

AULA 1 – HTML5

OBJETIVO DA AULA

Conhecer os principais elementos que compõem a linguagem HTML.

APRESENTAÇÃO

Você sabia que a maioria das páginas web visitada por você foi desenvolvida em HTML? HTML é uma linguagem de marcação de hipertexto desenvolvida por Tim Berners-Lee, também conhecido como “pai da internet”, na década de 90. Ao longo dos anos, a linguagem foi sendo aprimorada e atualmente estamos na sua quinta versão, também chamada de HTML5. Para criarmos e editarmos códigos em HTML é necessário usar um IDE (ambiente de desenvolvimento integrado) como: *VSCode*, *Sublime*, *Notepad++* etc. Na verdade, você pode utilizar até mesmo o bloco de notas para criar e editar códigos em HTML, porém ele não possui certos recursos que podem agilizar suas tarefas do dia a dia. Ao longo desta aula você aprenderá como criar um documento HTML.

CONTEÚDO

Atualmente, a web segue padrões estabelecidos pela W3C (*World Wide Web Consortium*), mas nem sempre foi assim. No final da década de 90 passamos por um período denominado como “**guerra dos navegadores**”, em que o Internet Explorer e o Netscape “brigavam pela supremacia da Web”, resultando em sites que só funcionavam em um determinado navegador (CLARK et al., 2014). Um verdadeiro pesadelo! Já imaginou se ainda fosse assim?

O que chamamos de padrões, é oficialmente chamado de “Recomendações” pela W3C. São as maneiras recomendadas de funcionamento para as tecnologias da web. Não há nenhum tipo de lei que obrigue navegadores e vendedores de ferramentas a adotá-las; em vez disso, a adoção é um acordo para o bem da Web e o benefício mútuo de todos (CLARK et al., 2014).

LINK

Visual Studio: <https://code.visualstudio.com/download>. Acesso em: 05/01/2023.

Sublimetext: <https://www.sublimetext.com/3>

Notepad: <https://notepad-plus-plus.org/downloads/>

W3C: <https://www.w3c.br/Padroes/>

Resumindo, o W3C “é uma organização internacional de padrões que desenvolve os pilares de tecnologias Web tais como HTML, CSS, SVG, XML e WCAG” (W3C BRASIL, 2017).

Agora que já entendemos a importância dos padrões, vamos entender o que é o HTML e quais são os seus elementos básicos. Como dito anteriormente, o HTML (*Hypertext Markup Language*) é uma linguagem de marcação de hipertexto utilizada para construção de páginas web. Segundo Ferreira e Eis (2015), um hipertexto nada mais é do que um conjunto de elementos (palavras, imagens, vídeos, áudio, documentos etc.) ligados por conexões. Juntos, estes elementos formam uma grande rede de informação.

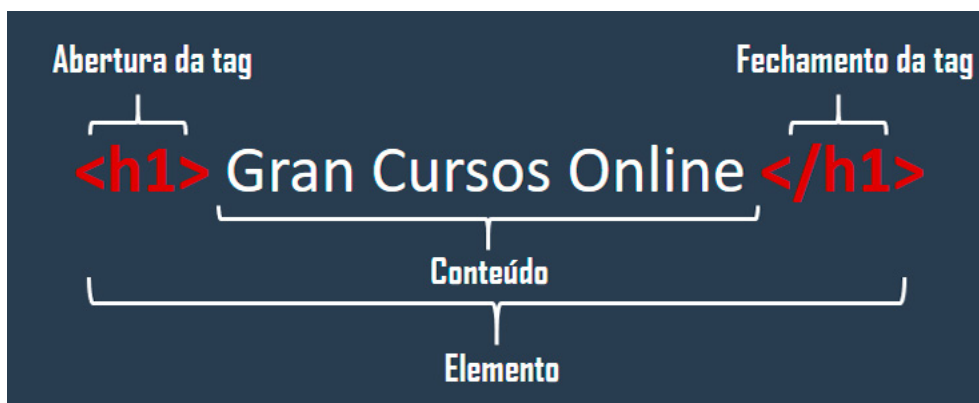
Para distribuir informação de uma maneira global, é necessário haver uma linguagem que seja entendida universalmente por diversos meios de acesso. O HTML se propõe a ser esta linguagem (FERREIRA e EIS, 2015).

O HTML5, a mais nova versão do HTML, trouxe consigo um diferencial semântico significativo e facilidade na manipulação de elementos via JavaScript ou CSS. O intuito desta nova versão não é reinventar a roda, pois ela é compatível com todos os recursos da especificação HTML4, entretanto, precisou-se modificar e melhorar alguns elementos e atributos para que os mesmos pudessem ser reutilizados de forma mais eficaz.

Resumindo, o HTML5 criou novas tags e modificou a função de outras. Seus elementos semânticos permitem que o HTML não só estruture documentos para a web, mas também descreva o significado do conteúdo presente no nosso código, tornando-o mais claro e limpo tanto para os desenvolvedores quanto para os navegadores e motores de busca que processam informações. Por exemplo, as versões anteriores do HTML “não continham um padrão universal para a criação de seções comuns e específicas como rodapé, cabeçalho, sidebar, menus e etc.”. Agora, contamos com as tag semânticas (<footer>, <header>, <aside>, <nav>, respectivamente) para nos ajudar a estruturar o nosso código semanticamente (FERREIRA e EIS, 2015). Em breve, detalharemos cada uma destas tags.

O que é uma **TAG**? Tag significa etiqueta, no HTML usamos estas etiquetas para marcar os nossos elementos, ou seja, podemos determinar onde começa e onde termina um determinado elemento, daí o nome linguagem de marcação. Vale ressaltar que, nem todas as tags precisam ser fechadas, como a tag
, utilizada para pular de linha. Observemos na Figura 1 a anatomia de um elemento HTML.

FIGURA 1 | Anatomia de um elemento HTML



Fonte: Adaptada de MDN Web docs (https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/HTML_basics).

As tags são representadas por parênteses angulares (< >), na Figura 1 temos o exemplo da tag <h1>, utilizada para definir um título. Na verdade, <h1> é a primeira e mais importante das seis tags (h1, h2, h3, h4, h5, h6) utilizadas para criar e diferenciar os títulos em um documento HTML. Agora que já sabemos o que é uma tag, vamos aprender quais são as tags que compõem a estrutura básica do HTML5. Vejamos (Figura 2):

FIGURA 2 | Estrutura básica HTML

```

1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4    <meta charset="UTF-8">
5    <title>Gran Cursos Online</title>
6  </head>
7  <body>
8
9  </body>
10 </html>

```


Fonte: Elaborado pela autora.

Vamos entender linha a linha do código acima:

- **Linha 1 – <!DOCTYPE html>:** o DOCTYPE não é uma tag, mas, sim, uma instrução que informa ao navegador qual é a versão da linguagem utilizada, que neste caso é o HTML5;
- **Linhas 2 e 10:** indicam a abertura e fechamento da tag <html>, considerada uma tag raiz, pois envolve todo o conteúdo da página e delimita o nosso documento HTML. Além disso, na tag <html> utilizamos o atributo “lang”, utilizado para definir a linguagem principal do documento, que no nosso caso é: “pt-br”.

- **Linhas 3 e 6:** indicam a abertura e fechamento da tag <head>, área do código responsável por representar a coleção de metadados do documento, também conhecida como cabeçalho. Nela podemos definir o título que ficará visível na barra de título do navegador, o conjunto de caracteres do documento, o link para uma folha de estilo etc. Vale ressaltar que, o conteúdo inserido aqui não é exibido na página para o usuário;
- **Linha 4:** neste caso, a tag <meta> está sendo utilizada para definir o conjunto de caracteres da sua página através do atributo charset= "UTF-8". Com isso, o navegador consegue entender qual é o formato de codificação de caracteres utilizado naquele documento e renderizar corretamente o texto que será exibido na página;
- **Linha 5:** a tag <title> é responsável por definir o título que será exibido na guia do navegador escolhido, além disso, o texto descrito na tag será utilizado para descrever a página, caso ela seja adicionada nos seus "favoritos";
- **Linhas 7 e 8:** indicam a abertura e fechamento da tag <body> que representa o corpo da sua página. Nela inserimos todo o conteúdo (texto, vídeo, imagem...) que será exibido na página para o usuário.

Bom, essas são as tags principais para criarmos qualquer página HTML, mas você deve estar se perguntando: "Como insiro uma imagem, um link ou até mesmo uma lista no meu documento de HTML?" É simples! Vejamos:

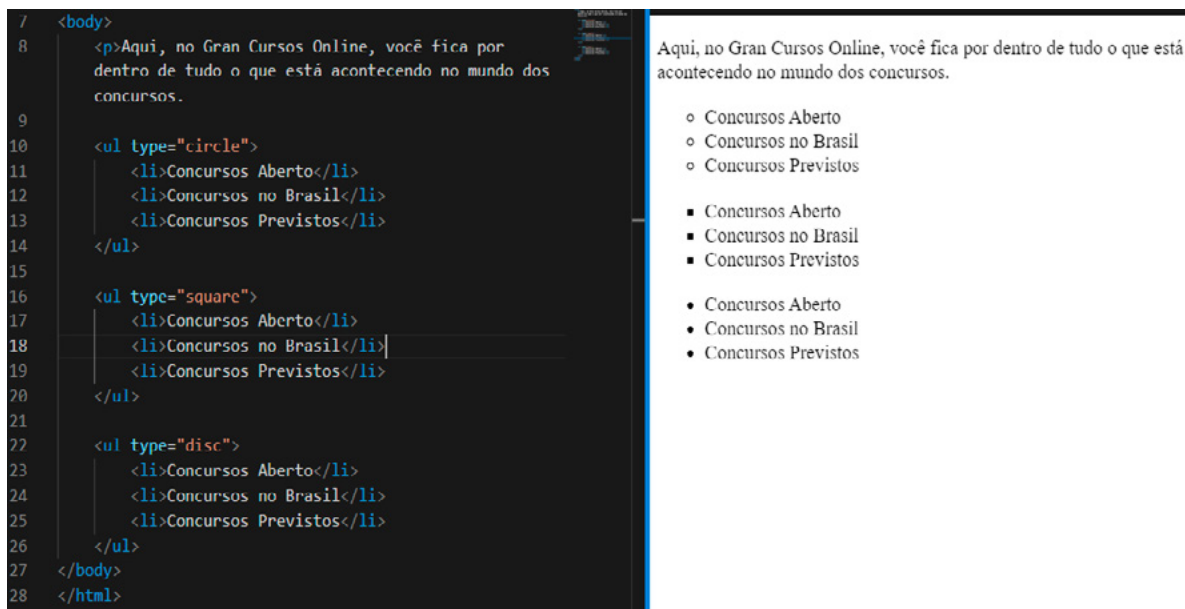
- **Imagem:** para inserir uma imagem devemos utilizar a tag , `` onde o atributo src (source) é responsável por apontar o caminho da imagem que você deseja incorporar na sua página e o atributo alt (alternative) é responsável por especificar um texto alternativo para os usuários que recorrem a leitores de tela para acessar páginas web. Além disso, quando a imagem não pode ser carregada por algum motivo, é o texto descrito no  Logotipo do nosso curso atributo **alt** que aparece na tela para o usuário ();
- **Link:** para inserir um link devemos utilizar a tag <a>. Por que a tag se chama "a" e não "link"? "A" é a forma abreviada de âncora. A tag <link> é utilizada para especificar, por exemplo, a relação de um documento HTML com um documento de CSS externo. Veja um exemplo de como criar um link para a página do **Gran Cursos Online**:

```
<a href="https://www.grancursosonline.com.br/">Estude para Concursos Públicos com o Gran Cursos Online</a>
```

Na página que será exibida no navegador, o texto que está entre as tags <a> aparecerá como link [Estude para Concursos Públicos com o Gran Cursos Online](https://www.grancursosonline.com.br/).

