# HTML5 & CSS3

## HTML: ESTRUTURA DA PÁGINA

Uma página HTML é composta de **elementos** e esses elementos, por sua vez, são definidos através das **tags**. Temos a tag de abertura e a de fechamento: **<element> </element>**. Entre as tags de abertura e fechamento encontra-se determinado conteúdo: **<element>** conteúdo **</element>**. As tags podem ser categorizadas inicialmente em dois tipos: em “nível de bloco” (***block-level***) e “em linha” (***inline***). Um elemento em **nível de bloco** ocupa toda a largura de seu elemento pai, que também chamamos de elemento container, criando assim um “bloco”. Já os **elementos *inline***, geralmente usamos para demarcação de conteúdos de texto.

**<!DOCTYPE html>**

Informa ao navegador a estrutura básica do HTML e qual a sua versão que estamos utilizando. Quando está escrito apenas html, indica que é a mais recente.

**<html>**

Representa a raiz do documento, serve como um container que engloba todos os outros elementos HTML, marcando o conteúdo a ser renderizado no navegador. A tag **lang=”pt-br”** acompanha a tag <html> e define a linguagem da página, no caso, português-brasil.

**<title>**

Define o título de uma página.

**<body>**

É onde fica todo o conteúdo de texto, imagem e vídeos, em que o usuário vê e interage. Nele os conteúdos são estruturados pelas demais tags do HTML.

**<script>**

Esse elemento contém instruções de script ou aponta para um arquivo de script externo por meio do atributo src.

**<head>**

É o lugar onde passamos as informações do documento (metadados) ao navegador, sendo interpretadas por ele. Como por exemplo, título do documento, links para folhas de estilo etc.

**<meta>**

Define metadados, ou seja, informações sobre dados de um documento HTML. As <meta> tags vão dentro do elemento <head> e são usadas para especificar o conjunto de caracteres, o autor do documento, as configurações da janela de visualização etc. A tag **<meta charset=”UTF-8”>** passa as informações do ***encoding*** da nossa página para o navegador.

**<link>**

É uma tag vazia, que contém apenas atributos e faz a relação do documento HTML com recursos externos, é comumente usado para vincular uma folha de estilo externa, também é usada para definir o favicon da página (ícone da aba do navegador), como outros recursos.

**<style>**

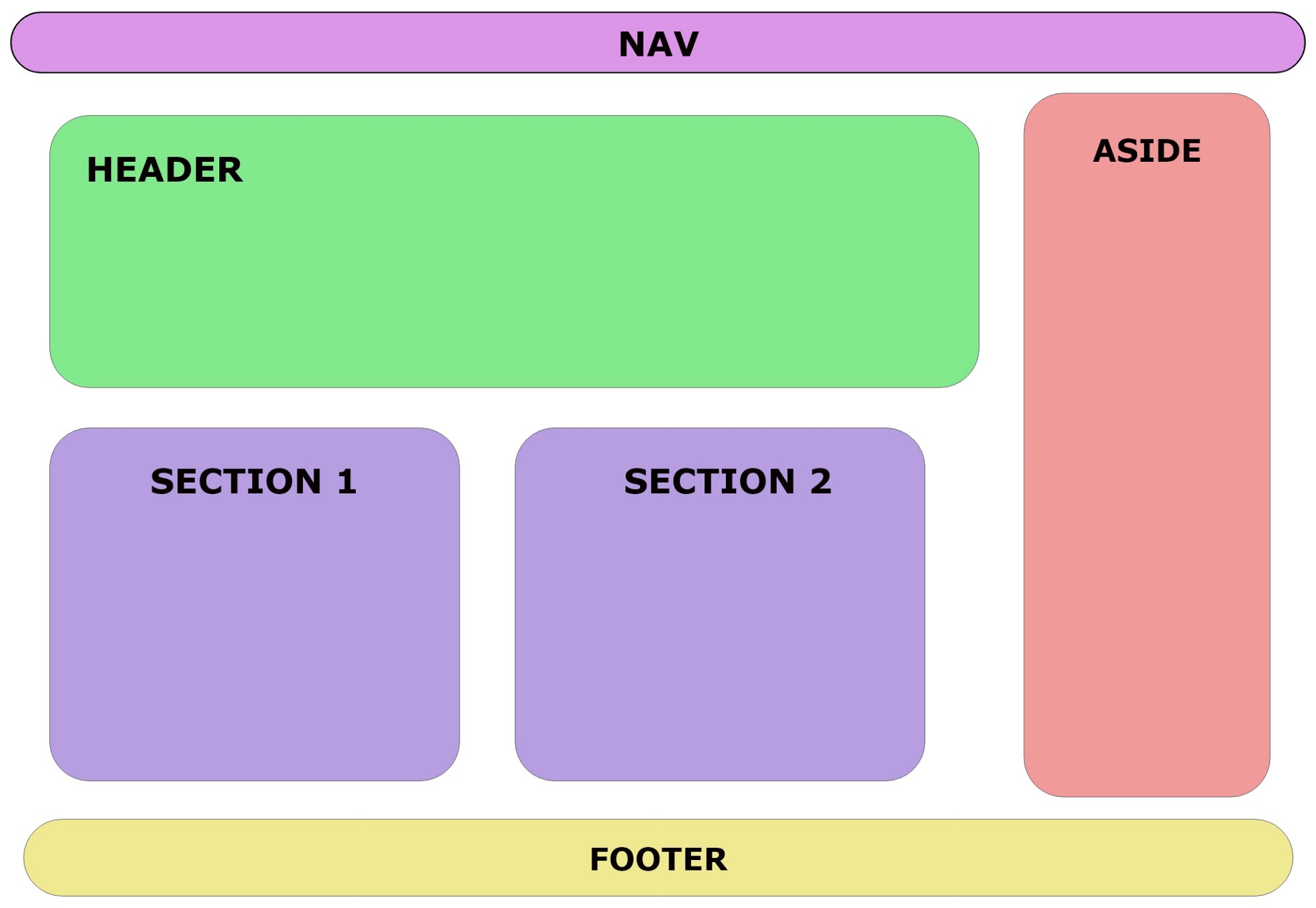
Essa tag é usada para declarar estilos (CSS) para um documento.

