Material do Curso de HTML5 CSS3

HTML

Estrutura básica de um documento HTML

- **<!DOCTYPE html> – A tag !DOCTYPE informa ao navegador a versão do HTML** que está sendo utilizada no documento. Por exemplo: no HTML5, basta incluir !DOCTYPE html, e assim o navegador já saberá que se trata de um documento na versão HTML5;
- **<html></html>** – Esta tag é o elemento básico da estrutura do HTML. Assim sendo, ela conterá todos os elementos do seu documento. Todo documento HTML deve iniciar e finalizar com essa tag;
- **<head></head>** – Essa tag delimita o cabeçalho do documento. Não possui nenhum valor visível, porém é capaz de transmitir ao navegador diversas informações muito úteis e essenciais a uma boa apresentação do seu documento HTML;
- **<title></title>** – Essa tag define o título da sua página, o nome que aparecerá na sua aba, janela ou guia. Por esta razão, a tag >title> é de grande importância para o SEO;
- **<meta/>** – Essa tag permite inserir metadados ao seu documento, no caso acima, a informação **charset="UTF-8"**, que garante a compatibilidade do código com os caracteres de padrão latino americano;
- **<body></body>** – Finalmente, a tag que representa o corpo do documento. Em síntese, é nessa tag que todos os elementos visíveis do seu site devem ser inseridos.

Tags de comentários em HTML

<!-- Meu comentário em HTML -->

Atributos

- **class="…"** – Atribui uma classe ao elemento (uma classe pode ser utilizada para um ou mais elementos);
- **id="…"** – Atribui um id ao elemento (um id deve ser único, ou seja atribuído a um único elemento);
- **style="…"** – Permite incluir elementos CSS (estilos) dentro da tag;
- **lang="…"** – Define o idioma principal do elemento;
- **title="…"** – Define o título do elemento;
- **alt="…"** – Define um texto alternativo e, por isso, é muito utilizado em imagens, auxilia nas práticas de SEO;
- **hidden** – Oculta o elemento;
- **align="…"** – Permite definir o padrão de alinhamento desse elemento, como por exemplo: right, center, left e justify;
- **width="…"** – Define uma largura para o elemento;
- **height="…"** – Define uma altura para o elemento.

Tags HTML estruturais

- **<header></header>** – Essas tags definem um cabeçalho. Portanto, todo conteúdo que estiver dentro dela faz parte de um cabeçalho, podendo ser utilizado dentro de outras sessões. Não confundir com as tags <head>;
- **<main></main>** – Essas tags representam o conteúdo principal do seu corpo, ou seja, o conteúdo relacionado diretamente com o tópico central da página ou com a funcionalidade central da aplicação;
- **<footer></footer>** – Essas tags definem um rodapé para a página, geralmente utilizadas no final da página;
- **<section></section>** – Essas tags definem uma sessão para sua página;
- **<article></article>** – Essas tags definem um artigo da sua página. Nesse sentido, são utilizadas para separar o conteúdo da sua página. Muito utilizado principalmente por blogs ou páginas de conteúdo;
- **<aside></aside>** – Essas tags representam uma seção de uma página cujo conteúdo é tangencialmente relacionado ao conteúdo do seu entorno, que poderia ser considerado separado do conteúdo;
- **<nav></nav>** – Essa tag define um conteúdo de navegação. Portanto, é muito utilizado em conjunto com listas e na criação de menus;
- ** ** – Define uma divisão da página. Desta forma, funciona como um container para conteúdo de fluxo. Uma vez que não possui um valor semântico, é muito utilizado para organizar melhor o conteúdo. Anteriormente ao HTML5, era utilizado no lugar das categorias acima.

A tags HTML de conteúdo

Tags HTML de título

<h1></h1> - Título de maior valor hierárquico
<h2></h2>
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6> - Título de menor valor hierárquico

Tags HTML de texto

- ** ** – Principal tag de texto, compõe um parágrafo;
- **** – Apesar de ter uma funcionalidade e características parecidas com os parágrafos, costumam ser utilizadas apenas para pequenas informações, como legendas de um formulário, legendas de uma imagem, etc. Também pode ser utilizada para formar um container;
- **<pre></pre>** – Tag utilizada para representar texto pré-formatado. Muito utilizada para inserir códigos;
- **** – Transforma o conteúdo em **negrito**;
- **** – Transforma o conteúdo em itálico;
- **
** – Essa tag não necessita de fechamento, ela executa a função de quebra de linha.
- **<hr/>** – Essa tag não necessita de fechamento, ela forma uma linha horizontal.

Tag de link HTML (Âncora)

Para realizar um link, podemos chamar as tags **<a>** com o atributo href. Por exemplo, caso você queira criar um link no seu texto que redirecione à página inicial do google:
Para acessar o Google, clique aqui.

Tags HTML de multimídia

<img>
<img src="imagens/imagem1.jpg" alt="imagem 1 da minha página">

<video> e <audio>

```
<video controls>
  <source src="video.mp4" type="video/mp4">
  <source src="video.ogg" type="video/ogg">
  Seu navegador não possui suporte para Vídeos.
</video>

<audio controls>
  <source src="musica.ogg" type="audio/ogg">
  <source src="musica.mp3" type="audio/mpeg">
  Seu navegador não possui suporte para áudio.
</audio>
```

<iframe>

Os iframes são muito utilizados na atualidade, servem para incluir recursos de uma outra página nesta página. Vale a pena conferir o exemplo da W3C Schools, pagina de tutorias pertencente ao grupo W3C, a organização atualmente responsável pelos padrões da web.
Portanto, para inserir um iframe, basta utilizar a tag com o atributo **src**. Além disso, é possível incluir um texto dentro do elemento, caso o navegador do usuário não possua suporte para tal.
Vejamos então o exemplo abaixo:

```
<iframe src="https://www.homehost.com.br">
  <p>Seu navegador não possui suporte para iFrames.</p>
</iframe>
```

Tags HTML de listas

Para poder criar uma lista, podemos utilizar uma lista ordenada, a partir das tags **
**, ou uma lista não ordenada, a partir das tags **
**. Posteriormente, incluímos dentro da lista os elementos da mesma, dentro das tags **
**.
Vejamos os exemplos a seguir:

```
<p>Minha lista ordenada:</p>
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
```

```
</ol>
<p>Minha lista não ordenada:</p>
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ul>
```

Tags HTML de formulário

Para iniciar um formulário, incluimos as tags **<form>** e **</form>**. Posteriormente, devemos incluir o conteúdo do formulário. Digamos que, por exemplo, você queira incluir três campos, sendo dois para coletar informações e um para receber uma mensagem. Desta forma, para criar os campos de preenchimento, deverá utilizar a **tag <input>**, enquanto para o campo referente à mensagem, deverá utilizar a tag **<textarea>**.

A tag <input>

A tag **<input>** possui o atributo **type**, que varia entre diversos tipos (vamos explicar os principais deles abaixo). Também há o atributo **placeholder**, que é um texto que **ficará disponível** enquanto nada for digitado nesse campo. Também é importante definir um atributo **name** para cada input.

- **<input type="text">** – Define um campo que receberá qualquer caractere;
- **<input type="email">** – Define um campo que receberá caracteres e verificará se o mesmo consiste em um e-mail válido;
- **<input type="submit" value="ENVIAR">** – Define um botão que servirá para o envio do formulário. Dentro dele, podemos atribuir o value, que será o texto dentro do botão de envio.

Existem outros tipos de **<inputs>** que podem ser estudados na **documentação disponível pela W3C**.

As tags <textarea></textarea>

Assim como a tag **<input>**, essa tag define um campo para o formulário. Porém, diferentemente, ela tem como principal característica ser uma área de preenchimento de texto, ou seja, permite que o usuário escreva um texto ou uma mensagem no seu interior. Também traz opções para que o usuário redimensione seu tamanho (resize). Dessa forma, podemos incluir uma area de texto utilizando as tags **<textarea>** e **</textarea>**.

Tags de estilos e scripts

Para podermos concluir esse tutorial, não poderíamos deixar de citar as tags **<style>** e **<script>**.

A tag **<style>** e **</style>** deve ser incluída no **<head>** do seu código HTML. Dentro dessa tag, é possível incluir todo o seu código CSS, ou seja, seu código de estilos.

Já a tag **<script>** e **</script>** tem como objetivo incluir códigos de scripts ao seu HTML, podendo ser incluída em qualquer parte. Contudo, recomenda-se fortemente que seja inserida após o **<footer>**. Dessa forma, podemos incluir nela um código javascript.

