendeu como criar uma página com os principais elementos dessa linguagem.

Depois, você viu que a página poderia estar funcional, mas sem o atrativo visual adequado. Quem veio nos salvar? Isso mesmo, as conhecidas Folhas de Estilo em Cascata, ou o CSS.

Com isso, você pôde perceber as numerosas possibilidades que o desenvolvedor web tem a seu dispor, em se tratando da aplicação de estilos. Lembra como nosso formulário HTML adquiriu vida nova após a formatação com CSS?

Mas chegamos a um determinado momento do desenvolvimento de nossa aplicação que nos revelou que o HTML e o CSS podem fazer muita coisa em uma página web, mas não podem fazer tudo. E se quiséssemos que o navegador ocultasse um botão quando o usuário clicasse nele? E se precisássemos da média de um aluno a partir de duas notas, lembra? Tínhamos um problema. Precisávamos de ajuda, e quem apareceu para nos socorrer? Isso mesmo, a linguagem JavaScript.

Com ela, você aprendeu que nossa aplicação web pode fazer muito mais do que apenas montar a estrutura da página ou formatar elementos. Vimos que o JavaScript nos oferece muitas e incríveis possibilidades, mas ainda há muito mais a ser descoberto!

Será que existem outros aspectos importantes da linguagem que ainda não abordamos? O que podemos dizer da execução de parte da aplicação no lado do servidor, já que, até agora, só trabalhamos com o front-end, código que executa na máquina cliente, por meio do navegador? E se precisarmos que nossa aplicação web, por exemplo, interaja com dados armazenados em banco de dados? Isso é possível de ser implementado?

Ainda temos muitas questões em aberto. Todas elas serão respondidas, e muito mais, na continuidade de nosso estudo. Aguardamos vocês em nossa próxima etapa. Até lá!

Referências

DAMINELLI, Herval. **Desenvolvimento de aplicações para web, com banco de dados, programação orientada a objetos, PHP, CSS, HTML e JavaScript.** IFSC Câmpus Florianópolis. 2024. Disponível em: <https://hdam.pro.br/>. Acesso em: 10 jun. 2024.

FREEMAN, E.; ROBSON, E. **Head First JavaScript Programming.** Sebastopol, CA: O'Reilly Media, 2014.

VARIÁVEIS e Operadores. **UFRJ** – Núcleo de Computação Eletrônica. [2024]. Disponível em: <http://www.nce.ufrj.br/ginape/js/conteudo/variaveis/operadores.htm>. Acesso em: 10 jun. 2024.

W3C. **CSS tutorial.** 2024. Disponível em:
<https://www.w3schools.com/css/default.asp>. Acesso em: 10 jun. 2024.

W3C. **HTML tutorial.** 2024. Disponível em:
<https://www.w3schools.com/html/default.asp>. Acesso em: 10 jun. 2024.

W3C. **JavaScript tutorial.** 2024. Disponível em:
<https://www.w3schools.com/js/default.asp>. Acesso em: 10 jun. 2024.