**Tags semânticas**

Tags semânticas são tags que possuem um significado, que dão sentido a informação de texto ao navegador e buscadores, como por exemplo, utilizar a tag <header> para cabeçalhos ou <article> para dar um significado de artigo para aquele bloco de texto, até mesmo <p> para indicar que aquele texto é um parágrafo, é uma boa prática tentar sempre utilizar essas tags semânticas para ajudar no entendimento do código, além de ajudar muito no SEO do site (Otimização para motores de busca, é o que ajuda o seu site a se rankear melhor nos motores de buscas como o Google). Segue abaixo a lista de tags semânticas citadas durante o artigo:

* **<header>**
* **<main>**
* **<footer>**
* **<section>**
* **<article>**
* **<aside>**
* **<nav>**
* **<ol>**
* **<ul>**
* **<li>**

Exemplo de estruturação com tags semânticas:



**Tags sem semântica**

As tags que não possuem semântica não definem um significado para aquele texto, normalmente são utilizadas apenas para fins de separação e estilização. Veja logo abaixo a lista de algumas tags sem semântica:

* **<div>:** serve para agrupar múltiplos elementos a **nível de blocos**;
* **<span>**: serve para agrupar elementos e representar elementos **em linha** visuais que não tenham uma semântica objetiva. Um exemplo de representação de elementos visuais é criarmos um botão utilizando CSS com menu hambúrguer, muito comum em sites responsivos. Ex.:

**HTML**

<button class="menu-hamburger">

  <span></span>

  <span></span>

  <span></span>

</button>

**CSS**

.menu-hamburger {

  border-radius: 10px;

  background: transparent;

  border: 1px solid #66ffa6;

  height: 50px;

  width: 50px;

  padding: 10px;

}

.menu-hamburger span {

  display: block;

  position: relative;

  border-radius: 5px;

  background-color: #66ffa6;

  height: 4px;

  width: 100%;

  margin-bottom: 7px;

}

.menu-hamburger span:last-child {

  margin-bottom: 0px;

}

O **resultado** de nosso exemplo será um botão com três linhas representando um menu como mostra a seguinte imagem:



* **<b>**
* **<i>**

## HTML: CABEÇALHO <header></header>

Como o cabeçalho é uma coisa muito comum em todas as páginas web, na última versão do HTML criaram uma tag específica para o cabeçalho. É uma tag mais semântica que diz para o navegador que aquilo ali é a primeira informação a ser apresentada. O cabeçalho da página deve ter mais destaque, devendo constar o nome do local e, se cabível, informações essenciais curtas. A **<header>** fica dentro do **<body>**.

**Obs.:** podemos usar uma classe no CSS para estilizar a tag **<h1>** que geralmente fica na **<header>**;

**Obs.:** não é recomendado criar estilos usando tags, o ideal é usarmos classes para tudo.

## HTML: TEXTO

**<h1** até **h6>** e **<p>**

Define o título e os parágrafos de um texto, respectivamente.

**Obs.:** uma página deve ter títulos e subtítulos bem definidos, os mecanismos de buscas consideram esse fator importante. Não obstante, uma página deve ter pelo menos uma tag h1 (princípio de SEO).

**<article>**

Inclui um artigo da página, muito utilizado em blogs e páginas de criação de conteúdo, também indica o principal conteúdo de texto da página.

**<aside>**

Representa uma seção que faz referência a outro conteúdo da página, como uma definição, uma explicação extra, avisos, biografia do autor, ou seja um conteúdo complementar.

**<nav>**

Contempla o menu de navegação das páginas do site, e dentro inserimos a listas e links com a tag <a href=””></a>.

**<div>**

Assim como as outras tags, também funciona como um container, porém a grande diferença é que a div não tem valor semântico, é apenas uma divisão na página para fins de layout.

**<strong>**

Dá destaque para algumas informações do texto, deixando-as em negrito.

**<em>**

Dá ênfase para algumas informações do texto, deixando-as em itálico.

**<sup>**

Deixa o texto marcado sobrescrito**.**

**<sub>**

Deixa o texto subscrito.

**<blockquote>**

Bloco de citação**.**

**<q>**

Citações curtas que não requerem marcações de parágrafo**.**

**<abbr>**

Abreviação.

**Obs.:** opcionalmente fornece uma descrição completa para ela. Se presente, o atributo **<title>** deve conter a descrição completa e apenas ela.

**<dfn>**

Instância de definição de um termo**.**

**<cite>**

Referência a um trabalho artístico**.**

**Obs.:** deve incluir o título do trabalho ou uma URL de referência, que pode ser em uma forma abreviada de acordo com as convenções usadas para a adição dos metadados de citação.

**<ins>**

Intervalo de texto que foi adicionado a um documento**.**

**<del>**

Parte do texto que foi excluída de um documento.

**Obs.:** este elemento é (não necessariamente) renderizado pelos navegadores com uma linha entre o texto.

**<address>**

Fornece informações de contato para seu ancestral **<article>** ou **<body>** mais próximo; no segundo caso, ele se aplica ao documento inteiro.

**Obs.:** a informação de contato fornecida por um conteúdo do elemento **<address>** pode tomar qualquer forma que é apropriada para o contexto, e pode incluir qualquer tipo de informação de contato necessária, como endereço físico, URL, email, telefone, mídia social, coordenadas geográficas, por aí vai. Deve incluir o nome da pessoa, grupo, ou organização para qual as informações de contato se referem.

**<s>**

Renderiza um texto tachado ou uma linha cortando o texto.

**Obs.:** use o elemento **<s>** para representar texto que não sejam relevantes ou que não estejam corretos. Não é apropriado o uso do **<s>** indicar edições no texto; para indicar edições no texto utilize **<del>** e **<ins>**, que são elementos mais apropriados.

**<br />**

Quebra uma linha.

**<hr />**

Desenha uma linha, para separação de tópicos.

**<hr>**

Essa tag constrói uma linha horizontal entre elementos, representa semanticamente uma quebra de conteúdo.

**<audio>**

Utilizada para inserir áudios no site, tendo como principais atributos: **src**, recebendo como valor o link ou diretório do audio, **controls** caso queira que seja possível controlar o áudio, **autoplay** para definir que o áudio de tocar automaticamente quando entrar no site, **type** recebendo como valor o tipo do áudio.

**<video>**

Utilizada para inserir vídeos no site, a tag possui atributos como **width** recebendo como valor a largura do vídeo em pixels, **height** recebendo como valor a altura do video em píxeis, caso não for informada esses atributos, será utilizado a largura e altura padrão do vídeo, **controls** (quando presente nos permite controlar o video), **src** recebendo como valor o link ou diretório do arquivo de vídeo.

## HTML: LISTAS E DIVISÕES DE CONTEÚDOS

**<ul>** cria uma lista **não**-ordenada **</ul>**

O exemplo abaixo:

**<ul>**

**<li>**1kg King Edward potatoes**</li>**

**<li>**100ml milk**</li>**

**<li>**50g salted butter**</li>**

**<li>**Freshly grated nutmeg**</li>**

**<li>**Salt and pepper to taste**</li>**

**</ul>**

gerará o seguinte resultado:

* 1kg King Edward potatoes
* 100ml milk
* 50g salted butter
* Freshly grated nutmeg
* Salt and pepper to taste

**<ol>** cria uma lista ordenada **</ol>**

O exemplo abaixo:

**<ol>**

**<li>**Chop potatoes into quarters**</li>**

**<li>**Simmer in salted water for 15-20 minutes until tender**</li>**

**<li>**Heat milk, butter and nutmeg**</li>**

**<li>**Drain potatoes and mash**</li>**

**<li>**Mix in the milk mixture**</li>**

**</ol>**

gerará o seguinte resultado:

1. Chop potatoes into quarters
2. Simmer in salted water for 15-20 minutes until tender
3. Heat milk, butter and nutmeg
4. Drain potatoes and mash
5. Mix in the milk mixture

**<ol reversed>**

Inverte a ordem da numeração de uma lista ordenada. Pegando o exemplo acima, a lista começará com o número 5 e decrescerá.