- **Lista:** para inserir uma lista podemos utilizar as tags ou . A tag cria uma lista não ordenada (marcadores) e a tag cria uma lista ordenada. Em ambos os casos utilizaremos a tag para criar os itens da lista. Vejamos:

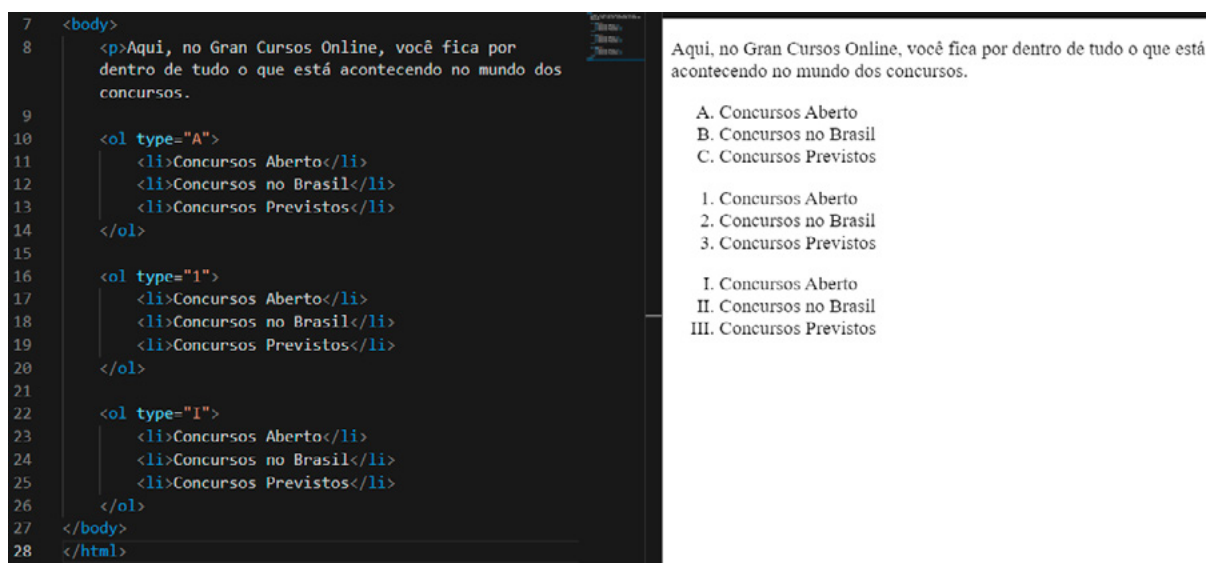
FIGURA 3 | Lista não ordenada



Fonte: Elaborado pela autora.

A Figura 3 ilustra uma lista não ordenada com 3 tipos de marcadores diferentes, são eles: circle ○, square ▪ e disc • (default). Para escolher um dos marcadores, basta adicionar na tag o atributo **type**.

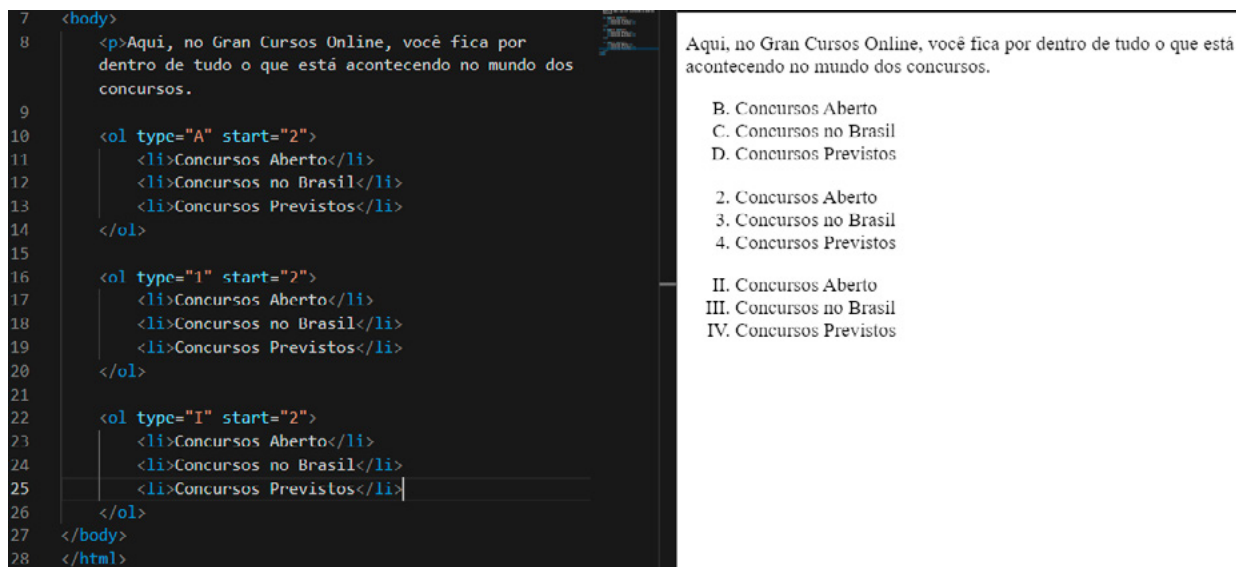
FIGURA 4 | Lista ordenada



Fonte: Elaborado pela autora.

A Figura 4 ilustra uma lista ordenada, para escolher o tipo de ordenação que a sua lista terá basta adicionar na tag o atributo **type** e escolher um dos tipos disponíveis, são eles: "A", "a", "1", "I" ou "i". Além disso, também é possível escolher por onde a sua lista começará, através do atributo **start**. Vejamos (Figura 5):

FIGURA 5 | Lista ordenada com atributo start

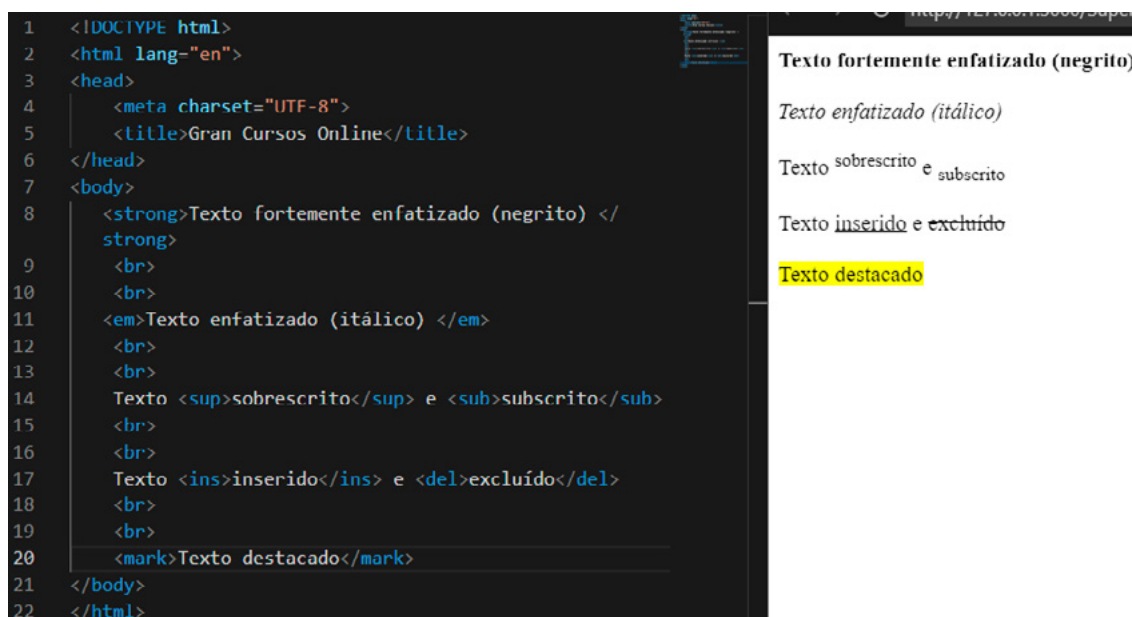


Fonte: Elaborado pela autora.

O atributo **start** só pode receber valores numéricos, sendo assim, mesmo que a lista escolhida por você seja uma lista alfabética, o valor atribuído para o atributo **start** deve ser um número, como mostra a Figura 5.

Agora vamos conhecer algumas tags para formatação de texto: , , <sup>, <sub>, <ins>, e <mark>. Veja a função de cada uma na Figura 6.

FIGURA 6 | Tags básicas para formatação de texto



Fonte: Elaborado pela autora.

Vamos encerrar este capítulo de HTML falando sobre as tags semânticas que definem as três principais partes de uma página, cabeçalho, corpo principal e rodapé. Tais tags já foram citadas anteriormente, são elas: <header>, <nav>, <article>, <aside>, <section>, <footer>.

