# Introdução a criação de websites com HTML e CSS

# HTML

## **Definições:**

HTML é uma linguagem de marcação. Ele é a base da web e está em sua 5° versão.

HTML é case insensitive, ou seja, não importa letra maiúscula ou minúscula.

Diagrama

Descrição gerada automaticamente

**Elemento** é a base do HTML, tudo dentro de um arquivo HTML é um elemento.

O elemento HTML é composto por uma tag de abertura, um atributo com seu conteúdo (opcional) e uma tag de fechamento (existem tag que não exigem tag de fechamento)

Uma **tag** em HTML é todo comando iniciado para indicar ao computador a estrutura do conteúdo a ser adicionado.

-A primeira linha do documento deve ser o **<!DOCTYPE html>**, apesar de parecer um elemento HTML ela apenas diz ao navegador que ele está lidando com um arquivo do tipo HTML5. Os elementos HTML vêm logo abaixo.

**<html>**

A tag **html** é a raiz do seu documento, todos os elementos HTML devem estar dentro dela. E nela nós informamos ao navegador qual é o idioma desse nosso documento, através do atributo **lang**, para o português brasileiro usamos **pt-BR**.

Obs.: “lang” faz referência a palavra “language”.

**<div>**

é um container genérico para conteúdo de fluxo, que de certa forma não representa nada. Ele pode ser utilizado para agrupar elementos para fins de estilos (usando class ou id), ou porque eles compartilham valores de atributos, como lang

**<head>**

A tag head contém elementos que serão lidos pelo navegador, como os metadados - um exemplo é o **charset**, que é a codificação de caracteres e a mais comum é a **UTF-8**, o JavaScript com a tag “**script**”, o CSS através das tags “**style**” (css dentro do html) e “**link**”(css fora do html), e o título da página com a tag “**title”.**

**Usando a tag <link>**

É obrigatório declarar o atributo rel=” “cujo pode conter vários valores, mas a pincipio usaremos “style sheet” para em seguida referenciar o arquivo com o CSS por meio do href=”arquivo”.

**<body>**

dentro da tag body colocamos todo o conteúdo visível ao usuário: textos, imagens, vídeos.

* background=”url” – Endereço da imagem de fundo
* bgcolor=”#” – Cor de fundo
* text=”#” – Cor do texto
* link=”#” – Cor do Link
* vlink=”#” Cor do link que foi visitado pelo usuário no mesmo navegador
* alink=”#” Cor do link ativo

**<section>**

Representa uma seção genérica de conteúdo quando não houver um elemento mais específico para isso.

**<header>**

É o cabeçalho da página ou de uma seção da página e normalmente contém logotipos, menus, campos de busca.

**<article>**

Representa um conteúdo independente e de maior relevância dentro de uma página, como um post de blog, uma notícia em uma barra lateral ou um bloco de comentários. Um article pode conter outros elementos, como header, cabeçalhos, parágrafos e imagens.

**<aside>**

É uma seção que engloba conteúdos relacionados ao conteúdo principal, como artigos relacionados, biografia do autor e publicidade. Normalmente são representadas como barras laterais.

**<footer>**

Esse elemento representa o rodapé do conteúdo ou de parte dele, pois ele é aceito dentro de vários elementos, como article e section e até do body. Exemplos de conteúdo de um <footer> são informações de autor e links relacionados.

**<h1>**-**<h6>**

Eles não foram criados na versão 5 do HTML e nem são específicos para semântica, mas servem para esse propósito. São utilizados para marcar a importância dos títulos, sendo <h1> o mais importante e <h6> o menos. Uma dica: use apenas um <h1> por página, pois ele representa o objetivo da sua página.

Texto

Descrição gerada automaticamente com confiança baixa

## Textos e links:

**<p>**

representa um parágrafo, mas ele não suporta apenas texto, podemos adicionar imagens, código, vídeos e vários outros tipos de conteúdo dentro dele.

**<Strong>**

serve para deixar um texto em negrito.

**<a>**

Um outro elemento interessante e extremamente necessário na web é o **<a>** - que significa anchor/âncora, ele representa um hyperlink, é ele que interliga vários conteúdos e páginas na web.