**<ol start="**10**">**

Atribui um número para o item inicial e resultará em:

1. Chop potatoes into quarters
2. Simmer in salted water for 15-20 minutes until tender
3. Heat milk, butter and nutmeg
4. Drain potatoes and mash
5. Mix in the milk mixture

**<li value="**3**">**

Indica o valor ordinal atual do item na lista, impondo um valor de numeração a um item de uma lista ordenada e resultará em:

1. total de votos 123
2. total de votos 99
3. total de votos 63
4. total de votos 63
5. total de votos 41

**Obs.:** o único valor possível para este atributo é um número, ainda que a lista seja exibida com algarismos romanos, ou letras. A lista de itens que virá em seguida continuará a ser numerada a partir desta posição.

**<dt>** cria lista de definição e identifica o termo nela **</dt>**

O exemplo abaixo:

**<dl>**

**<dt>**Sashimi**</dt>**

**<dd>**Sliced raw fish that is served with condiments such as shredded daikon radish or ginger root, wasabi and soy sauce**</dd>**

**<dt>**Scale**</dt>**

**<dd>**A device used to accurately measure the weight of ingredients**</dd>**

**<dd>**A technique by which the scales are removed from the skin of a fish**</dd>**

**<dt>**Scamorze**</dt>**

**<dt>**Scamorzo**</dt>**

**<dd>**An Italian cheese usually made from whole cow's milk (although it was traditionally made from buffalo milk)**</dd>**

**</dl>**

gerará o seguinte resultado:

Sashimi

Sliced raw fish that is served with condiments such as shredded daikon radish or ginger root, wasabi and soy sauce

Scale

A device used to accurately measure the weight of ingredients

A technique by which the scales are removed from the skin of a fish

Scamorze

Scamorzo

An Italian cheese usually made from whole cow's milk (although it was traditionally made from buffalo milk)

**<li>**

Identifica cada um dos itens da lista.

**<div>**

Divide o conteúdo genérico da página e não interferem no padrão visual, apenas organiza o código.

**Obs.:** a melhor técnica para alinhar uma div ao centro é usar o cálculo da ***margin*** automático. Delegamos para o navegador fazer a conta de calcular a largura que resta, dividir por dois, e adicionar metade em cada um dos lados.

**<section>**

Divide o conteúdo **semântico** da página **</section>**

**Obs.:** só devemos utilizar a tag <section> quando tivermos um **bloco onde o conteúdo tenha o mesmo significado, o mesmo sentido**.

**<main>**

Divide o conteúdo mais importante, principal, da página**.**

**Obs.:** as tags semânticas fazem com que o navegador entenda melhor o nosso conteúdo, e com isso carregue melhor a nossa página, deixando o código mais legível.

**<fieldset>**

Divide o conteúdo de um formulário.

**<nav>**

Representa uma seção de uma página que aponta para outras páginas ou para outras áreas da página, ou seja, uma seção com links de navegação.

## HTML: IMAGENS

**<img src=”**nome do arquivo da imagem e sua extensão ***imagem.jpg*” >**

**<img id/class=”**caso haja, algo que lembre a imagem**” src...>**

**Obs.:** será ***id*** se só houver um elemento de imagem no projeto, caso contrário, será ***class***. As imagens do projeto devem estar na **mesma pasta**.

**<img src=”**...**” alt=”**mensagem alternativa que será exibida caso o endereço não localize o alvo (o arquivo)**”>**

**<a href="**link**">**

**<img src="**/path/to/imgs/imagem.jpg**" />** (imagem como link)

**</a>**

## HTML: LINKS (âncoras)

**<a href=”http://**...**”>**texto exibido na página**</a>** cria links para outras páginas, sejam elas do nosso projeto ou páginas externas;

**<a href="**link**" target="\_blank" >**texto exibido na página**</a>** abre o link em outra aba;

**<a href="**link**" rel="nofollow" >**texto exibido na página**</a>** diz ao SEO para desconsiderar esse link;

Cardápio (menu) de links:

**<ul>**

**<li>** **<a href="**link 1**">**Home**</a>** **</li>**

**<li>** **<a href="**link 2**">**About us**</a>** **</li>**

**<li>** **<a href="**link 3...**">**Contact**</a>** **</li>**

**</ul>**

**<a href="mailto:**fulano@example.com**?subject=**assunto do e-mail**">**link**</a>** envia um e-mail com um assunto específico;

Pseudoclasses para links:

**:link** = quando o link ainda não foi visitado;

**:visited** = quando o link já foi visitado;

**:hover** = quando o ponteiro do mouse passa sobre sua área;

**:active** = quanto o link é ativado (ao clicar do mouse).

**Obs.:** o mais comum é desativar o sublinhado e reativá-los quando o mouse é posicionado sobre eles.

**a**:link, **a**:visited {text-decoration: none}

**a**:hover, **a**:active {text-decoration: underline}

## HTML: TABELAS

Uma tabela é composta por linhas e células. As tabelas começam com a tag **<table>**, as linhas da tabela são definidas pela tag **<tr>** (table row). Dentro das linhas temos as células, elas são definidas pela tag **<td>** (table data). Vamos ao nosso primeiro exemplo de tabela:

**<table>**

**<tr>**

**<td>**campo1**</td><td>**campo2**</td><td>**campo3**</td>**

**</tr>**

**<tr>**

**<td>**campo1**</td><td>**campo2**</td><td>**campo3**</td>**

**</tr>**

**<tr>**

**<td>**campo1**</td><td>**campo2**</td><td>**campo3**</td>**

**</tr>**

**</table>**

Resultado:

|  |  |  |
| --- | --- | --- |
| campo1 | campo2 | campo3 |
| campo1 | campo2 | campo3 |
| campo1 | campo2 | campo3 |

## HTML: FORMULÁRIOS

O formulário pega os dados digitados pelo usuário, associa cada dado a um nome de campo e os envia ao servidor. Lá no outro lado (no servidor), uma linguagem de servidor recebe os dados e faz alguma coisa com eles, seguindo a lógica do script criado pelo programador.

Um formulário web é a **porta de entrada dos dados de sua aplicação**. É nele que seu usuário faz a entrada de dados. Abaixo, temos um exemplo de formulário:

A tag **<form>** é o início do formulário, automaticamente, sabemos que a tag **</form>** é o seu fechamento. Dentro das tags <form> colocamos os **controles** (***inputs***), ***labels*** e alguns botões.

Um formulário na web normalmente é chato de se preencher, só que ele é a alma dos aplicativos web, pois é através de seus campos que o usuário faz a inserção dos dados e, dessa forma, interage com o sistema.