CSS

Como adicionar CSS ao documento

Inline: dentro da linha de comando do próprio HTML, dentro da tag <body>.

<tag style=”css” />

Ex.:
<div style=”background-color: blue”> !-- Conteúdo --!

Internal: dentro do campo correspondente à tag <head>. No próprio documento HTML.

Ex.:
```
<html>
	<head>
		<style>

	!-- Conteúdo CSS --!

		</style>
	</head>
</html>
```

External: Cria-se um documento CSS à parte. Estabelece-se um vínculo entre os documentos.

```
<html>
	<head>
		<link rel=”stylesheet” href=”style.css”/>
	</head>
</html>
```

SELETORES

Element Selector:

Ex.:
```
h1 {
color: blue
}
```

(Serve para outras tags como: a, p, li etc...)

Class Selector:

Ex.:

```
.note {
  font-size: 20px;
}
```

(Nesse caso, todos os elementos html que forem rotulados com a mesma class serão afetados pela ação do CSS).

Id Selector:

Ex.:
```
#id-selector-demo {
  color: green;
}
```

(O Id, pelo que entendi, é uma rotulação individual).

Attribute Selector:

```
li[value="4"] {
  color: blue;
}
```

(Nesse caso, as tags com o mesmo valor de atributo serão afetadas).

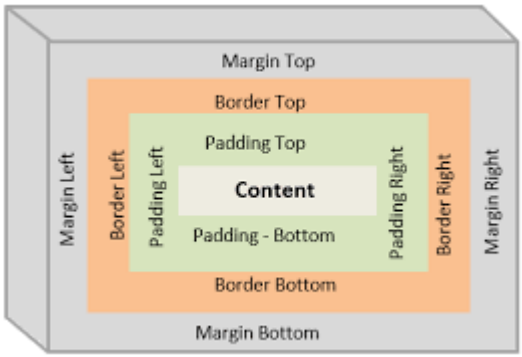
Ex.:
No arquivo html: Drag Me
No arquivo CSS: p [draggable=”true”] {
color: blue;
}

Universal Selector:

```
* {
  text-align: center;
}
```

(O comando aplica-se a tudo).

CSS Box Model



Ex.:

```
div {
    width: 200px;
    height: 200px;
  }

  #first {
    background-color: cadetblue;
    padding: 20px;
    border: 10px solid black;
  }

  #second {
    background-color: gold;
    border: solid black;
    border-width: 20px 10px;
    margin-left: 260px;
  }
```

Combinando Seletores:

```
/* Group */
h1,h2 {
 color: blueviolet;
}

/* Child */
.box > p {
 color: firebrick;
}

/* Descendent */
.box li {
 color: blue;
}

/* Chained */
li.done {
 color: green;
}

/* Multiple Combiners */
ul p.done {
 font-size: 0.5rem;
```

CSS Position:

A propriedade position em CSS controla como um elemento é posicionado num documento HTML, afetando o layout e a forma como os elementos se relacionam uns com os outros. Ela permite alterar a posição de um elemento, tirando-o do fluxo normal da página e posicionando-o de forma flexível.

- static:

Posicionamento padrão, o elemento segue o fluxo normal da página e não pode ser deslocado.
O *position static*, que em português significa estático, é o mais utilizado de todos, pois trata-se do valor que já vem por padrão quando criamos elementos no HTML. Seu comportamento, como o nome sugere, é ser estático. Ele apenas segue o fluxo junto dos outros elementos da página normalmente, tendo o canto superior esquerdo como referência.
Este elemento não aceita as propriedades auxiliares *top*, *bottom*, *left* e *right*. No exemplo abaixo, mesmo aplicando o código position: static e top: 20px, o elemento não move 20 pixels do topo para baixo. Ele continua estático sendo alinhado no canto superior esquerdo.

- relative:

O elemento é posicionado com base na sua posição original, deslocando-se a partir daí, mas sem sair do fluxo normal.
O *position relative*, que em português significa relativo, é usado quando queremos alterar a posição de um elemento tendo como referência a posição inicial dele. Ao aplicarmos essa propriedade em um elemento, ele não muda de posição, pois já vai estar posicionado em sua posição de referência. Porém, quando aplicamos top: 50px, e left: 50px no elemento, sua posição se altera 50 pixels de cima para baixo, e da esquerda para direita tendo como referência a sua posição inicial.

- absolute:

O elemento é retirado do fluxo normal e posicionado em relação ao seu elemento pai (ou ao documento, se não houver pai posicionado).
Position absolute, ou posicionamento absoluto em português, possui dois comportamentos diferentes. O primeiro é quando o elemento com essa propriedade possui um elemento pai de valor diferente de *static*. Neste caso, ele terá este elemento pai como referência para ser posicionado. Todo elemento pai que tiver qualquer valor de *position*, menos o *static*, será a referência para posicionar o elemento filho absolute!
O segundo comportamento é quando o elemento com *position absolute* não tem elemento pai, ou este elemento pai possui *position static*. Nesta situação, ele irá ignorar estes elementos e será posicionado a partir do canto esquerdo superior do documento, podendo até mesmo sobrepor a eles. Ou seja, é importante utilizar o *absolute* tendo certeza da necessidade do seu uso (não que isso não seja necessário para os outros valores), pois ele pode desalinhar o layout da página.

- fixed:

O elemento é retirado do fluxo normal e posicionado em relação ao viewport (a área visível do navegador), permanecendo fixo mesmo com rolagem.
O *position fixed*, que em português significa fixo, faz com que o elemento que recebeu esta propriedade não se mova na tela. Mesmo que uma página tenha rolamento (scroll). Ao utilizarmos o rolamento para esquerda e direita, ou para cima e para baixo, o elemento não se move.

- sticky:

O elemento é posicionado como relative até que atinge um ponto de rolagem definido, onde passa a ser posicionado como fixed.
O *position sticky*, que em português significa pegajoso ou colado, é parecido com o fixed, porém a sua diferença é que, em vez dele ficar fixo em relação à tela, ele fica fixo em relação ao rolamento da página.
Este tipo de position é utilizado sempre em conjunto das propriedades auxiliares, pois precisa de um posicionamento de referência. Abaixo é possível conferir isso. Aplica-se top: 0%; e left: 0%, que faz com que quando utilizado o rolamento para baixo, o elemento suba, porém quando chega no limite do topo da tela, ele fica "colado", e o mesmo acontece para o lado esquerdo, quando arrastamos a barra de rolagem para a direita, o elemento chega no limite da tela do lado esquerdo, no entanto, também fica "colado" no limite deste lado da tela. O elemento volta a ocupar a sua posição inicial quando voltamos com a barra de rolagem como antes.

CSS Display:

O CSS display é uma propriedade utilizada para organizar os elementos na página HTML. Ela pode conter diferentes valores, entre eles o flex e o grid, que possibilitam a criação de layouts responsivos. Dessa forma, podemos construir aplicações que são facilmente adaptadas aos diferentes tamanhos de tela.

Ou seja, display é uma propriedade que especifica o comportamento de exibição (o tipo de caixa de renderização) de um elemento.

Ex.:
```
p {display: none;}
div {display: inline;}
.exemplo {display: block;}
#exemplo {display: inline-block;}
```

ex.2:
```
.green {
    display: inline-block;
    width: 200px;
    height: 200px;
    background-color: green;
  }
```

No qual o atributo **value** pode conter os seguintes valores:

- inline;
- block;
- contents;
- flex;
- grid;
- inline-block;
- inline-flex;
- inline-grid;
- inline-table;
- list-item;

- run-in;
- table;
- table-caption;
- table-column-group;
- table-header-group;
- table-footer-group;
- table-row-group;
- table-cell;
- table-column;
- table-row;
- none;
- initial;
- inherit.

Renderizando o elemento HTML de forma responsiva e flexível: flex

O valor **CSS display flex** é utilizado para organizar os elementos HTML de forma responsiva. Na prática, ao definirmos um elemento com esse valor, ele funciona como um container para agrupar os elementos filhos, que são considerados **flex-itens** e são organizados nos sentidos horizontal ou vertical.
Ao definirmos o display como flex, podemos utilizar outras propriedades adicionais para estabelecer como os elementos serão dispostos na página ou em um espaço específico. São elas:

- **flex-direction**: determina a direção em que os elementos serão dispostos, que pode ser **row** (linha) ou **column** (coluna), ou ainda, **row-reverse** e **column-reverse** para exibir os itens na posição invertida;
- **flex-wrap**: para definir como o conteúdo de um elemento será exibido dentro do espaço. Em caso de texto, por exemplo, ele pode ser forçado a permanecer na mesma linha ou dividido conforme a necessidade. Os valores possíveis são: **nowrap** para não quebrar a linha, **wrap** para quebrar e **wrap-reverse**, em que o conteúdo é preenchido no sentido de baixo para cima;
- **flex-flow**: representa uma forma abreviada de escrever os valores dos atributos **flex-direction** e **flex-wrap**;
- **justify-content**: indica como os elementos filhos serão alinhados horizontalmente no espaço definido como container. Os valores possíveis são: **flex-start**, **flex-end**, **center**, **space-between**, **space-around** e **space-evenly**;
- **align-itens**: representa o alinhamento vertical dos elementos pertencentes ao container. Os valores possíveis são: **flex-start**, **flex-end**, **stretch**, **center** e **baseline;**
- **align-content**: funciona semelhante ao **align-itens**, entretanto, alinha o conteúdo verticalmente em relação às linhas.

Renderizando o elemento HTML como um container de grade: grid

O **CSS display grid** permite organizar os elementos da página de forma horizontal e forma vertical, simultaneamente. Para isso, ele também conta com uma série de propriedades adicionais para ajudar a definir como os itens serão dispostos na página ou no espaço disponível.

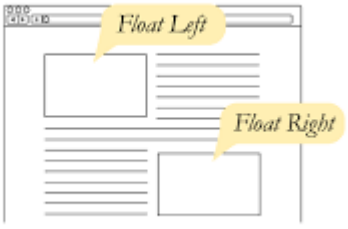
Basicamente, o **grid** é formado por um conjunto de linhas e colunas. Se nada for especificado, os elementos são organizados em bloco, ou seja, um em cada linha. Por isso, é preciso definir outras propriedades para organizar no formato desejado. São elas:

- **column-gap**: define o espaço entre as colunas;
- **row-gap**: define o espaço entre as linhas;
- **gap**: estabelece o mesmo espaço de distanciamento entre linhas (**row-gap)** e colunas (**column-gap)**;
- **grid-area**: determina a área do elemento, ou seja, quantas linhas e colunas o item ocupará no grid;
- **grid-auto-columns**: define o valor automático para a coluna;
- **grid-auto-flow**: indica o sentido em que os itens serão inseridos automaticamente na grade;
- **grid-auto-rows**: define o valor automático para a linha do grid;
- **grid-column-start**: especifica o início do grid;
- **grid-column-end**: especifica o final do grid;
- **grid-column**: define a junção das propriedades **grid-column-start** e **grid-column-end**;
- **grid-row-gap**: indica o tamanho do espaço entre as linhas;
- **grid-column-gap**: indica o tamanho do espaço entre as colunas do grid;
- **grid-gap**: representa a junção das propriedades **grid-row-gap** e **grid-column-gap**;
- **grid-row-start**: indica a linha em que o item será posicionado;
- **grid-row-end**: indica em que linha será o fim do item;
- **grid-row**: representa a união das propriedades **grid-row-start** e **grid-row-end**;
- **grid-template**: representa a união das propriedades **grid-template-rows**, **grid-template-columns** e **grid-areas**;
- **grid-template-areas**: especifica a área do grid com o nome dos itens do grid;
- **grid-template-columns**: estabelece o tamanho das colunas e a quantidade para o layout grid;
- **grid-template-rows:** indica o tamanho das linhas do grid.

Obs.: uma boa referência para entender de forma didática esse conteúdo é a página https://blog.betrybe.com/css/css-display/ .

CSS Float:

Em CSS, a propriedade float especifica como um elemento deve flutuar. O elemento flutuante será removido do fluxo normal da página, mas permanecerá como parte dele — ou seja, o elemento será deslocado para a esquerda ou direita até tocar a borda de seu contêiner ou de outro elemento flutuante.



```
img {
    float: left;
  }

  .dog {
    background-color: coral;
    float: right;
  }

.button {
 float: left;
 background: gray;
 font-size: 48px;
 color: white;
 padding: 3px;
 border-radius: 5px;
 border: 3px solid black;
 margin-right: 10px;
}
```