- **<header>**: representa o cabeçalho de um documento HTML ou até mesmo de uma seção <section>. Nele podemos agrupar índices de conteúdo (h1, h2, h3, h4, h5, h6), inserir logotipo, inserir campo de busca e também incluir a lista de navegação que será definido pela tag <nav>. Exemplo de uso:

```

<header>
  <h1>Gran Cursos Online</h1>
  <h2>O Gran aposta tudo na sua aprovação</h2>
</header>

```

- **<nav>**: como mencionado acima, a tag <nav> é responsável por representar o menu de navegação da página, muita das vezes criado pelas tags , e <a>. Vale ressaltar que, o <nav> pode ser usado em qualquer lugar do documento que contenha uma lista de links, seja no <header>, no <aside> ou no <footer>. Exemplo de uso:


```
<header>
  <nav>
    <ul>
      <li><a href="#">Cursos</a></li>
      <li><a href="#">Questões</a></li>
      <li><a href="#">Pós-graduação</a></li>
      <li><a href="#">Blog</a></li>
    </ul>
  </nav>
</header>
```

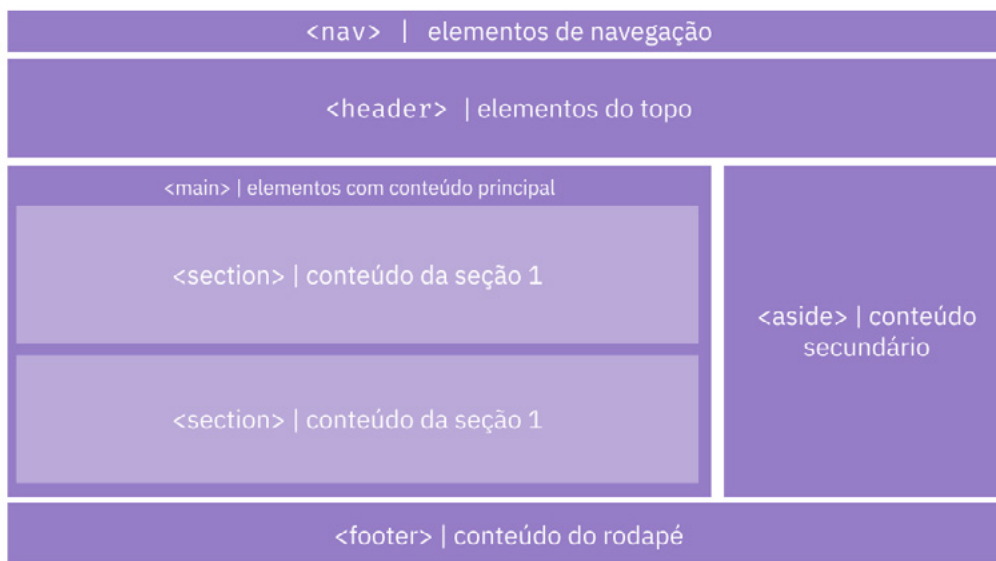
- **<article>**: responsável por representar um “conteúdo que não precisa de outro para fazer sentido em um documento HTML” (DEVMEDIA, 2017). Muito utilizado para representar artigos de blog, *posts*, blocos de comentários etc.;
- **<aside>**: responsável por representar um conteúdo adicional da sua página, como, por exemplo, uma área de publicidade, destacando informações que não fazem parte, propriamente dito, do conteúdo principal. Exemplo de uso:

```
<aside>
  <nav>
    <ul>
      <li><a>Link 1</a></li>
      <li><a>Link 2</a></li>
      <li><a>Link 3</a></li>
    </ul>
  </nav>
</aside>
```

- **<section>**: responsável por representar uma seção e/ou tópicos de um documento HTML. Por exemplo, pode utilizar a <section> para dividir uma página nas seguintes seções: introdução, destaque, novidades e informações de contato;
- **<footer>**: responsável por representar o rodapé de um documento HTML. Nele inserimos informações como: links para redes sociais, informações de autoria, contatos etc. Mesmo sendo a tag responsável por representar o rodapé, o <footer> pode ser inserido em qualquer lugar do documento, não necessariamente no fim dele.

Observe a Figura 7 e veja como esses elementos semânticos podem ser organizados em uma página HTML.

FIGURA 7 | Exemplo de organização dos elementos semânticos



Fonte: Webdev Book (<https://webdev.jesielviana.com/frontend/css>).

Vale ressaltar que, isso não é uma regra e sim um exemplo, cada um irá organizar os seus elementos semânticos segundo o seu contexto e sua necessidade.

CONSIDERAÇÕES FINAIS

Nesta aula aprendemos os principais conceitos associados a uma linguagem de marcação padrão (HTML) para criação e manutenção de páginas Web. Além disso, aprendemos também as tags semânticas e a sua importância no sentido de melhorar a visibilidade da sua página pelos motores de busca. Em seguida aprenderemos como estilizar (CSS) páginas HTML e como tornar o seu conteúdo mais dinâmico (JavaScript).

MATERIAIS COMPLEMENTARES

Tags semânticas do HTML5:

<https://youtu.be/pKOp7pQQzNM>

W3C HTML:

<https://www.w3schools.com/html/default.asp>

Semântica no HTML5? Que elementos são?

https://youtu.be/Afy_Hzyo0mQ

REFERÊNCIAS

CLARK, Richard; STUDHOLME, Oil; MURPHY, Christopher; MANIAN, Divya. *Introdução ao HTML5 e CSS3: a evolução da Web*. Rio de Janeiro, RJ: Alta Books, 2014.

FERREIRA, Elcio; EIS, Diego. *HTML5: Curso W3C Escritório Brasil*, 2015. Disponível em: <<https://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>> Acesso em: 26 de out. de 2022.

HTML Semântico: Conheça os elementos semânticos da HTML5. *DevMedia*, 2017. Disponível em: <<https://www.devmedia.com.br/html-semantico-conheca-os-elementos-semanticos-da-html5/38065>>. Acesso em: 31 de out. de 2022.

NOVO roteiro para o futuro das publicações está em andamento: W3C e IDPF estão agora oficialmente unificados. *W3C Brasil*, 2017. Disponível em: <<https://www.w3c.br/Noticias/NovoRoteiroParaFuturoDasPublicacoes>>. Acesso em: 26 de out. de 2022.

AULA 2 – CSS3

OBJETIVO DA AULA

Conhecer as principais propriedades que compõem a Folha de Estilos em Cascata – CSS (*Cascading Style Sheets*).

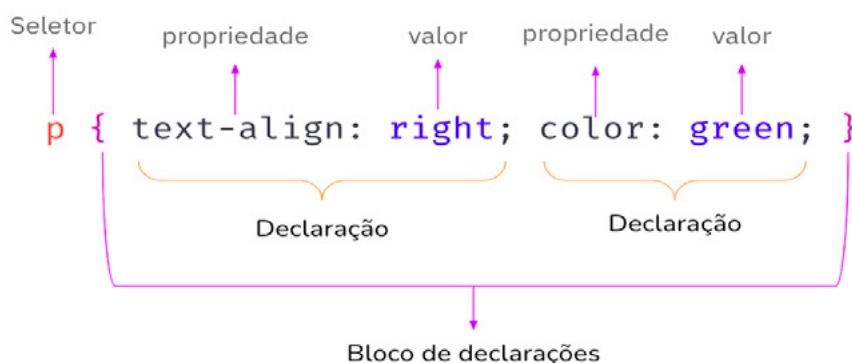
APRESENTAÇÃO

Uma página ou um site feito apenas com HTML não é muito bonito, certo? Antes do CSS, toda a estilização tinha que ser incluída na marcação HTML, utilizando tags e atributos que caíram em desuso com a chegada do HTML5. Agora que já aprendemos alguns dos elementos que compõem a linguagem HTML, chegou o momento de aprendermos como estilizar estes elementos. A Folha de Estilos em Cascata (CSS) é uma linguagem baseada em regras utilizada para aplicar estilos (cor de fundo, borda, cor de texto, tamanho da fonte...) ao seu documento. As características visuais aplicadas aos elementos são representadas pelo que chamamos de propriedades e essas propriedades podem ser combinadas, trazendo possibilidades de personalização quase infinitas. Ao longo desta aula você vai aprender as principais **propriedades** do CSS.

CSS (CASCADING STYLE SHEETS) – FOLHA DE ESTILO EM CASCATA

“Um estilo é essencialmente uma regra que instrui ao navegador como formatar algo em uma página web” (MCFARLAND, 2015). É como se o estilo estivesse falando o seguinte para o navegador: “Ei, navegador, faça isso se parecer com isto.” A estrutura da sintaxe do CSS é bem simples, porém, existem algumas regras que você precisa saber.

FIGURA 1 | **Sintaxe CSS**



Fonte: Webdev Book (<https://webdev.jesielviana.com/frontend/css>).

Livro Eletrônico

Como pode ser visto na Figura 1, sua sintaxe é composta por um seletor (elemento HTML que você deseja estilizar) e um bloco de declaração. O bloco de declarações pode conter ter uma ou mais declarações separadas por ponto e vírgula (;) e cada declaração contém uma propriedade CSS e um valor, separados por dois pontos (:). Resumindo, você seleciona um elemento e depois declara que tipo de efeito você deseja aplicar nele. Simples, não é mesmo? O exemplo da Figura 1, está definindo o alinhamento do texto à direita e a cor do texto verde para todos os parágrafos presentes no seu documento de HTML. Agora vamos entender detalhadamente cada um dos itens que compõem a sintaxe do CSS:

- **Seletor:** responsável por informar ao navegador qual elemento de HTML será estilizado. O CSS possui uma ampla variedade de seletores e sua principal função é “encontrar” o elemento HTML com base no nome do elemento (tag), da classe (class), do identificador (id) etc. Observe na Tabela 1 os tipos mais comuns de seletores:

TABELA 1 | **Seletores mais comuns**

Seletor	Exemplo	O que ele seleciona
tag ou elemento	p { }	Todos os elementos HTML de determinado tipo (tag). Qualquer tag/elemento pode ser usada como seletor.
class	.my-class { }	O(s) elemento(s) na página com a classe especificada. Vários elementos podem possuir a mesma classe. O seletor de classe inicia com um ponto (.).
id	#my-id { }	O(s) elemento(s) na página com o id especificado. Só deve haver um elemento com o mesmo id por página HTML. O seletor de id inicia com uma hashtag (#).
pseudoclassee	a:hover { }	O(s) elemento(s) especificado(s), mas somente quando estiver no estado especificado. No exemplo é utilizado o hover que é um estado ativado quando o mouse está sobre o elemento. O seletor de pseudoclassee é definido com o nome do seletor, dois pontos e o estado (ex.: seletor:estado).

Fonte: Webdev Book (<https://webdev.jesielviana.com/frontend/css>).

- **Bloco de declarações:** delimitado por {}, o bloco de declarações é responsável por incluir todas as opções de formatação que você deseja aplicar a um determinado seletor;
- **Declaração:** indica qual é a instrução de formatação através da propriedade e do valor associado a ela;

- **Propriedade:** css oferece uma ampla variedade de propriedades. Uma “propriedade é a forma pela qual você estiliza um elemento HTML. Cada propriedade está relacionada a uma característica que pode ser modificada no elemento HTML” (WEBDEV BOOK, 2019);
- **Valor:** “o valor da propriedade é a forma pela qual você define o estilo para determinada propriedade, afetando diretamente a aparência do elemento HTML” (WEBDEV BOOK, 2019).

Vale ressaltar que, uma única propriedade não irá fazer milagre e transformar a sua página em uma verdadeira obra de arte, o segredo aqui é combinar diferentes propriedades para obter páginas com um design incrível.

Aprenda mais sobre as propriedades e suas variações no site oficial da MDN (<https://developer.mozilla.org/pt-BR/docs/Web/CSS/Reference>). Acesso em: 05/01/2023.). que separou e organizou as propriedades em ordem alfabética.