O elemento **<a>** tem vários atributos, mas vamos focar em dois, o href e o target.

O **href** representa o *hyperlink* para onde sua âncora aponta, pode ser uma página do seu ou de outro site, um e-mail e até mesmo um telefone, os dois últimos precisam dos prefixos [**mailto:”seu**](mailto:)**-email”** **e** [**tel:”seu**](tel:)**-telefone”,** respectivamente.

O **target** neste momento vai servir para nos ajudar a abrir nossos links em outra aba do navegador usando o valor **\_blank**.

* href=”endereço” Localização do endereço que será direcionado.
* name=”??” Nome do autor
* target=”?” para onde o link será aberto: \_self, \_blank, \_top, \_parent.
* href=”url#bookmark” Adiciona o site aos favoritos.
* href=”mailto:email” É utilizado para criar um link para o e-mail. Dependerá do cliente de e-mail instalado no dispositivo do usuário.

Imagens:

A web também é feita de imagens e para representá-las temos o elemento **<img>**, ele é um daqueles elementos **sem tag de fechamento**.

O elemento img é bem simples, contendo apenas 2 atributos próprios, o src e o alt.

O **src** é obrigatório e guarda o caminho para a imagem que você quer mostrar na página.

O **alt** não é obrigatório, mas é altamente recomendado por melhorar a acessibilidade, ele mostra a descrição da imagem caso ela não carregue e leitores de tela usam esse atributo para descrever a imagem para o usuário saber o que ela significa.

## Listas:

Os últimos elementos que veremos neste módulo são os relacionados a lista: **<ul>, <ol> e <li>.**

Listas servem para agrupar uma coleção de itens, como uma lista de ingredientes ou, como será no nosso caso, uma lista com contatos.

O elemento **ul** cria uma lista não ordenada, onde a ordem dos elementos não é importante, e é representada com pontos, círculos ou quadrados.

O **<ol>** serve para criar lista ordenadas, nessas a ordem importa, portanto elas são representadas com números, algarismos romanos ou letras.

E o elemento **li** é um item dentro de uma dessas listas. Um <li> pode conter vários tipos de conteúdo, como parágrafos, imagens e até outras listas.

* <ol> … </ol> Lista ordenada
* <ul> … </ul> Lista sem ordem
* <li> … </li> Itens de uma lista. Podem ser utilizados tanto com as tags <ol> quanto <ul>
* <ol type=”?”> Define o tipo de ordenação: A, a, I, i, 1
* <ol start=”??”> Define o valor de início da lista.
* <ul type=”?”> Define o tipo de lista não-ordenada: disc, circle, square
* <li value=”??”> Define o valor do item
* <li type=”??”> Define o tipo do item
* <dl> … </dl> Lista de conceitos
* <dt> … </dt> Termo ou frase
* <dd> … </dd> Definição detalhada sobre o item
* <br> Quebra de linha

**Caracteres especiais:**

* &lt; < – Menor que;
* &gt; > – Maior que;
* &amp; & – E comercial;
* &quot; ” – Sinal de citação;
* &copy; © – Copyright;
* &trade; ™ – Trademark;
* &nbsp; – Espaço;

## Comandos úteis no VScode:

Ao criar um arquivo, podemos escrever na tela **“!” ou “html:5”** e a estrutura html básica será escrita automaticamente.

Em listas, se eu quiser criar mais de uma eu escrevo **ul>li\*” nº”**

Extensão omni theme para melhorar o visual do editor

Ctl + ou – para controlar o zoom

Extensão live server para atualizar automaticamente o site que você está criando

# CSS

CSS é uma linguagem de estilo cuja finalidade é a de formatar páginas.