Um formulário pode (e deve) conter elementos que formam um par **nome=valor**. Ex.: um campo de entrada de texto (***text box***) chama-se *pais* e o seu **valor** é o texto *Brasil*. Quando esse formulário submeter seus dados para o servidor, ele poderá trabalhar com a variável pais e seu valor será Brasil.

A tag ***input*** compõem a maioria dos **controles básicos**.

**<input** type="" name=""**/>**

A propriedade ***name*** será associada ao **valor do campo**. Alterando-se a propriedade type podemos obter os seguintes **controles** **básicos**:

* ***Textbox*** (input type=”***text***”)

**<input** type="text" name="user-name"**/>**

* ***Textarea*** (sabe aquelas textbox grandonas? São as textbox de várias linhas ou ***textbox* *multiline***)

<textarea> ... </textarea>

**Obs.:** as propriedades ***rows*** e ***cols*** referem-se ao número de **linhas** e **colunas** respectivamente.

* **Campos de *senha*** (input type=”***password***”)

**<input** type="password" name="old-pass" **/>**

**Obs.:** os campos de senha são semelhantes às ***textboxs***, a diferença é que os **dados aparecem escondidos** pelo sinal de asterisco (\*).

* **Caixa de *checagem*** (input type=”***checkbox***”)

**<input** type="checkbox" name="company" checked**/>**

**Obs.:** as ***checkboxes*** são como uma escolha booleana *true* ou *false*, sim ou não. Normalmente vem em grupo, mas também podem vir sozinhas. Para dizer qual a ***checkbox*** foi ticada (**selecionada**), usamos a propriedade ***checked***.

* **Botões de *opção*** (input type=”***radio***”)

**<input** type="radio" name="band-rock" value="beatles"**/>**

**<input** type="radio" name="band-rock" value="led-zeppelin"**/>**

**<input** type="radio" name="band-rock" value="pink-floyd"**/>**

Repare que a propriedade ***name*** é igual para o mesmo grupo de opções e que a propriedade ***value*** deve ser definida pelo programador. Para exibir o controle ***radio*** selecionado, usamos a propriedade ***checked***. Os ***radio buttons*** são como uma opção entre, no mínimo, duas.

Para exibir o controle ***radio*** selecionado, usamos a propriedade ***checked***.

**<input** type="radio" name="company" checked="checked"**/>**

O botão ***radio*** sofre do mesmo problema que o botão ***checkbox***, quando não clicado ele não envia nenhuma informação ao servidor, absolutamente nada. Mas o problema pode ser amenizado, uma vez que você pode definir um valor na hora de carregar o formulário. Quero dizer que ele virá pré-selecionado, ou seja, não haverá controle “não ticado”, pois pelo menos uma opção já foi selecionada no carregamento do formulário.

* Botões (***button***, ***submit*** e ***reset***)

**<button** name="button"**>**Enviar formulário**<button/>**

**<input** type="reset" value="Apagar formulário" **/>**

* **Enviando *arquivos*** (input type=”***file***”)

**<input** type="file" **/>**

O “***input file***” é um controle feito para realizar **uploads de arquivos**. Ele vem acompanhado de uma ***text box***. Ao clicar no botão “***Browse***”, o navegador abre a janela “*arquivos*” de seu sistema operacional e você poderá selecionar os arquivos de seu hard-disk.

Há também os **controles não básicos**:

* Caixa de múltipla seleção (***list box***)
* Caixa de seleção (***combo box***)

<select id=cbPais>

<option value="">Select Country</option>

<option value="usa">Usa</option>

<option value="ca">Canada</option>

<option value="me">Mexico</option>

</select>

A caixa de seleção é um controle parecido com um menu de opções. As opções (que vão dentro da tag ***select*** utilizam a tag ***option***.

Se, por exemplo, o usuário clicar na opção “Canada” será enviado para o servidor o **par/valor** *cbPais=ca*. Quando não definimos a propriedade ***value***, o valor passa a ser o texto contido entre as tags ***options***, ou seja, o **par/valor** seria *cbPais=Canada*.

* Text box multiline (***text area***)

Cada controle deve ser acompanhado do elemento ***label***. Esse elemento pode ajudar a adicionar estrutura e aumentar a acessibilidade aos formulários web. Em muitos navegadores, clicar no elemento ***label*** fará com que o elemento do formulário ganhe o foco. Ex.:

<p>

<label for="author">Name: </label> <input name="author" id="author" type="text" />

</p>

Posicionar os ***labels*** de modo que eles apareçam verticalmente acima dos elementos do formulário é realmente muito simples. ***Labels*** são elementos ***inline*** por padrão. Entretanto, configurar sua propriedade ***display*** como ***block*** fará com que eles gerem sua própria caixa de bloco, forçando os elementos “de entrada” (***inputs***) para a linha de baixo. A largura das caixas de entrada de texto varia entre diferentes navegadores, portanto, para consistência, você deve configurar explicitamente a largura das suas caixas de texto. Nesse exemplo, ***pixels*** são usados, mas, naturalmente, você poderia utilizar ***ems*** para criar um layout de formulário mais autoajustável.

## HTML: ESTRUTURA BÁSICA COM DEFINIÇÕES E EXEMPLOS

**<!DOCTYPE html>** (define a estrutura básica do HTML e qual versão do HTML estamos utilizando)

**<html lang=”pt-br”>** (marca o conteúdo a ser renderizado no navegador e define a linguagem da página)

**<head>** (passa as informações pro navegador)

**<meta charset=”UTF-8”>** (*enconding*: palavras como “ç”, acentos e certos símbolos serão lidos)

**<title>**TÍTULO DA PÁGINA**</title>**

**<link rel=”stylesheet” href=”**arquivo .css desejado” **>** (chama o arquivo CSS e marca o html com ele)

**</head>**

**<body>** (separa todo o conteúdo da página)

**<header>** (cabeçalho da página)

**<h1>**TÍTULO DO CABEÇALHO OU IMAGEM**<img src=”**logo.png**”></h1>**

**<div>** (divide um conteúdo genérico, sem importância)

**<nav>** (separa uma seção com links de navegação)

**<u>** (lista *não ordenada* para colocar as opções das abas de navegação da página)

**<li><a href=”**link que será carregado**”>**Home**</li>** (marca os itens da lista)

**<li><a href=”**link**”>**Sobre nós**</li>**

**<li><a href=”**link**”>**Contato**</li>**

**<li><a href=”**link**”>**...**</li>**

**</ul>**

**<nav>**

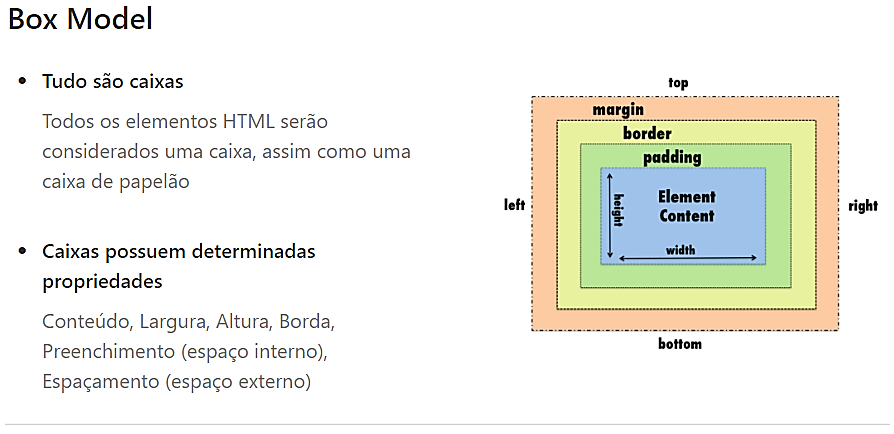
**<div>**

**</header>**

**</body>**

**</html>**

## BOX MODEL

****

## SINTAXE DO CSS

As **tags** têm um valor, um propósito, um significado, e para mexer na apresentação visual dos elementos, não usamos elas. Usamos o CSS.

O **CSS**, que em uma tradução seria ***folha de estilo em cascata***, é a forma como mexemos em cada um dos elementos granularmente, em cada um dos pontos desses elementos para podermos apresentar o nosso site do modo como quisermos. A primeira coisa a se fazer é procurar exatamente a tag do parágrafo que queremos modificar. Por que na tag do parágrafo? Porque quero que seja aplicado em todo o conteúdo que está ali dentro. Por isso a “*folha de estilo em cascata*”. Quando vamos em um elemento anterior, ele reflete para todos os elementos que estão abaixo. Como é uma folha de estilos em cascata, a cascata vai descendo, e **o que eu faço no elemento pai reflete para o elemento filho**.

**Propriedade**: identificador que possui um nome legível, que define o que será considerado ou editado;

**Valor**: descreve como o recurso será tratado pelo motor. Cada propriedade possui um conjunto de valores válidos, definida por uma gramática formal, bem como um significado semântico, implementado para o motor do navegador.

A propriedade e valor são chamados de **declaração**, e qualquer motor do CSS calcula quais declarações serão aplicadas para todos um único elemento da página em ordem adequadamente, a fim de exibi-lo com o estilo correto.

Tanto as propriedades como os valores **são *case-sensitive*** no CSS. Os pares se separam por dois pontos e espaços em branco antes, entre e depois de propriedades e valores, porém os espaços dentro da declaração são ignorados. Ou seja, o caractere “**:**” serve para atribuir um valor a uma propriedade.

Declarações são agrupadas em **blocos**, que estão delimitados na estrutura com uma chave de abertura '**{**' e fechadas com outra '**}**'. Os blocos às vezes podem estar aninhados, a abertura e fechamento de chaves no bloco CSS deve ser realizada. Esses blocos são chamados de **blocos de declaração** e as declarações dentro deles são separadas por '**;**'. Um bloco de declaração pode não conter nenhuma declaração. Espaços em branco em volta das declarações não são consideradas. Não é necessário que a última declaração possua ponto e vírgula, apesar de ser considerada uma boa prática pois previne o esquecimento de acrescê-la quando for necessário aumentar o bloco.

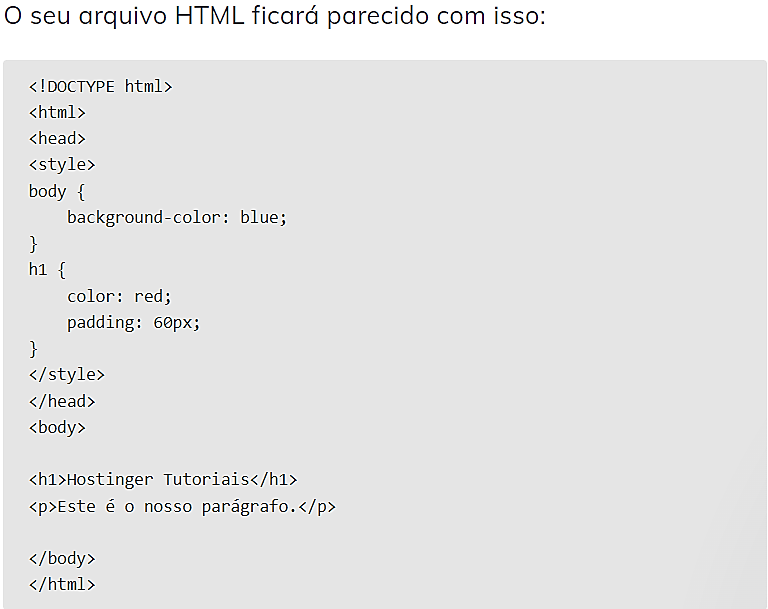
**TIPOS DE CSS**

**CSS interno:** ou incorporado, requer que você adicione a tags **<style>** na seção **<head>** do seu documento HTML. Este estilo de CSS é um método efetivo de estilizar uma única página. Contudx o, usar esse estilo em múltiplas páginas pode consumir muito tempo, já que você precisa definir as regras CSS para cada página do seu site;

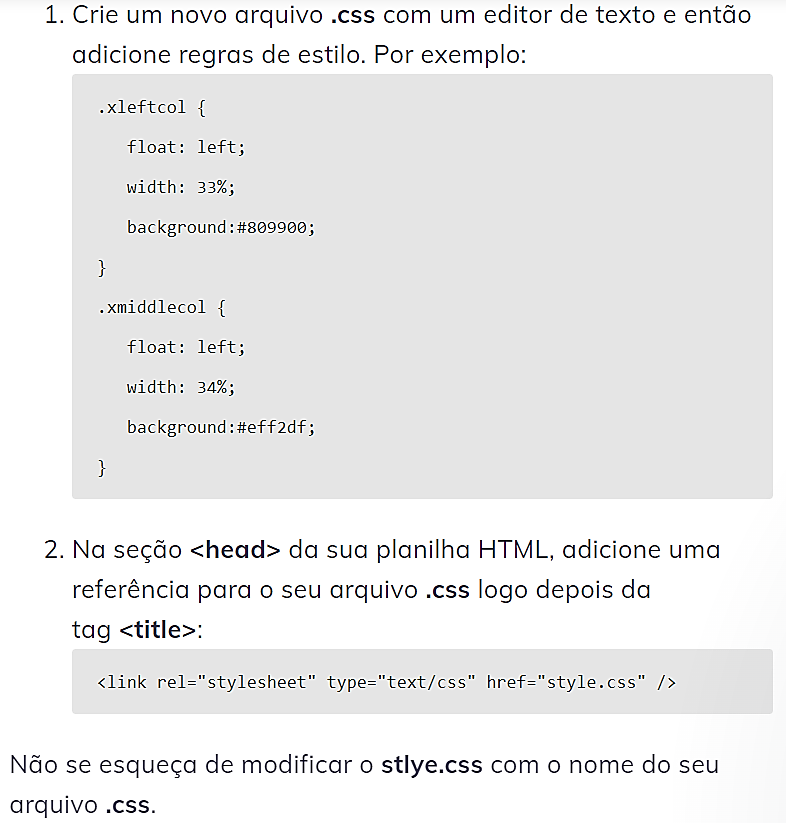
**CSS externo:** método mais eficiente, onde você vai linkar as páginas da internet com um arquivo .css externo,que você pode criar usando qualquer editor de texto no seu dispositivo especialmente se você está estilizando um site grande. Ao editar um arquivo .css, você pode modificar um site inteiro de uma só vez;

**CSS *inline*:** usado para dar estilo a um elemento HTML específico. Para este estilo de CSS, você somente vai precisar adicionar o atributo style para cada tag HTML, sem usar os seletores. Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual. Gerenciar o seu site pode se tornar uma tarefa bem difícil de você só usa o CSS *inline*. Contudo, o CSS *inline* no HTML pode ser útil para algumas situações. Por exemplo, em casos onde você não tem acesso aos arquivos CSS ou precisa aplicar estilos para um elemento único.

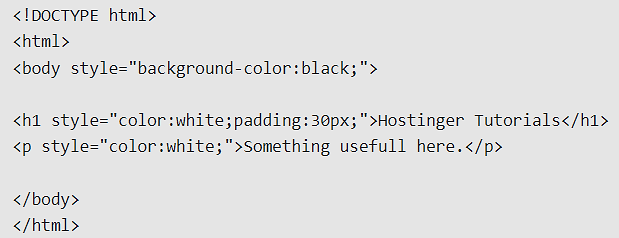
**EXEMPLO DE CSS INTERNO**



**EXEMPLO DE CSS EXTERNO**

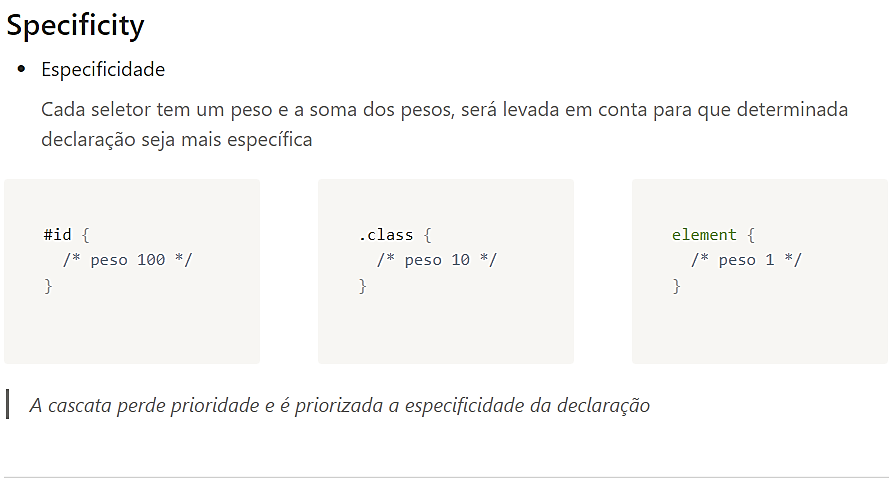


**EXEMPLO DE CSS *INLINE***







****

**CSS: REPRESENTAÇÃO DE CORES**

**Nome da cor:** *green*, *yellow*, *red*, etc.;

**Código hexadecimal para cores**: consiste em **seis letras ou números** precedidos do “**#**”, onde os dois primeiros elementos representam a intensidade de vermelho, o terceiro e quarto elementos representam a intensidade de verde e os dois últimos a intensidade de azul. Os valores variam entre **00 (mais escuro)** até **FF (mais claro)**. Ex.: #FFFFFF que representa a cor branca, #FF0000 que representa a cor vermelha, ou #000000 que representa a cor preta;

**RGB:** ou padrão *Red*, *Green*, *Blue* é baseado nas cores-luz, e é utilizado em basicamente todos os dispositivos eletrônicos que tenham uma tela colorida. Nele, as tonalidades base são o vermelho (R), verde (G) e azul (B). Cada uma delas tem um valor que vai de 0 a 255. Ex.: rgb(255,255,255) representa a cor branca, rgb(255,0,0) representa a cor vermelha, ou rgb(0,0,0) que representa a cor preta.

**CSS: APRESENTAÇÃO DOS TEXTOS**

**<style>** é inserida dentro da **<head>** e, dentro dessa tag, podemos colocar marcações de CSS referentes aos elementos que temos no nosso HTML **</style>**

**<link rel=”stylesheet” href=”**nome do arquivo **.css** desejado**” >**

**Obs.:** lembrar que no **arquivo .css** é necessário prestar atenção na **hierarquia dos elementos**;

**text-align** = alinhamento;

**font-size** = tamanho da fonte;

**background** = cor de fundo;

**color** = cor do texto;

***border***: cria uma borda ao redor do elemento;

***border-radius***: faz os desenhos de cantos arredondados nas bordas;

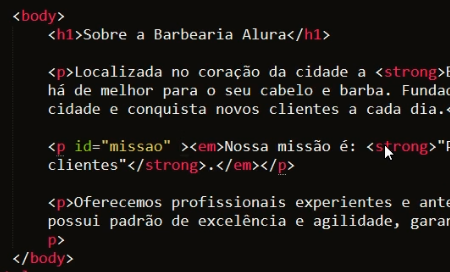
***text-transform: uppercase*** (transforma o texto para ter todas as letras maiúsculas);

***font-weight: bold*** (deixa o texto em negrito com CSS);

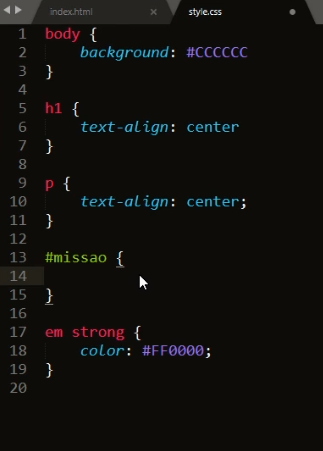
***text-decoration: none*** (remove a decoração do texto);

**CSS: SELETORES DE *ID* (identificadores)**

No HTML, podemos ter vários parágrafos, ou seja, vários elementos e tags iguais. Como fazemos para diferenciá-los e estilizar cada um separadamente? Para isso, utilizamos o marcador de identificação (**id**). Eles **selecionam quais elementos um conjunto de regras CSS se aplica**. Na prática, no arquivo HTML colocaremos **id = “”** depois do(a) elemento/tag desejado(a). Ex.:



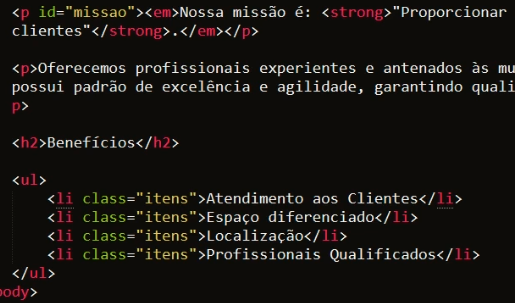
Logo após nomear o **id**, vamos recortar o código CSS e colar no arquivo CSS. O “missao" é um dos parágrafos, ele vem logo depois do anterior. E como fazemos para marcar? Toda vez que estamos marcando uma tag e queremos referenciar ela no CSS, usamos o nome da tag. Toda vez que temos um identificador no HTML e queremos referenciar ele no CSS, usamos a **#.** Logo: #missao.



**CSS: CLASSES**

De maneira simples, uma classe é uma maneira de identificar um **grupo** de elementos, como por exemplo, **class="**button**"**; já o **id** é um identificador **único** para um elemento. Em resumo, **Id**s servem para **itens únicos** do código, já as **classes** servem para **itens repetidos**. As classes servem para replicarmos a aplicação do nosso CSS, sem precisar copiar e colá-lo em vários elementos. Aplicamos a classe para isso.

Os elementos no HTML podem conter várias classes, exemplo: **class="**button btn-success**"**; mas somente um único ID: **id="**myButton**"**.



**CSS: PSEUDO-CLASSES**

Pseudoclasse é uma palavra-chave adicionada a seletores que especifica um estado especial do elemento selecionado. Ela permite que você aplique um estilo a um elemento não apenas em relação ao conteúdo da árvore do documento, mas também em relação a fatores externos como o histórico de navegação (***:visited***, por exemplo), o status do seu conteúdo (como ***:checked*** em certos elementos de um formulário), ou a posição do mouse (como ***:hover***, que permite saber se o mouse está sobre um elemento ou não).

Segue aqui o link do índice de pseudoclasses padrão: <https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-classes#%C3%ADndice_de_pseudo-classes_padr%C3%A3o>.

**CSS: DIMENSÕES**

Propriedade **height**: ajusta a **altura** do elemento;

Propriedade **width**: ajusta a **largura** do elemento;

Propriedade **padding**: ajusta o **espaçamento interno** do elemento. O elemento aumenta o seu tamanho;

Propriedade **margin**: ajusta o **espaçamento externo** do elemento.

**CSS: PROPRIEDADE *DISPLAY***

Permite que você defina a maneira como determinado elemento HTML deve ser renderizado, se um elemento é tratado como um bloco ou elemento *inline* e o layout usado para seus filhos, como layout de fluxo, *grid* ou *flex*.

Formalmente, o ***display*** define os tipos de exibição internos e externos de um elemento. O tipo externo define a participação de um elemento no layout de fluxo; o tipo interno define o layout dos filhos. Alguns valores de *display* são totalmente definidos em suas próprias especificações individuais. Mas por enquanto só estudaremos os comportamentos ***inline***, ***block*** e o ***inline-block***.

***display:* *block***: **permite** o redimensionamento tanto da largura quanto de sua altura do elemento ao qual ele está marcando. No entanto, **não permite** elementos vizinhos, pois "ocupam" toda a largura da linha em que estão inseridos;

***display: inline***: **não permite** a definição nem da largura e nem da altura, porém **permite** a exibição de outros elementos ao seu lado;

***display: inlin-block***: já os elementos definidos por este são a mistura entre ***inline*** e ***block***. Elementos que tenham essa definição **permitem** que outros elementos sejam definidos ao seu lado, **e permitem** o redimensionamento tanto da largura quanto de sua altura. É exatamente a característica que se quer nos ícones, que fiquem ao lado um do outro e se possa definir suas dimensões.

**CSS *RESET***

O navegador, por tentar deixar o conteúdo mais legível, sai botando um monte de estilo para a gente. Ou seja, ele cria o estilo para todas as tags que ele acha adequadas. Não é uma coisa que nós queremos pois não estamos exibindo um conteúdo qualquer, não estamos controlando a apresentação. Nós dominamos completamente o que estamos exibindo, já o navegador, nesse caso, está atrapalhando e, para anular/zerar o que o navegador faz – e o navegador faz muita coisa, ele mexe em muitas tags, mexe em muitos comportamentos – nós precisamos zerar a margem, o espaçamento tanto do body, quanto do HTML. Enfim, a gente precisa fazer um monte de detalhes e como isso é uma função que é repetida muitas vezes, por muita gente, a comunidade se juntou e criou um arquivo que é o Reset CSS, o qual remove os estilos que o navegador cria automaticamente. Segue o código:

/\* http://meyerweb.com/eric/tools/css/reset/

   v2.0 | 20110126

   License: none (public domain)

\*/

html, body, div, span, applet, object, iframe,

h1, h2, h3, h4, h5, h6, p, blockquote, pre,

a, abbr, acronym, address, big, cite, code,

del, dfn, em, img, ins, kbd, q, s, samp,

small, strike, strong, sub, sup, tt, var,

b, u, i, center,

dl, dt, dd, ol, ul, li,

fieldset, form, label, legend,

table, caption, tbody, tfoot, thead, tr, th, td,

article, aside, canvas, details, embed,

figure, figcaption, footer, header, hgroup,

menu, nav, output, ruby, section, summary,

time, mark, audio, video {

    margin: 0;

    padding: 0;

    border: 0;

    font-size: 100%;

    font: inherit;

    vertical-align: baseline;

}

/\* HTML5 display-role reset for older browsers \*/

article, aside, details, figcaption, figure,

footer, header, hgroup, menu, nav, section {

    display: block;

}

body {

    line-height: 1;

}

ol, ul {

    list-style: none;

}

blockquote, q {

    quotes: none;

}

blockquote:before, blockquote:after,

q:before, q:after {

    content: '';

    content: none;

}

table {

    border-collapse: collapse;

    border-spacing: 0;

}

**CSS: *POSICION***

É uma propriedade utilizada para determinar o posicionamento dos elementos na página. Ela é usada em conjunto com propriedades auxiliares, entre elas: *bottom*, *top*, *left* e *right*, que ajudam a definir de que forma o elemento será posicionado na tela. É importante entender como o ***CSS position*** funciona, pois essa propriedade pode sobrepor elementos, além de ser utilizada para aplicar efeitos interessantes, como a fixação do cabeçalho ao rolar a página. Por padrão, seu valor é sempre definido como ***static***, ou seja, acompanha o fluxo normal do posicionamento na página. Portanto, se nada for especificado, ele será posicionado conforme a ordem natural dos elementos usados na página. Dentre os seus **valores** estão:

**position** **static**: determina que o elemento utilizará o posicionamento padrão, ou seja, atribuirá o comportamento natural na página. É importante dizer que, ao utilizar esta condição, as propriedades *bottom*, *top*, *left* e *right* atribuídas ao elemento são ignoradas pelo navegador;

**position relative**: determina que o elemento será posicionado de acordo com o fluxo natural da página. Entretanto, ao utilizá-lo em conjunto com as propriedades auxiliares, o posicionamento será ajustado de acordo com as indicações. Por exemplo, se definirmos left: 20px, o elemento será afastado 20 pixels à esquerda a partir de seu ponto de origem;

**position fixed**: indica que o elemento terá um valor fixo em relação ao ***viewport***, ou seja, à área visível da tela;

**position absolute**: indica que o elemento ficará em uma posição fixa em relação à página (elemento pai);

**position sticky**: tem o comportamento variado. No primeiro momento, ela funciona com a posição relativa até atingir determinado limite, que pode ser a altura da tela ou o tamanho de um container, caso ele esteja inserido em uma <div>, por exemplo. Ao fazer a rolagem da página e atingir o limite do bloco, o elemento definido como sticky terá o comportamento como fixed e respeitará os valores definidos pelas propriedades auxiliares para indicar a posição. Essa propriedade é utilizada, por exemplo, para manter o header fixo no topo ao rolar a página. Ou seja, a posição do elemento é relativa até que ultrapasse os limites do bloco;

**position initial**: definir o valor de position para ***initial*** é o mesmo que dizer ao navegador para utilizar o valor padrão definido para aquele elemento, que é o posicionamento ***static***.;

**position inherit**: herda as características de posicionamento do elemento pai.

Dentre suas **propriedades** estão:

***bottom***: posiciona o elemento conforme a distância indicada e utiliza a área inferior da tela como ponto de início;

***top***: posiciona o elemento a partir do topo da página;

***left***: posiciona o elemento a partir do lado esquerdo da página;

***right***: determina a posição do elemento a partir do lado direito da página;

***clip***: define a área em que o conteúdo de um elemento será visualizado. Essa propriedade só pode ser usada em conjunto com o posicionamento ***absolute*** ou ***fixed***;

***overflow***: é utilizada para indicar qual ação o navegador deve tomar quando o tamanho do elemento exceder o limite determinado. Os valores possíveis são:

* ***scroll***: para adicionar barras de rolagens na vertical e horizontal, mesmo que uma delas fique desabilitada;
* ***hidden***: para esconder o conteúdo em excesso;
* ***auto***: adiciona barras de rolagem de acordo com a necessidade, ou seja, se o excesso de conteúdo for na direção horizontal, apenas essa barra será adicionada;
* ***visible***: valor padrão para a propriedade e exibe o conteúdo fora dos limites do elemento.

***z-index***: é utilizada para ordenar os elementos na página, especialmente quando eles se sobrepõem. Na prática, funciona como um indicador de camadas, no qual o valor maior corresponde à posição superior. É importante dizer que o ***z-index*** só funciona em conjunto com as propriedades ***absolute***, ***relative*** ou ***sticky***.

Quando colocamos uma imagem de fundo em um elemento, o CSS, por padrão, copia e cola a imagem diversas vezes até ocupar todo o espaço do elemento

A tabela Unicode

Uma revisão do conteúdo aprendido no treinamento anterior

Uma introdução ao projeto do treinamento

A criação da página de contato

Um pouco sobre os formulários

A criar um formulário HTML

A tag que o representa é a <form>

A tag <input>, para a entrada de dados do usuário

A criar uma etiqueta para o input, com a tag <label>

A conectar um input com o seu label

Colocamos um id para o input e associamos esse id ao atributo for do label

Alguns tipos de input, como text e submit

Que label possui o display inline e o input possui display inline-block

A estilizar o nosso formulário

O textarea, para entradas de texto de mais de uma linha

O input do tipo radio

Como agrupar vários input do tipo radio, impedindo que mais de um input seja selecionado

O input do tipo checkbox

Que podemos criar um input dentro de um label, assim associando-os

Mais estilizações para a nossa página

Como funciona a hierarquia no CSS

O select, que é seletor, um campo de seleção de um item, e o option, que representa cada opção do seletor

Alguns tipos de inputs para celular: email, tel, number, password, date, datetime, month e search

Como não permitir que um campo não seja preenchido, através do atributo required

Como exibir uma sugestão de preenchimento para os campos, através do atributo placeholder

Como deixar uma opção marcada por padrão nos nossos input radio e checkbox, através do atributo checked

Como estruturar melhor o nosso código com fieldset e legend

Como adicionar uma alternativa à imagem, descrevendo-a, com o atributo alt

Como estilizar o botão de envio de formulário

A realizar transições nos nossos elementos, com a propriedade CSS transition

A modificar o estilo do ponteiro do mouse, quando passar por cima de determinado elemento, através da propriedade CSS cursor

A realizar transformações nos nossos elementos, como aumentar proporcionalmente a escala de determinado elemento ou rotacioná-lo, através da propriedade CSS transform

A criar uma tabela HTML

A tag table, que representa a tabela

A tag tr, que representa a linha da tabela

A tag td, que representa a célula da tabela

A tag thead, que representa o cabeçalho da tabela

A tag tbody, que representa o corpo da tabela

A tag th, que representa a célula do cabeçalho da tabela

A tag tfoot, que representa o rodapé da tabela

A estilizar a tabela

A ajustar a página principal para utilizar os mesmos padrões da página de produtos

Medidas proporcionais com CSS

Como funciona a flutuação dos elementos e como modificá-la, com a propriedade float do CSS

Como limpar o float, com a propriedade clear do CSS

A utilizar fontes externas nas nossas páginas

Como incorporar um mapa à nossa página

Como incorporar um vídeo à nossa página

A melhorar mais ainda a semântica da página principal, com novas divisões, classes, etc

Novas pseudo-classes

Como aplicar um background gradiente na página

Pseudo-elementos

* Seletores avançados CSS
  + Seletor >, para acessar os filhos de determinado elemento. Por exemplo, para acessar todos os p dentro de main:
  + **main** > **p** {

}COPIAR CÓDIGO

* + Seletor +, para acessar o primeiro irmão de determinado elemento. Por exemplo, para acessar o primeiro p após um img:
  + **img** + **p** {

}COPIAR CÓDIGO

* + Seletor ~, para acessar todos os irmãos de determinado elemento. Por exemplo, para acessar todos os p após um img:
  + **img** ~ **p** {

}COPIAR CÓDIGO

* + Seletor not, para acessar os elementos, exceto algum. Por exemplo, para acessar todos os p dentro de main, exceto o p que tem id missao:
  + **main** **p**:not(**#missao**) {

}COPIAR CÓDIGO

* Como fazer contas com CSS, com a propriedade calc

Como manipular a opacidade dos elementos, com a propriedade CSS opacity

Como manipular a opacidade das cores

Como adicionar um sombreamento em volta dos elementos, com a propriedade CSS box-shadow

Como adicionar um sombreamento em textos, com a propriedade CSS text-shadow

Design responsivo: como ajustar o estilo da nossa página de acordo com o tamanho da tela do dispositivo que a acesse

Meta tag de Viewport

Media Queries

Como adicionar vídeos:

<div class="video">

<iframe width="560" height="315" src="https://www.youtube.com/embed/wcVVXUV0YUY" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

</div>