LINK



TRÊS MANEIRAS DE VINCULAR O CSS AO HTML

Daí você deve estar se perguntando: como devo vincular o CSS a um documento HTML? Basicamente, existem três maneiras diferentes de fazer isso, são elas:

- **Inline:** *inline* significa em linha, então, neste modo as regras são declaradas dentro da tag através do atributo **style**. Vejamos: `<h1 style = "font-family: arial; text-align: center"> Gran Cursos </h1>`. Portanto, esta não é uma boa prática, pois mistura o código de CSS com HTML, tornando mais complexo a reutilização, evolução e manutenção do mesmo;
- **Interno ou Incorporado:** neste modo as regras são declaradas na área comportamental do nosso código HTML, ou seja, na tag <head> através da tag <style>. Vejamos:

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Gran Cursos Online</title>
6      <style>
7          h1{
8              font-family: arial;
9              text-align: center;
10         }
11     </style>
12 </head>

```

Portanto, também não é considerado uma boa prática, pelos mesmos motivos descritos no inline. Imagine um site com muitas páginas, para padronizá-las seria necessário copiar e colar a folha de estilo interna em cada uma das páginas. Chato, né? Imagine se você resolve trocar, por exemplo, a fonte escolhida inicialmente, você precisaria editar o estilo em todas as páginas. Pior ainda, não é mesmo? Sendo assim, a melhor solução para estes problemas citados acima é utilizar a folha de estilo externa.

- **Externo ou Importado:** neste modo as regras são declaradas em um arquivo.css, ou seja, separadas do arquivo HTML. **ATENÇÃO:** aqui não é necessário utilizar <style>. Podemos vincular uma folha de estilo a uma página HTML de duas formas diferentes:

1) **tag <link>:** `<link rel="stylesheet" href="css/estilo.css">`, onde **rel="stylesheet"** indica o tipo de link, que neste caso é uma folha de estilo e o **href** indica a localização do arquivo.css;

2) **diretiva @import:** a diretiva deve ser adicionada dentro da tag <style> HTML. Vejamos:

```

<style>
    @import url(css/estilo.css);
</style>

```

Ambos fazem a mesma coisa, porém a tag <link> é a mais utilizada; é uma questão de preferência.

PRINCIPAIS PROPRIEDADES DE TEXTO

Agora que já sabemos como integrar o CSS no HTML, aprenderemos as principais propriedades para a formatação de texto:

- **color:** altera a cor do texto. Podemos mencionar a cor de 4 maneiras diferentes: pelo nome em inglês, código em hexadecimal, código em RGB e código RGBA, o mais novo recurso do CSS3. Vejamos:

nome	color: blue; Exemplo
hexadecimal	color: #0000FF; Exemplo
RGB	color: RGB(0,0,255); Exemplo
RGBA (red, green, blue, alpha)	color: RGBA(0,0,255,0.5); Exemplo

Repare que o RGBA é muito parecido com o RGB, seu diferencial é o canal alpha que permite controlar a opacidade (transparência) da cor. O valor assumido pelo canal alpha pode variar de 0 (transparente) a 1 (opaca).

- **font-family:** define uma lista de fontes que será utilizada para estilizar o elemento. Os nomes das fontes escolhidas devem estar separados por vírgulas, utilizando a ordem de preferência. Vejamos:

```
<p style="font-family: Trebuchet MS, Arial, Times, sans-serif;">Gran Cursos</p>
```

Se o usuário tiver a fonte Trebuchet MS, o navegador irá usá-la; caso contrário, usará a Arial; caso contrário, usará a Times; e caso o usuário não tenha nenhuma destas fontes instaladas, a fonte padrão sem serifa do sistema operacional será utilizada. Uma fonte sem serifa é aquela que não possui os prolongamentos nas suas extremidades.

- **font-size:** define o tamanho da fonte. Esta propriedade permite tais unidades: px (pixels), em (unidade relativa), pt (pontos), %, entre outras;
- **font-weight:** define a intensidade ou espessura da fonte. Esta propriedade pode ter os seguintes valores: **normal** | **bold** | **bolder** | **lighter**, entre outros;
- **font-style:** define o estilo da fonte. Esta propriedade pode ter os seguintes valores: **normal** | **italic** | **oblique**;

- **font-variant:** transforma em maiúsculas de menor altura (**GRAN CURSOS**). Esta propriedade pode ter os seguintes valores: **normal** | **small-caps**;

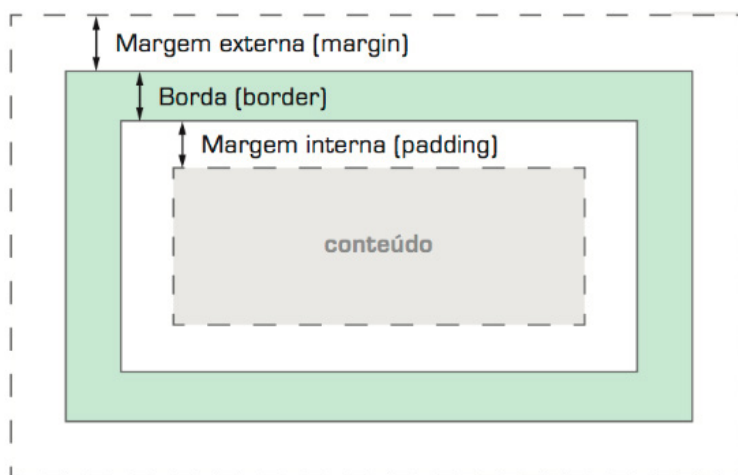
```
<p style="font-variant: small-caps">Gran Cursos</p>
```

- **line-height:** define o espaçamento entre linhas. Nesta propriedade é comum o uso da unidade **em**, pois com ela deixamos o espaçamento proporcional ao tamanho da fonte, por exemplo, `line-height: 2.5em`; (2.5 vezes o tamanho da fonte);
- **text-indent:** define a indentação (ou tabulação) da primeira linha do texto, por exemplo, `text-indent: 10px`;
- **text-transform:** define efeitos de capitalização do texto, são eles: caixa alta (*uppercase*), caixa baixa (*lowercase*) ou primeiras letras das palavras em maiúscula (*capitalize*);
- **Text-decoration:** define uma linha decorativa para o texto. Esta propriedade pode ter os seguintes valores: *overline* (superlinhado), *underline* (sublinhado), *line-through* (riscado), *none* (sem efeito). Vejamos:

<code><p style="text-decoration: overline;"> Gran Cursos </p></code>	<u>Gran Cursos</u>
<code><p style="text-decoration: underline;"> Gran Cursos </p></code>	<u>Gran Cursos</u>
<code><p style="text-decoration: line-through;"> Gran Cursos </p></code>	Gran Cursos
<code><p style="text-decoration: none;"> Gran Cursos </p></code>	Gran Cursos

Outra coisa que devemos considerar é que no HTML tudo é uma caixa (box). Como assim? “A maioria dos elementos HTML podem ser pensados como caixas, que podem ser agrupadas de forma horizontal, vertical ou colocada dentro de outra” (WEBDEV BOOK, 2019).

FIGURA 2 | Caixa de um elemento HTML



Fonte: Webdev Book (<https://webdev.jesielviana.com/frontend/css>)

O *box* (caixa) que representa um elemento HTML (Figura 9) é composto por conteúdo, margem interna (*padding*), borda (*border*) e margem externa (*margin*). Todas essas partes que compõe o elemento é conhecida como *box model* (WEBDEV BOOK, 2019).

Além das propriedades de texto que aprendemos acima, vamos conhecer também algumas propriedades muito utilizadas para estilizar o espaço que o elemento ocupa, são elas (WEBDEV BOOK, 2019):

- **width** – define a largura de um elemento que pode ser definido com valor fixo usando pixel (px) ou valor dinâmico usando porcentagem (%);
- **height** – define altura de um elemento que pode ser definido com valor fixo usando pixel (px) ou valor dinâmico usando porcentagem (%);
- **margin** – define o espaçamento externo de um elemento;
- **border** – define a borda do elemento;
- **padding** – define o espaçamento interno, ou seja, espaço entre a borda e o conteúdo;
- **background-color** – define a cor de fundo do elemento.



DESTAQUE

Quando as propriedades *padding*, *margin* e *border* recebem um único valor (*padding: 2px;*) alteram os quatro lados de um elemento: topo, direita, baixo e esquerda, porém é possível especificar o valor de forma individual e separadamente (*padding-top: 2px; padding-right: 4px; padding-bottom: 6px; padding-left: 8px;*) ou em conjunto (*padding: 2px 4px 6px 8px;*), passando os 4 valores de uma vez.

CONSIDERAÇÕES FINAIS

Nesta unidade aprendemos a sintaxe utilizada na linguagem CSS, como ela pode ser aplicada e as suas principais propriedades. Além disso, vimos que o CSS foca em toda a parte estética, enquanto o HTML foca na estrutura e na semântica do documento. Vimos também que existe uma forte relação entre o HTML e o CSS, porém, é importante ressaltar que é uma boa prática separar o conteúdo da representação visual do site, pois isso facilita a reutilização, a evolução e a manutenção do código.

MATERIAIS COMPLEMENTARES

Lista de propriedade em CSS:

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/Reference>

Principais Propriedades de Texto CSS:

<https://youtu.be/vtP3Vqo2Kog>

Aprenda Construir Layouts Elegantes com CSS Grid e Flexbox:

https://youtu.be/1mf4mZE9°_4

O que é CSS? Guia Básico para Iniciantes:

<https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css>

Apostila de CSS – Curso W3C Escritório Brasil:

<https://www.w3c.br/pub/Cursos/CursoCSS3/css-web.pdf>

REFERÊNCIAS

CSS. *Webdev Book*, 2019. Disponível em: <<https://webdev.jesielviana.com/frontend/css>>. Acesso em: 02 de nov. de 2022.

MCFARLAND, David Sawyer. *CSS3: o manual que faltava*. Rio de Janeiro, RJ: AltaBooks, 2015.

AULA 3 – ALGORITMOS E LINGUAGEM JAVASCRIPT

OBJETIVO DA AULA

Entender a importância de um algoritmo no mundo da programação.

APRESENTAÇÃO

Certamente, você já deve ter ouvido falar de algoritmo em algum momento de sua vida. Segundo o dicionário online DICIO, um algoritmo é um “conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas” (ALGORITMO, 2022). A grande questão é que os algoritmos são escritos em linguagem natural e o computador não consegue entender a nossa linguagem, daí foram desenvolvidas as linguagens de programação que servem de “ponte” entre nós (humanos) e as máquinas. Resumindo, “um algoritmo escrito em Linguagem Natural passa a ser chamado de Programa depois de convertido para uma linguagem aceita por um computador”, que é a linguagem de máquina (BARRETO et al., 2015). Ao longo desta aula vamos aprender o que é um algoritmo e a linguagem de programação JavaScript.

ALGORITMOS E LINGUAGEM JAVASCRIPT

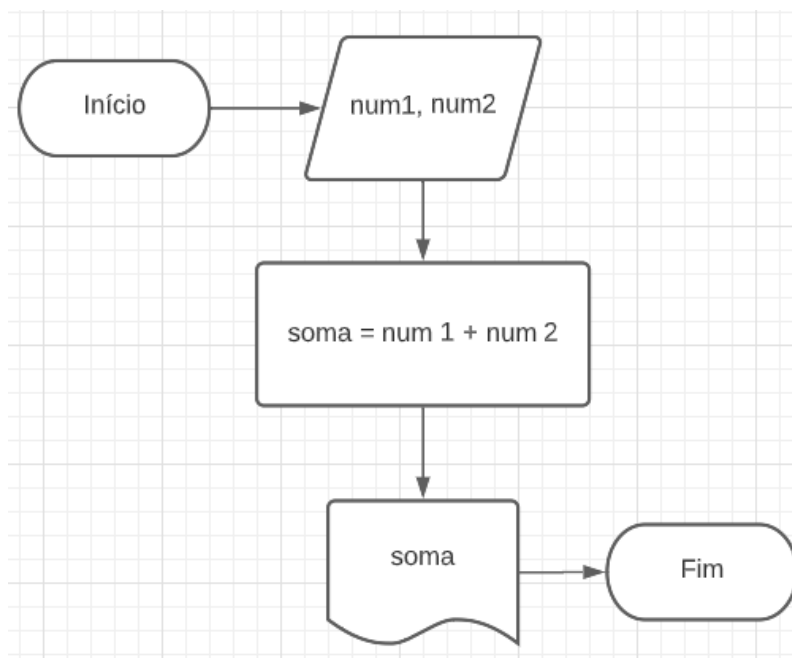
Você sabia que os algoritmos fazem parte do nosso cotidiano? Sim, é verdade. Eles estão presentes nos motores de busca, nos anúncios tendenciosos, nas redes sociais e até mesmo nas tarefas mais simples do nosso dia a dia. Já parou para pensar que uma receita de bolo é um algoritmo? É sério! Uma receita de bolo nada mais é do que um conjunto definido de instruções (pré-aquecer o forno, misturar os ingredientes, despejar na assadeira e assar), logo, é considerado um algoritmo. E o que isso tem a ver com a nossa área de conhecimento? Algoritmos computacionais realizam ações automatizadas tendo como insumo valores de entrada e produzindo valores de saída. Vejamos abaixo as três formas diferentes de representar um algoritmo:

- **Descrição Narrativa:** algoritmo escrito em linguagem natural, porém a linguagem natural abre espaço para má interpretação, ambiguidade e imprecisão, dificultando a transição deste algoritmo para um programa. Exemplo – Algoritmo para mostrar o resultado da soma de dois números:

- 1) Obter os dois números que serão somados;
- 2) Somar os dois números;
- 3) Mostrar o resultado obtido na soma dos dois números.

Livro Eletrônico

- **Fluxograma:** o algoritmo é representado por símbolos gráficos predefinidos, ou seja, é representado por uma linguagem visual. Como o fluxograma não é rico em detalhes e regras, também pode dificultar a transição deste algoritmo para um programa. Exemplo – Algoritmo para mostrar o resultado da soma de dois números:



Saiba mais detalhes sobre os símbolos do fluxograma aqui: <https://www.edrawsoft.com/pt/explain-algorithm-flowchart.html>. Acesso em: 05/01/2023.

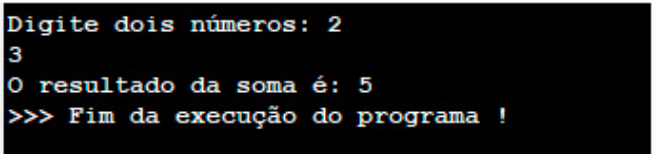
LINK



- **Pseudocódigo ou Portugal:** o algoritmo é representado através de uma linguagem própria que se aproxima de uma linguagem de alto nível. O pseudocódigo:
(...) consiste na descrição estruturada, por meio de regras pré-definidas, de passos (ou instruções) a serem realizados para a resolução do problema, utilizando a linguagem natural para representar o raciocínio (LEITE, 2015).

Exemplo – Algoritmo para mostrar o resultado da soma de dois números:

```
1 Algoritmo "semnome"
2 Var
3 num1, num2: inteiro
4 soma: inteiro
5 Inicio
6   escreva("Digite dois números: ")
7   leia(num1, num2)
8   soma <- num1 + num2
9   escreva("O resultado da soma é:", soma)
10 Fimalgoritmo
11
```



Vamos entender linha a linha do código acima:

- **Linha 1:** região para nomear o algoritmo que está sendo criado;
- **Linhas 2, 3 e 4:** região de definição de variáveis. As variáveis **num 1** e **num 2** serão utilizadas para armazenar os valores que serão inseridos pelo usuário, enquanto a variável **soma** irá armazenar o resultado da soma dos dois números. Repare que todas as variáveis foram declaradas como inteiro, isso significa que elas só receberão números inteiros, ou seja, números sem casas decimais;
- **Linha 5:** indica o início do algoritmo em português;
- **Linha 6:** indica que será mostrado uma mensagem na tela para o usuário;
- **Linha 7:** lê os valores que foram inseridos pelo usuário e os armazena nas variáveis **num1** e **num2**;
- **Linha 8:** aqui foi utilizado um operador de atribuição (<-) para atribuir o resultado da soma dos dois números para a variável **soma**;
- **Linha 9:** mensagem com o resultado para o usuário;
- **Linha 10:** indica o fim do algoritmo em português.

Agora que já sabemos o que é um algoritmo e as formas de representá-lo, precisamos estabelecer um mecanismo de comunicação com o computador, para isso, vamos construir programas a partir de algoritmos. Para construir programas utilizaremos linguagens de programação, um pouco mais complexas para nós (humanos), porém mais fáceis de serem entendidas pelo computador. A linguagem de programação que aprenderemos é o JavaScript.

JAVASCRIPT

JavaScript é uma linguagem de programação voltada para web. Para trabalhar com desenvolvimento web você precisa basicamente saber HTML (utilizado para definir o conteúdo da página), CSS (utilizado para especificar a apresentação da página) e JavaScript (utilizado programar o comportamento da página). É uma linguagem de alto nível, interpretada, dinâmica e não tipada. O que significa ser não tipada? Significa que JavaScript é um tipo de linguagem que permite que você declare uma variável sem definir o seu tipo.

Vale ressaltar que, JavaScript é completamente diferente da linguagem Java. Seu nome foi uma grande jogada de marketing das empresas Sun e Netscape, que causa confusão até hoje (FLANGAN, 2013). JavaScript segue a sintaxe básica do Java, porém é uma linguagem mais livre, como vimos anteriormente não é necessário declararmos todas as variáveis, classes e métodos. Além disso, JavaScript é uma linguagem interpretada, ou seja, é uma linguagem executada em tempo de execução, funciona como se fosse um intérprete fazendo tradução simultânea, onde cada linha é lida, analisada e convertida para o processador, uma após a outra.

De acordo com Flanagan (2013), JavaScript é uma linguagem orientada a objetos. Os objetos formam a base da programação em JavaScript, pois através deles é possível obter diversas informações sobre o ambiente onde a página que contém o *script* está sendo aberta ou mesmo modificar características desse ambiente. Os **objetos** são estruturas de dados mais complexas do que as variáveis e podem conter diversos valores de tipos diferentes ao mesmo tempo. Alguns desses valores armazenados em objetos são chamados de **propriedades**, que representam as suas características intrínsecas. As propriedades possuem valores associados a elas e alguns desses valores podem ser modificados, já outros podem apenas ser consultados. Além disso, os objetos podem conter os chamados **métodos**, funções que podem ser aplicadas sobre um determinado objeto, retornando um determinado resultado. Um método não pode ser modificado, pois grande parte dos objetos utilizados são fornecidos pelo navegador como parte da linguagem JavaScript.

Atualmente, JavaScript tanto pode ser usado para o desenvolvimento do lado do cliente (maneira como JavaScript funciona em seu navegador) quanto para o desenvolvimento do lado do servidor (uso da linguagem de codificação na lógica de *back-end* do servidor). Na próxima aula, aprenderemos um pouco mais sobre a linguagem JavaScript.

CONSIDERAÇÕES FINAIS

Nesta aula aprendemos o que é e qual a importância de um algoritmo. Vimos que é possível representá-lo de três formas diferentes: narrativa, fluxograma e pseudocódigo. Na narrativa mostramos o passo a passo num formato de texto para alcançar uma solução final, no fluxograma contamos com o auxílio de símbolos, formas e setas e no pseudocódigo usamos

uma linguagem própria que se aproxima de uma linguagem de alto nível. Vimos também que para construir programas precisamos de uma linguagem de programação, dentre todas as linguagens existentes, vamos nos aprofundar na linguagem JavaScript.

MATERIAIS COMPLEMENTARES

Apostila de Algoritmos e Programação:

<http://www.univasf.edu.br/~andreza.leite/aulas/AP/introducao.pdf>

Tutorial JavaScript:

<https://www.w3schools.com/js/>

Guia de JavaScript:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction>

REFERÊNCIAS

ALGORITMO. In: DÍCIO, Dicionário Online de Português. Porto: 7Graus, 2022. Disponível em: <<https://www.dicio.com.br/algoritmo/>>. Acesso em: 22 de out. de 2022.

BARRETO, Muniz Jorge; ALMEIDA, Fernandes Maria Aparecida; POZZEBON, Eliane. *Material de apoio das disciplinas: introdução à computação*, 2015. Disponível em: <<http://www.inf.ufsc.br/~j.barreto/cca/arquitet/arq3.htm>>. Acesso em: 03 de nov. de 2022.

FLANAGAN, David. *JavaScript: o guia definitivo*. Porto Alegre, RS: Bookman, 2013.

LEITE, Andreza. *Algoritmos e programação*, 2015. Disponível em: <<http://www.univasf.edu.br/~andreza.leite/aulas/AP/introducao.pdf>> Acesso em: 05 de nov. de 2022.

AULA 4 – VARIÁVEIS, COMANDOS DE ENTRADA/SAÍDA E OPERADORES

OBJETIVO DA AULA

Conhecer os comandos básicos do JavaScript.

APRESENTAÇÃO

Nesta aula aprofundaremos nosso conhecimento na linguagem JavaScript. Aprenderemos o conceito de variáveis e os comandos básicos de entrada e saída da linguagem. Além disso, conheceremos também os seus operadores básicos: aritméticos, relacionais e lógicos. Os operadores nos permitem criar expressões que serão avaliadas ao longo do desenvolvimento. É importante lembrar que nesta linguagem as instruções são chamadas de declaração e são sempre separadas por um ponto e vírgula (;). Além disso, JavaScript é **case sensitive**, isso significa que a linguagem diferencia maiúsculo de minúsculo (num1 é diferente de Num1), precisamos ficar atentos a este detalhe.

VARIÁVEIS

O que é uma variável? De acordo com Flanagan (2013):

(...) quando um programa precisa manter um valor para uso futuro, ele atribui o valor (ou “armazena” o valor) a uma variável. A variável define um nome simbólico para um valor e permite que o valor seja referido pelo nome.

As variáveis são espaços de memória RAM, vale ressaltar que, no caso do JavaScript, uma variável só existe no espaço de memória ocupado por uma página aberta no navegador. Quando a página exibida na janela do navegador é modificada, todas as variáveis criadas pelos scripts desta página são perdidas.

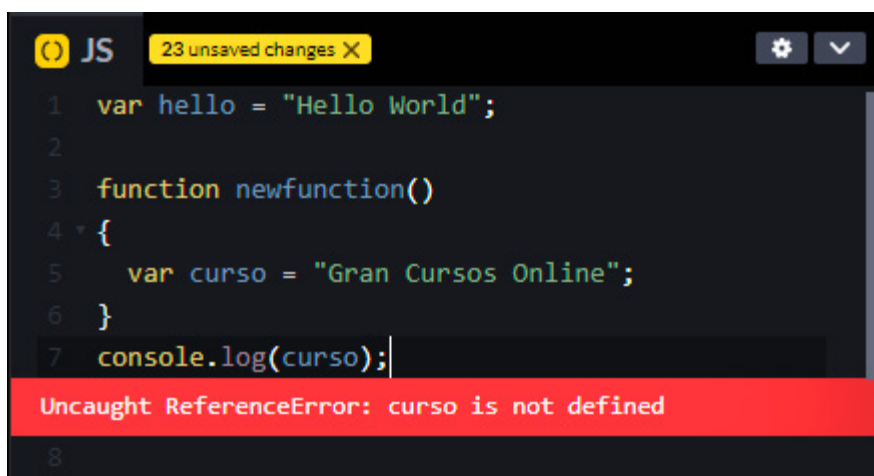
Temos a liberdade para criar os nomes das variáveis, mas precisamos ter atenção as seguintes regras:

- Os nomes de variável podem incluir letras do alfabeto, tanto minúsculas como maiúsculas, lembrando que JavaScript é *case sensitive*;
- Devem começar com uma letra, underline (_), ou cifrão (\$);
- Elas podem conter números, porém não pode começar por eles;
- Os nomes de variável não podem incluir espaços nem quaisquer outros caracteres de pontuação.

Livro Eletrônico

Podemos declarar uma variável de três formas diferentes, são elas:

- **var**: muito utilizada antes da versão EcmaScript 6. Por conta de alguns problemas com o **var**, a nova versão trouxe novas maneiras de declarar variáveis (*let* e *const*). Declarações com **var** tem escopo global ou escopo de função/local. Além disso, variáveis declaradas como **var** podem ser declaradas de novo e atualizadas. Vamos entender o que isso significa.... No exemplo abaixo, podemos observar que a variável **hello** tem o seu escopo global, pois está fora de uma função, enquanto a variável **curso** tem o seu escopo de função/local, ou seja, não pode ser acessada fora da função. Vejamos:



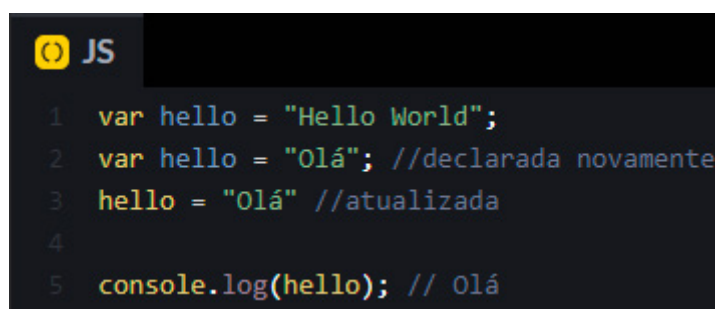
```

1  var hello = "Hello World";
2
3  function newfunction()
4  {
5      var curso = "Gran Cursos Online";
6  }
7  console.log(curso);

```

Uncaught ReferenceError: curso is not defined

Observe que neste exemplo obtivemos um erro, que nos diz que a variável **curso** não está disponível fora da função *newfunction*. Além disso, variáveis declaradas como **var** podem ser declaradas novamente e atualizadas. Vejamos:

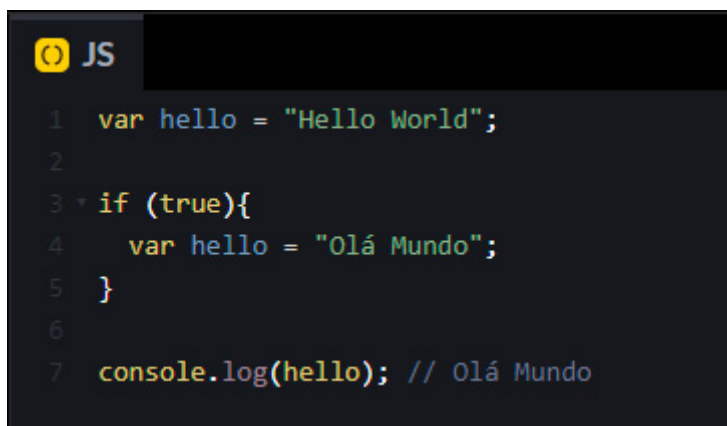


```

1  var hello = "Hello World";
2  var hello = "Olá"; //declarada novamente
3  hello = "Olá" //atualizada
4
5  console.log(hello); // Olá

```

O fato dela poder ser atualizada em qualquer parte do escopo e não gerar erro pode trazer alguns problemas. Vejamos:

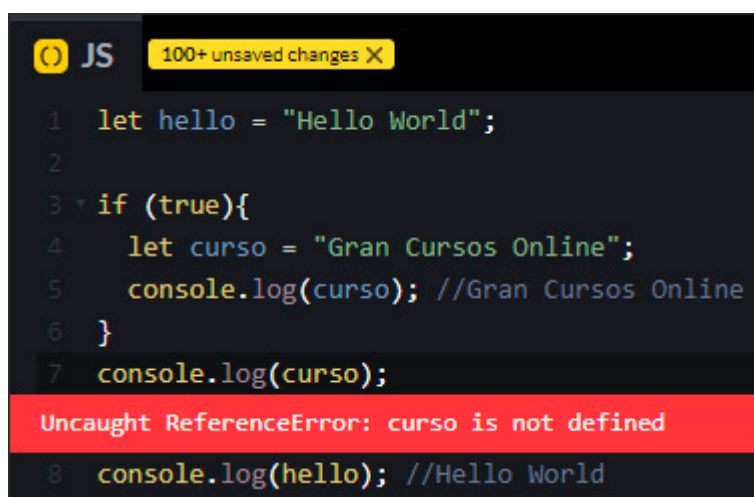


```

JS
1  var hello = "Hello World";
2
3  if (true){
4    var hello = "Olá Mundo";
5  }
6
7  console.log(hello); // Olá Mundo
  
```

Uma vez que a condição for verdadeira, a variável **hello** é redefinida como “Olá Mundo”. Isso pode não ser um problema, caso você queira que a variável **hello** seja redefinida. Agora, se você já usou a variável **hello** em outras partes do seu código, poderá ser surpreendido com o resultado que poderá obter. Portanto, tome cuidado! Isso pode causar muitos bugs no seu código, por esse motivo foram criadas outras maneiras de declarar como *let* e *const*.

- **let**: usada para declarar uma variável local de escopo de bloco. O que é um bloco? Um bloco nada mais é do que um porção de código envolto por {}. Sendo assim, uma variável declarada como **let** em um bloco, só estará disponível dentro daquele bloco. Vejamos:



```

JS 100+ unsaved changes X
1  let hello = "Hello World";
2
3  if (true){
4    let curso = "Gran Cursos Online";
5    console.log(curso); //Gran Cursos Online
6  }
7  console.log(curso);
Uncaught ReferenceError: curso is not defined
8  console.log(hello); //Hello World
  
```

Podemos observar que o uso da variável **curso** fora de seu bloco {} retornou um erro e o uso da variável **hello** não apresentou erro porque se encontra dentro do seu escopo de bloco. Assim como no **var**, a variável declara como **let** pode ser atualizada, porém, não pode ser declarada novamente.

```
JS
1 let hello = "Hello World";
2 hello = "Olá Mundo";
3 console.log(hello);
```

```
JS 100+ unsaved changes X
1 let hello = "Hello World";
2 let hello = "Olá Mundo";
Uncaught SyntaxError: Identifier 'hello' has already been declared
3 console.log(hello);
```

Diferentemente do uso do **var**, com o **let** se a mesma variável for definida em escopos diferentes, serão consideradas variáveis diferentes, já que são de escopos diferentes. Vejamos:

```
JS
1 let hello = "Hello World";
2
3 if (true){
4   let hello = "Olá Mundo";
5   console.log(hello); //Olá Mundo
6 }
7 console.log(hello); // Hello World
```

Isso faz com que o uso do **let** seja mais apropriado do que o uso do **var**, tendo em vista que você não precisa se preocupar se usou este nome de variável em algum outro momento, já que a variável existe somente dentro daquele escopo. Além disso, como uma variável não pode ser declarada mais de uma vez em um escopo, o problema que ocorre com **var** mencionado anteriormente, não acontece.

- **const**: a variável declarada como *const* possui seu valor constante, ou seja, seu valor não pode ser modificado. “Uma constante se comporta em relação ao escopo onde foi declarada da mesma forma que uma variável declarada como **let**”, ou seja, só pode ser acessada dentro do bloco que foi declarada (DEV MEDIA, 2020). A diferença é que uma constante não pode ser atualizada e nem declarada novamente.

```
JS 100+ unsaved changes X
1 const hello = "Hello World";
2 hello = "Olá";
Uncaught TypeError: Assignment to constant variable.
```

```
JS 100+ unsaved changes X
1 const hello = "Hello World";
2 const hello = "Olá";
Uncaught SyntaxError: Identifier 'hello' has already been declared
```

DESTAQUE

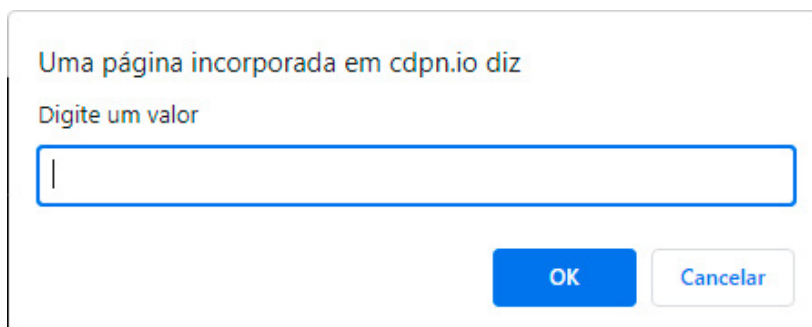
Resumindo:

(...) as declarações de **var** têm escopo global ou escopo de função enquanto **let** e **const** tem escopo de bloco. Variáveis **var** podem ser atualizadas e declaradas novamente dentro do seu escopo. Variáveis **let** podem ser atualizadas, mas não declaradas novamente. Variáveis **const** não podem ser atualizadas nem declaradas novamente (JESUS, 2021).

COMANDOS DE ENTRADA/SAÍDA

O objeto *document*, que representa a página propriamente dita, possui o método *write*, que juntos formam a instrução **document.write()** e possui a função de escrever informações na página para o usuário. Nela é possível escrever strings, conteúdo de variáveis e tags HTML. Qualquer expressão válida em JavaScript pode ser escrita com a utilização desse método. Logo, **documento.write** pode ser utilizado como método de saída. Além disso, podemos utilizar o **console.log()**, como visto nos exemplos anteriores, para acompanhar os valores de variáveis durante a execução de algum código.

Podemos utilizar o método **prompt()** como método de entrada. Este método faz exibir uma janela com uma mensagem e um campo aberto para que o usuário possa digitar informações.



A janela aberta pelo **prompt()** possui dois botões, *Ok* e *Cancelar*, usados para confirmar e cancelar a entrada de dados, respectivamente. A informação digitada pode ser armazenada em uma variável.

OPERADORES

“Os operadores são utilizados em JavaScript para expressões aritméticas, expressões de comparação, expressões lógicas, expressões de atribuição e muito mais” (FLANAGAN, 2013). Vejamos:

- **Operadores Aritméticos:** retornam o resultado da operação aritmética entre os valores.

Operador	Descrição	Exemplo
+	retorna a soma entre dois valores ou concatena strings	$a + b$
-	retorna a diferença entre o primeiro e o segundo valor	$a - b$
*	retorna o produto entre dois valores	$a * b$
/	retorna o quociente da divisão do primeiro valor pelo segundo	a / b
%	retorna o módulo (resto) da divisão inteira do primeiro valor pelo segundo	$a \% b$
-	retorna o inverso de uma variável	-a

- **Operadores Relacionais (comparação):** retornam um resultado booleano, ou seja, *true* ou *false*.

Operador	Descrição	Exemplo
==	verifica se dois valores são iguais entre si	a == b
!=	verifica se dois valores são diferentes entre si	a != b
>	verifica se o primeiro valor é maior que o segundo	a > b
>=	verifica se o primeiro valor é maior ou igual ao segundo	a >= b
<	verifica se o primeiro valor é menor que o segundo	a < b
<=	verifica se o primeiro valor é menor ou igual ao segundo	a <= b

- **Operadores lógicos:** efetuam comparações lógicas entre dois valores.

Operador	Descrição	Exemplo
&&	comparação do tipo “E”, retornando true caso os dois valores sejam verdadeiros e false caso pelo menos um dos valores seja falso	a && b
	comparação do tipo “OU”, retornando true caso pelo menos um dos valores seja verdadeiro e false caso os dois valores sejam falsos	a b
!	comparação do tipo “NÃO”, retornando true caso o valor seja falso e false caso o valor seja verdadeiro	!a

- **Operadores de atribuição:** são utilizados como forma de abreviar determinadas expressões.

Operador	Descrição	Exemplo
+=	atribui a primeira variável o valor da soma dela mesma com a segunda variável	a += b
-=	atribui a primeira variável o valor da subtração dela mesma com a segunda variável	a -= b
*=	atribui a primeira variável o valor do produto dela mesma com a segunda variável	a *= b
/=	atribui a primeira variável o valor da divisão dela mesma pela segunda variável	a /= b
%=	atribui a primeira variável o valor do módulo (resto) da divisão inteira dela mesma pela segunda variável	a %= b
++	acrescenta 1 a variável	a++ ou ++a
--	diminui 1 da variável	a-- ou --a

CONSIDERAÇÕES FINAIS

Nesta unidade aprendemos os principais conceitos de JavaScript: variáveis, comandos de entrada/saída e operadores. Vimos que variáveis podem ser declaradas de três formas diferentes. Variáveis declaradas como **var** podem ser atualizadas e declaradas novamente, desde que seja dentro do seu escopo. Variáveis declaradas como **let** podem ser atualizadas, porém, não podem ser declaradas novamente, e ao contrário do **var** e do **let**, as variáveis declaradas como **const** não podem ser atualizadas e muito menos declaradas novamente. Com esses conceitos já conseguimos fazer uns scripts básicos. Na próxima aula colocaremos esses conceitos em prática.

MATERIAIS COMPLEMENTARES

Operadores Aritméticos:

<https://youtu.be/wkradRZd93g>

Operadores Relacionais (comparação):

<https://youtu.be/hoPy34YaR4U>

Operadores Lógicos:

<https://youtu.be/ivGxVPnWSCA>

Var, Let e Const:

<https://youtu.be/ZOx7iTnBqFQ>

REFERÊNCIAS

FLANAGAN, David. *JavaScript: o guia definitivo*. Porto Alegre, RS: Bookman, 2013.

JAVASCRIPT: Variáveis e constantes. *DevMedia*, 2020. Disponível em: <<https://www.devmedia.com.br/javascript-variaveis-e-constantes/41012>> Acesso em: 05 de nov. de 2022.

JESUS, Gabriel. *Qual a diferença entre var, let e const no JavaScript?* 2021. Disponível em <<https://pt.linkedin.com/pulse/qual-diferen%C3%A7a-entre-var-let-e-const-javascript-gabriel-de-jesus>> Acesso em: 05 de nov. de 2022.

AULA 5 – CRIANDO SOLUÇÕES WEB

OBJETIVO DA AULA

Praticar os conceitos vistos até o momento.

APRESENTAÇÃO

Nesta aula colocaremos a mão na massa. Isso mesmo! Vamos fazer exemplos práticos juntos para testar os nossos conhecimentos em relação aos conceitos aprendidos nas aulas anteriores. Preparados? Vamos lá! O primeiro exemplo que faremos é o mesmo que usamos na aula de algoritmos, a ideia é mostrar o resultado da soma de dois números.

EXEMPLOS PRÁTICOS

EXEMPLO 1

Para criar o nosso primeiro exemplo: pedir dois números para o usuário e apresentar o resultado da soma destes números na tela, precisaremos de um documento de HTML e outro documento de JavaScript. Neste exemplo, vamos fazer com dois arquivos separados, mas você poderia fazer tudo no documento de HTML, porém esta não é uma boa prática.

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <title>Meu Primeiro Exemplo</title>
6  </head>
7  <body>
8      <input type="text" id="n1"> <br>
9      <input type="text" id="n2"> <br>
10     <p id="res">Resultado: </p>
11     <button onclick="Soma()"> Soma </button>
12
13     <script src="teste.js"></script>
14 </body>
15 </html>
```

Link para ter acesso ao código: <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/3be66e0788cdebdea8ac45dbaadd0b9e8f060c2/Livro%20Eletr%C3%B4nico/Unidade1Aula5a.>

No documento de HTML criamos a estrutura básica que aprendemos na aula 1. Dentro do `<body>`, nas *linhas 8 e 9* inserimos duas caixas de texto para que o usuário consiga digitar os 2 números que serão somados. Para isso, utilizamos a tag `<input>` com o atributo **type="text"**, essa combinação projeta na tela um campo de texto. O atributo *type* pode receber outros valores como: *password*, *button*, *radio* e entre outros.



LINK

Aprenda mais sobre os campos de formulário em: https://www.w3schools.com/html/html_forms.asp. Acesso em: 05/01/2023



Repare que em cada um dos inputs criamos um identificador (**id**) que deve ser único por todo o documento, é como se fosse o nosso CPF. E para que serve? Serve para identificarmos aquele elemento, seja utilizando scripts ou estilizando com CSS. Na *linha 10*, criamos um parágrafo `<p>` onde o resultado será exibido. Na *linha 11*, criamos um botão com a tag `<button>` e utilizamos também o evento **onclick = Soma()**, ou seja, ao clicarmos no botão SOMA ele “chamará” a função **Soma()**. Agora, você não precisa se preocupar em entender o que é e para que serve uma função, pois ainda estamos aprendendo os conceitos básicos. Em outro momento, falaremos da função em detalhes. O botão também poderia ter sido criado desta forma: `<input type="button" value="Soma" onclick="Soma()">`. Na *linha 13*, estamos associando o documento de HTML a um documento de JavaScript externo. Agora vamos analisar o nosso script:

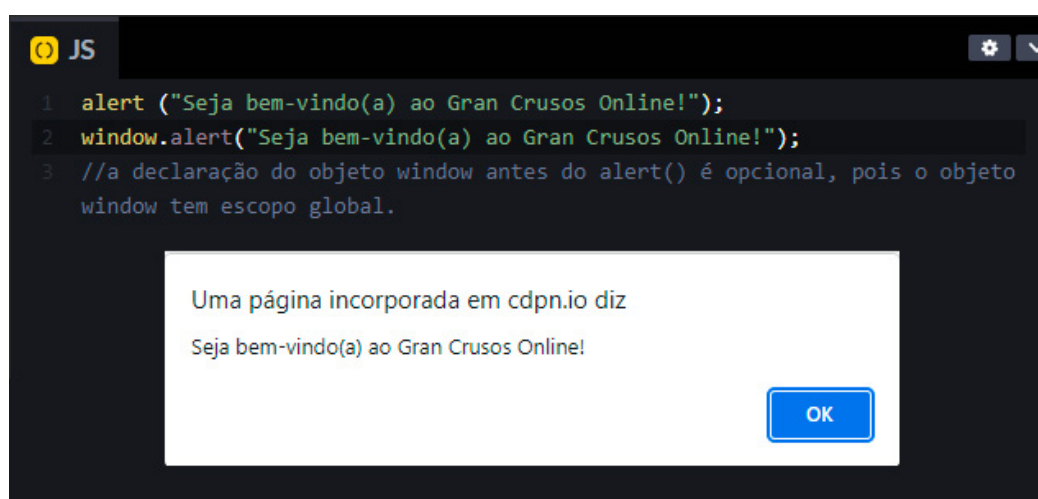
```
1 function Soma()
2 {
3     var num1 = parseInt(document.getElementById("n1").value);
4     var num2 = parseInt(document.getElementById("n2").value);
5     var soma = num1 + num2;
6     document.getElementById("res").innerHTML = "Resultado: " + soma;
7 }
```

No arquivo de JavaScript criamos uma função (*function*) chamada Soma() e nela somaremos os dois números digitados pelo usuário. O evento **onclick="Soma()"**, que vimos acima, executa a função Soma(). Nas *linhas 3 e 4*, estamos criando duas variáveis (num1, num2) que irão receber os valores digitados pelo usuário. A função **parseInt()** está sendo utilizada para converter a *string* recebida pela caixa de texto para um número inteiro. O método **getElementById()** está sendo utilizado para obter um elemento do documento a partir de seu atributo ID especificado, por isso colocaremos um id para cada um dos inputs no HTML. Resumindo, esta função está pegando os valores que foram digitados nos elementos que possuem os identificadores n1 e n2 e armazenando nas variáveis num1 e num2, respectivamente. Na *linha 5*, criamos uma variável **soma** para armazenar o resultado obtido por *num1 + num2*. Na *linha 6*, estamos exibindo na tela o resultado da soma, além de usar o **getElementById()** também estamos utilizando a propriedade **innerHTML** para exibir o conteúdo no HTML, neste caso, o conteúdo é o texto "Resultado" que está sendo concatenado com o resultado da variável **soma**. Pronto! Finalizamos aqui o mesmo exemplo que fizemos na aula de Algoritmos, só que agora usando a linguagem de programação JavaScript. Vale ressaltar que, este mesmo exemplo pode ser feito de outras maneiras.

EXEMPLO 2

Agora aprenderemos a manipular três caixas de diálogos que são muito comuns em JavaScript: *alert*, *confirm* e *prompt*:

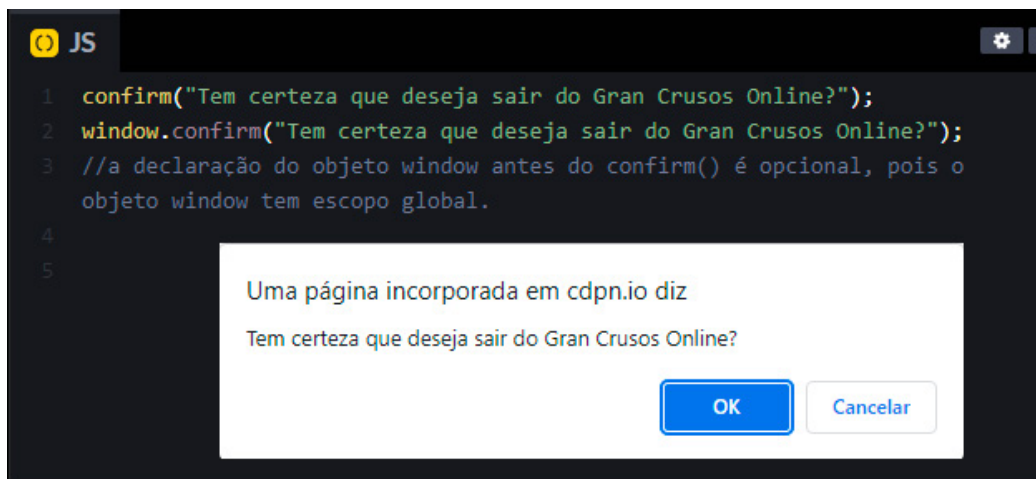
- **alert()**: "o método alert() exibe uma mensagem no navegador por meio de uma caixa de diálogo, que nada mais é que uma pequena janela *popup*" (NOLETO, 2022). O método alert() pode ser inserido de duas formas diferentes, como pode ser visto no exemplo abaixo.



A caixa de diálogo exibe, além do texto digitado, um botão de confirmação para indicar que a pessoa realmente leu a mensagem. Este recurso é muito utilizado para avisar sobre uma determinada falha, para dar boas-vindas em um novo cadastro efetuado em uma página e muitas mais (NOLETO, 2022).

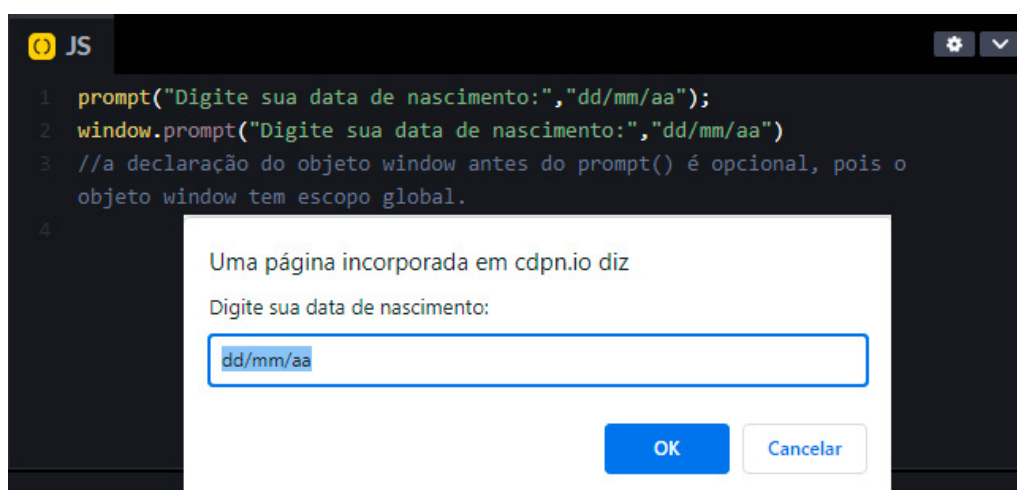
- **confirm():** o método confirm():

(...) permite que a pessoa usuária da página decida se deseja ou não executar uma ação determinada. Para isso, exibe uma janela modal com uma mensagem e dois botões: um de confirmação e outro que cancela a ação (NOLETO, 2022).



Vale ressaltar que, se a sua opção for o botão "OK" o método confirm() retornará VERDADEIRO, se for o botão "CANCELAR" o método confirm() retornará FALSO, por isso, se faz necessário utilizar uma variável para armazenar o retorno e decidir o que será a partir desta resposta. Outro ponto importante é que no método confirm() não é possível alterar o texto dos botões OK e CANCELAR.

- **prompt():** o método prompt() é muito utilizado para obter alguma informação do usuário, pois exibe uma caixa de diálogo com uma caixa de texto, onde o usuário pode entrar com a informação solicitada.



Neste caso, também se faz necessário utilizar uma variável para armazenar o valor digitado pelo usuário.



PRA PRATICAR

Aproveite para deixar como desafio o seguinte exercício: tente aprimorar o código do exemplo 1 utilizando o `prompt()` para pegar os números digitados pelo usuário e o `alert()` para exibir o resultado da soma.



LINK

Acesse o link: <https://blog.betrybe.com/javascript/javascript-alert/> para visualizar e testar aplicações reais utilizando os métodos `alert()`, `confirm()` e `prompt()`.



EXEMPLO 3

Para finalizar o nosso capítulo de mão na massa, vamos fazer um exemplo baseado no exercício da W3C disponível em: https://www.w3schools.com/js/js_intro.asp. Segue o link das imagens que vamos utilizar para reproduzir o exemplo:

Lâmpada acesa – https://www.w3schools.com/js/pic_bulbon.gif

Lâmpada apagada – https://www.w3schools.com/js/pic_bulboff.gif

Neste exemplo veremos como podemos utilizar o JavaScript para alterar valores de atributo HTML. Neste caso, o JavaScript vai alterar o valor do atributo **src** (source) de uma tag ``.


```

1  <html lang="pt-br">
2  <head>
3    <meta charset="UTF-8">
4    <title>Exemplo JS</title>
5  </head>
6  <body>
7    <h2>O que o JavaScript pode fazer?</h2>
8    <p>JavaScript pode alterar os valores de atributo HTML.</p>
9    <p>Nesse caso, o JavaScript altera o valor do atributo src
    (source) de uma imagem.</p>
10
11    <button onclick="document.getElementById('imagem').src='https://
    www.w3schools.com/js/pic_bulbon.gif'">Acende Lâmpada</button>
12
13    
14
15    <button onclick="document.getElementById('imagem').src='https://
    www.w3schools.com/js/pic_bulboff.gif'">Apaga Lâmpada</button>
16
17  </body>
18  </html>

```

Link para ter acesso ao código: <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/cc91e7a2747dfcb7c69e862ea4623eabfe0140a2/Unidade1Aula5b>

Também usamos o método **getElementById()** e do evento **onclick**, vistos no Exemplo 1. Ao clicar no botão “Acende Lâmpada”, o método **getElementById()** altera o atributo **src** da tag **** com o caminho da lâmpada acesa. Ao clicar no botão “Apaga Lâmpada”, o método **getElementById()** altera o atributo **src** da tag **** com o caminho da lâmpada apagada. Legal, né? Veja como fica:

O que o JavaScript pode fazer?

JavaScript pode alterar os valores de atributo HTML.

Nesse caso, o JavaScript altera o valor do atributo src (source) de uma imagem.



Acende Lâmpada

Apaga Lâmpada

O que o JavaScript pode fazer?

JavaScript pode alterar os valores de atributo HTML.

Nesse caso, o JavaScript altera o valor do atributo src (source) de uma imagem.



Acende Lâmpada

Apaga Lâmpada

Agora vamos fazer o mesmo exemplo de forma diferente, utilizaremos um documento de JS externo e os botões chamarão as funções presentes neste documento. Para isso, nosso HTML deve ficar assim:

```

1  <html lang="pt-br">
2  <head>
3      <meta charset="UTF-8">
4      <title>Exemplo JS</title>
5  </head>
6  <body>
7      <h2>O que o JavaScript pode fazer?</h2>
8      <p>JavaScript pode alterar os valores de atributo HTML.</p>
9      <p>Nesse caso, o JavaScript altera o valor do atributo src
      (source) de uma imagem.</p>
10
11     <button onclick="acendeLampada()">Acende Lâmpada</button>
12
13     
14
15     <button onclick="apagaLampada()">Apaga Lâmpada</button>
16
17     <script src="teste.js"></script>
18
19 </body>
20 </html>

```

Lembrando que, é necessário estabelecer o elo entre o HTML e o JavaScript, como pode ser visto na *linha 17*. Repare que neste caso, os eventos **onclick** presentes nos botões apontam para as funções (*linhas 11 e 15*).

Nosso documento externo de JavaScript deve ficar assim:

```

1  function acendeLampada()
2  {
3      document.getElementById('imagem').src='https://www.w3schools.com/
      js/pic_bulbon.gif'
4  }
5
6  function apagaLampada()
7  {
8      document.getElementById('imagem').src='https://www.w3schools.com/
      js/pic_bulboff.gif'
9  }

```

Ambos os exemplos fazem a mesma coisa, porém de forma diferente.

CONSIDERAÇÕES FINAIS

Nesta aula, tivemos a oportunidade de colocar a mão na massa e testar alguns dos conceitos vistos nas aulas anteriores. A ideia é que a partir de agora você consiga dar os seus próprios passos. Aproveite para explorar e editar os códigos dos exemplos acima, deixando tudo do seu jeitinho. Coloque em prática outros recursos que não foram contemplados aqui.

Resumindo, aprendemos neste capítulo como converter uma *string* em um inteiro, utilizando a função **parseInt()** e como obter um elemento do documento a partir de seu atributo ID especificado, utilizando o método **getElementById()**. Além disso, aprendemos também como manipular as caixas de diálogo *alert*, *confirm* e *prompt*.

MATERIAIS COMPLEMENTARES

Playlist JavaScript e HTML para iniciantes:

https://youtube.com/playlist?list=PLWd_VnthxxLdQyD4SiYIXFJK9nsxUFkHt

REFERÊNCIAS

NOLETO, Cairo. *Javascript alert, confirm e prompt: caixas de diálogo Popup!* 2022. Disponível em: < <https://blog.betrybe.com/javascript/javascript-alert/> >. Acesso em: 06 de nov. de 2022.