Uma regra de css é constituída por um grupo de seletores

Texto

Descrição gerada automaticamente com confiança baixa

SELETOR {PROPRIEDADE: VALOR}

O**atributo Id** especifica uma identificação única para o elemento HTML. Por questões de boas práticas, não deve ser reutilizado e nem conter espaços em seu nome, pois o navegador irá identificar o espaço como parte dele, já que os elementos não podem ter mais de um Id. É geralmente utilizado para referenciar os elementos em scripts Javascript, através da **função nativa** document.getElementById(). Caso usado no CSS e referenciado através do símbolo de **cerquilha (#).**

O**atributo global class** especifica uma ou mais classes para o elemento HTML. Esse atributo pode ser reutilizado, ajudando a pessoa desenvolvedora a não repetir códigos, além de permitir o uso de diferentes classes simultaneamente. Para melhor organização do projeto, também possuímos padrões para nomear essas classes como o [Block Element Modifier](https://www.alura.com.br/artigos/nomes-de-classes-no-css). É possível acessar elementos específicos pela sua classe no Javascript através de funções como o document.getElementsByClassName() ou através do **seletor de classe CSS**, representado pelo símbolo **ponto (.)**.

**há o caso em que apenas um elemento possui duas classes**. Dentro do atributo class, é interpretado que são classes distintas a cada “espaço” entre as palavras. Dessa maneira, podemos colocar duas ou mais classes em um mesmo elemento. (usado quando dois elementos que dividem o mesmo estilo, mas possuem algumas diferenças entre si! Assim, você não precisa repetir essas estilizações que eles possuem iguais.)

o **id e class podem ser usados no mesmo elemento** ao mesmo tempo. Mas para que isso? Imagine que você quer aplicar uma função, mas o elemento compartilha a mesma classe com outros? Para conseguir individualizar ele, podemos recorrer ao id.

Podemos exemplificar com um botão de envio de formulário, que irá precisar enviar um alerta para confirmar o envio de informações, mas ao mesmo tempo compartilha estilizações com outros botões

Texto

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

Box model é a representação do elemento HTML (as tags) em um bloco retangular, que está dividido em 4 partes: conteúdo, espaçamento interno (propriedade padding), a borda (propriedade border) e a margem (propriedade margin)

Propriedades:

Background = cor de fundo

Font-family = tipo de fonte da letra

Max-width = largura máxima

Min-width = largura minima

Margin = tamanho da margem

Border = tamanho tipo e cor da borda

Border-radius = arredondando as bordas

Color = cor

Font-size = tamanho da fonte

Text-transform = maiúscula, minúscula, somente a primeira maiúscula...

Text-decoration = riscar links com line-through, colocar negrito com bold

Font-style = italico

Margin-bottom

Padding = margem do padding

Text-allign = alinhamento do texto

List-style-type = modificando listas

Padding-left

List-style: = define o tipo de marcador da lista

Line-height = espaçamento entre linhas

Font-weight = peso ou a intensidade da fonte (ex.: negrito)

Propriedades flexbox:

Display = ativa a propriedade flex

Flex-direction = define se a ordenação será em linha ou coluna se será padrão ou inversa

Float = determina que um elemento deve ser retirado do seu fluxo normal e colocado ao longo do lado direito ou esquerdo do seu containêr, onde textos e elementos em linha irão se posicionar ao seu redor

Flex-wrap = determina o tipo de quebra de linha

Flex-flow = flex-direction+flex-wrap

Jstify-content = indica de que forma será feito o alinhamento dos itens dentro do container

Flex = flex-grow+flex-shrink+flex-basis

Align-items = funciona igual ao justify-content, porém, no eixo transversal.

Align-content = Organiza as linhas dentro de um flex container quando há espaço extra no eixo transversal

Flex-grow = habilidade de um flex item de crescer, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção.

Flex-basis = Define o tamanho padrão para um elemento antes que o espaço remanescente do container seja distribuído.

Flex-shrink = Define a habilidade de um flex item de encolher, caso necessário.

Order = Por padrão os flex items são dispostos na tela na ordem do código. Mas a propriedade order controla a ordem em que aparecerão no container.

Align-self = faz o alinhamento item a item sobrescrevendo o aign-items.

PSEUDOELEMENTOS:

Um pseudo-elemento CSS é uma palavra-chave adicionada a um seletor que permite que você estilize uma parte específica do elemento selecionado.

Texto

Descrição gerada automaticamente

Você pode utilizar apenas um pseudo-elemento em um seletor. Ele deve aparecer depois da declaração de um elemento simples.

Observação: Como regra, os dois pontos devem ser usados duas vezes  (::)  ao invés de uma única vez  (:). Isso distingue pseudo-classes de pseudo-elementos.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa