HTML e CSS

HTML e CSS - uma abordagem prática

Professor: **Eduardo Costa**

Introdução ao HTML

HTML é a linguagem utilizada para a construção de sites. É um acrônimo inglês Hypertext Markup Language ou em português Linguagem de Marcação de Hipertexto. Em resumo é a linguagem que você utiliza para demarcar o seu texto, onde você informa ao navegador web qual parte do seu documento se refere um determinado trecho. Ex: **<h1>**Título do documento**</h1>**. O texto anterior refere-se a um título de um documento qualquer e está delimitado entre as tags de abertura e fechamento <h1></h1>.

HTML não é uma linguagem de programação e sim de marcação. É utilizada para especificar as partes de um documento e ser exibida no navegador de internet como tal.

“HTML (abreviação para a expressão inglesa HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada na construção de páginas na Web. Documentos HTML podem ser interpretados por navegadores.” (Wikipédia)

Elementos de um site

Você sabe como funciona um website?

- Logomarca
- Títulos
- Menus de navegação

- Área de conteúdo
- Dentre outros

HTML, CSS e JAVASCRIPT

Um site Web é desenvolvido utilizando-se das tecnologias HTML, CSS E JavaScript no Front End:

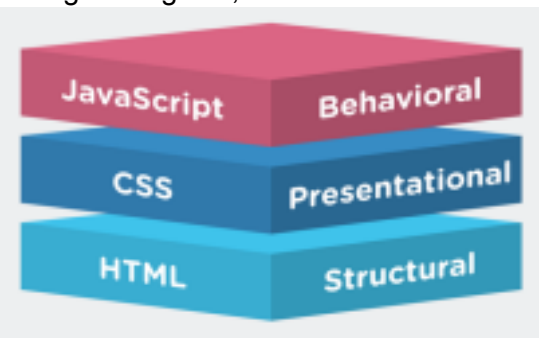
FRONT-END

Apresentação visual do sistema, marketing de negócio, etc.

HTML: É a estrutura do seu documento

CSS: Apresentação visual do documento

JavaScript: Interatividade e comportamento



BACK-END

Regras de negócio, API para disponibilizar acesso às regras de negócio etc.

Linguagens de programação - PHP, JAVA, C#, NODEJS (JavaScript) utilizadas para manipular as regras de negócio do sistema.

Banco de dados: Mysql, SQL Server, Postgres, Maria DB, Oracle - utilizados para armazenamento e manipulação dos dados do negócio.

Servidor de Aplicação: Apache, IIS, NodeJS - Servidor que controla o acesso e disponibilidade do sistema em geral.

Neste curso trabalharemos as principais características fundamentais de Front-End. Para mais detalhes sobre essas tecnologias, eu, professor Eduardo, indico a leitura complementar das seguintes documentações, cujas quais utilizaremos no decorrer do curso:

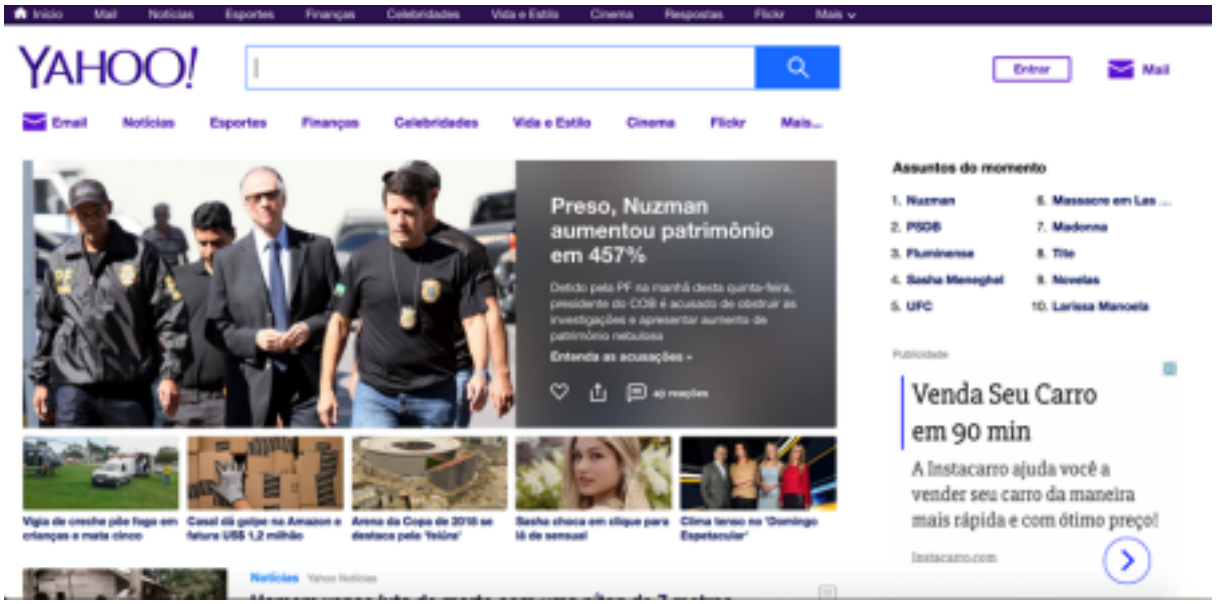
W3 Schools - https://www.w3schools.com/

MDN - Mozilla Developer Network - https://developer.mozilla.org/pt-BR/

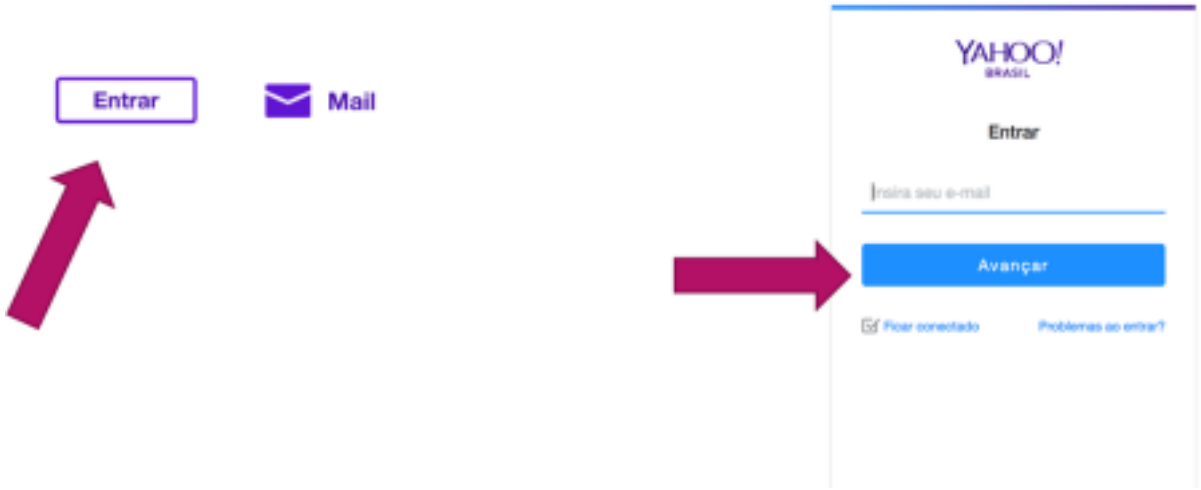
Você também poderá **testar** seus **códigos online** em: https://codepen.io/

SITE SEM O CSS

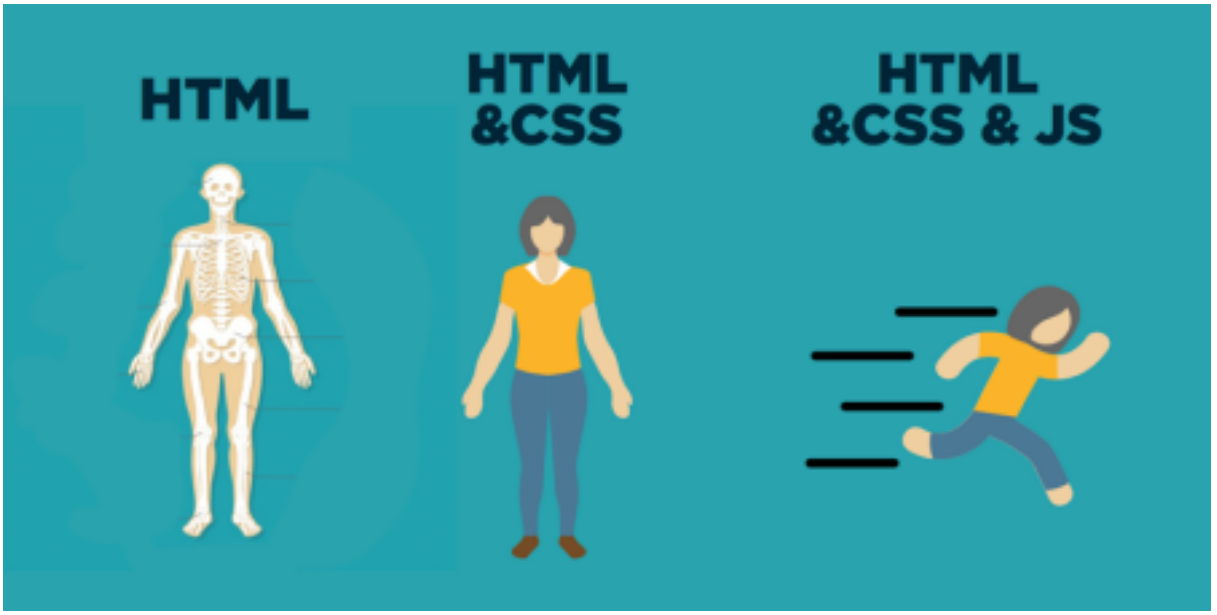
SITE COM O CSS



SITE COM JAVASCRIPT



Repare na imagem abaixo, o HTML:



O começo de tudo se dá então através do HTML, que por sua vez insere/chama os outros integrantes CSS e JavaScript.

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Título - aparece na aba do navegador</title>
    <link rel="stylesheet" href="endereço do arquivo de css">
    <script src="endereço do arquivo de scripts"></script>
</head>
<body>

</body>
</html>
```

Se HTML não é uma linguagem de programação, então como esse carinha age dentro do nosso site?

Através de agrupamento e organização de conteúdos a partir da estrutura de tags

Um arquivo HTML é um arquivo de texto especial. Ao invés de você o salvar com a extensão **.txt** ou **.doc** você deverá salvar como **.html:**

arquivo.html

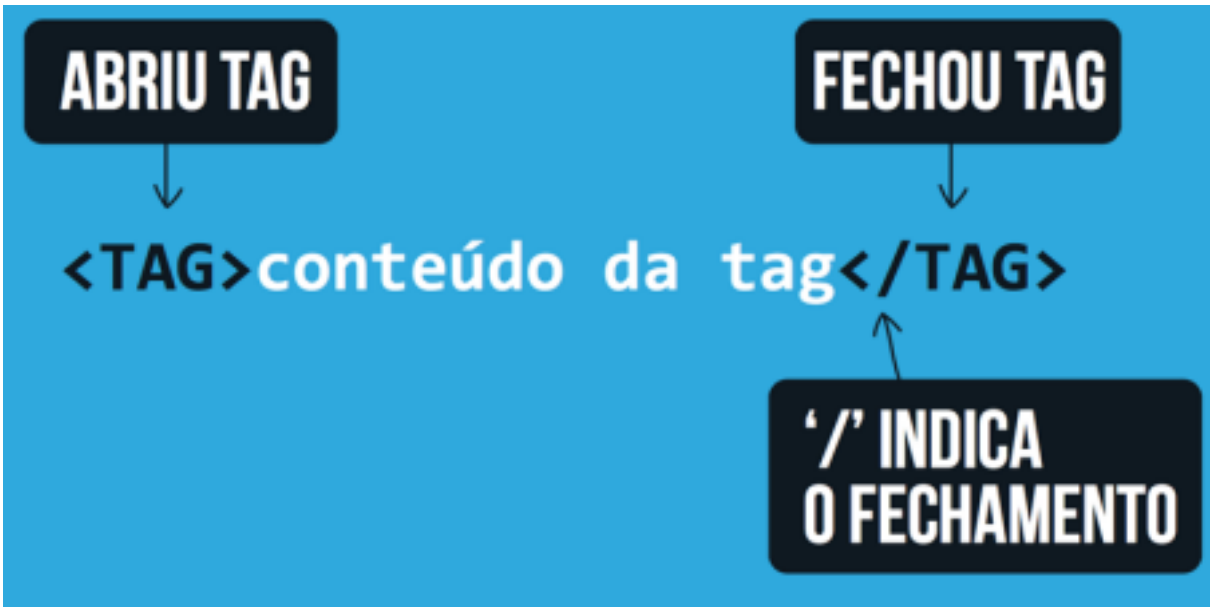
Estrutura Básica

A estrutura inicial do HTML5 é a seguinte:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>My Document</title>
</head>
<body>
    <h1>Boa noite é</h1>
</body>
</html>
```

Tags de marcação do documento

- Um documento HTML é composto por tags cujas quais contém nomes definidos entre os sinais **<** e **>**.
-
- A maioria das tags são fechadas pelo caracter barra */*.
-
- Ex: `<body> </body>` - par de chaves que demarca o corpo do documento html para o navegador.
-
- Outras Tags podem ser **<únicas >** como no caso de uma quebra de linha, ex: **
**



As tags podem conter atributos que são características internas com valores específicos, que adicionam mais informações à tag e podem alterar seu comportamento padrão

Ex: tag img com o atributo title de valor Título da imagem. <img **title**="Título da imagem">

Tags de marcação

Títulos

Em títulos de documentos utilizamos as tags **Hx** onde x é um número que representa o nível e varia de 1 até 6.

Ex:

```
<h1>HTML BÁSICO</h1>
```

```
<h2>Introdução aos documentos HTML</h2>
```

Parágrafos

Já para definir um texto dentro de um parágrafo utilizamos a tag P Ex:

```
  <h1>HTML BÁSICO</h1>
  <h2>Introdução aos documentos HTML</h2>
<p>
    O HTML é uma linguagem de marcação de hipertexto utilizada para
    representar documentos na web, através do navegador...
</p>
```

Quebra de linha

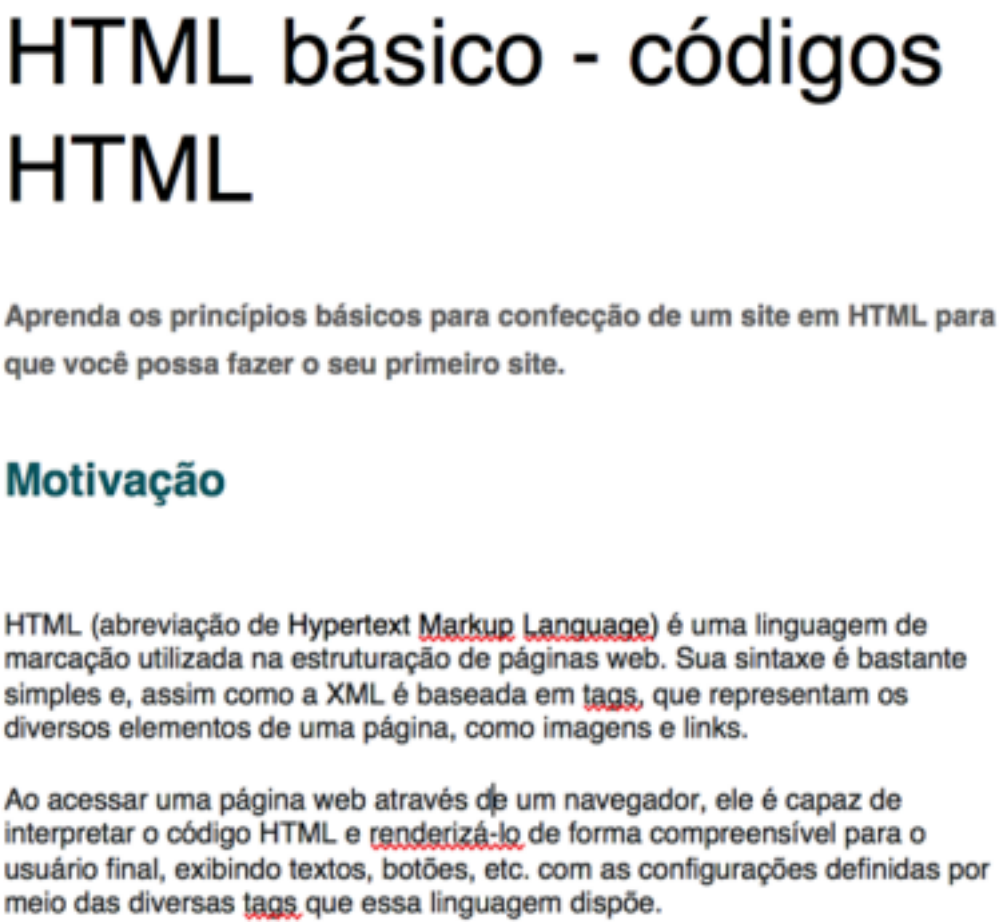
Quebra de linha utilizamos para informar ao navegador que o texto deve ser quebrado naquele momento

Ex:

```
 <p>
   <h1>HTML BÁSICO</h1>
   <h2>Introdução aos documentos HTML</h2>
O HTML é uma linguagem de marcação de hipertexto <br > utilizada para
representar documentos na web, através do navegador... </p>
```

Exercício de Aplicação

Produza o resultado da figura abaixo, com o nome titulos_e_paragrafos.html



Listas

As listas são utilizadas para representar uma lista de itens em um documento. Existem 3 tipos de listas:

- LISTAS ORDENADAS **OL><**
- LISTAS NÃO ORDENADAS **<UL>**
- LISTA DE DESCRIÇÃO **<DL>**

Não Ordenadas

As listas não ordenadas se dão pelas tags de listas <UL>:

```
<h1>LISTA DE COMPRAS</h1>
<h2>Itens para macarronada</h2>
<ul>
      <li>Macarrão<li>
<li>Molho<li>
<li>Carne moída<li>
<li>Queijo ralado<li>
<ul>
```

Ordenadas

As listas ordenadas se dão pelas tags de listas <OL> e atributo TYPE – 1 A a i I:

```
<h1>LISTA DE COMPRAS</h1>
<h2>Itens para macarronada</h2>
<ol>
    <li>Macarrão</li>
    <li>Molho</li>
    <li>Carne moída</li>
    <li>Queijo ralado</li>
</ol>
```

```
<h1>LISTA DE COMPRAS</h1>
<h2>Itens para macarronada</h2>
<ol type="i">
    <li>Macarrão</li>
    <li>Molho</li>
    <li>Carne moída</li>
    <li>Queijo ralado</li>
</ol>
```

```
<h1>LISTA DE COMPRAS</h1>
<h2>Itens para macarronada</h2>
<ol type="a">
    <li>Macarrão</li>
    <li>Molho</li>
    <li>Carne moída</li>
    <li>Queijo ralado</li>
</ol>
```

Lista com descrição

```
<h1>LISTA DE COMPRAS</h1>
<h2>Itens para macarronada</h2>
<dl>
    <dt>Macarrão</dt>
    <dd>Macarrão espaguete pra mim é melhor que o parafuso</dd>
    <dt>Molho</dt>
    <dd>Molho de macarrão tem que ser da marca quero</dd>
    <dt>Carne moída</dt>
```

```
    <dd>Tem que estar temperada com coentro, alho e cebolinha, aí é tudo
de bão :)</dd>
</dl>
```

Listas com sublistas

As listas com sublistas são, na realidade, listas dentro de listas:

```
<h1>Lista com sublista</h1>
    <ul>
        <li>Sobremesas</li>
        <li>Bebidas
            <ol>
                <li>pepsi</li>
                <li>coca cola</li>
            </ol>
        </li>

        <li>Comidas
            <ol type="a">
                <li>lazanha</li>
                <li>pizza</li>
            </ol>
        </li>
        <li>diversos</li>
        <li>outros</li>
    </ul>
```

Imagens

<IMG> tag única com os seguintes atributos ATRIBUTOS:

- **src**: endereço da imagem.
- **title**: Título caso não seja possível carregar.
- **alt**: Texto alternativo para descrição
- **width**: largura da imagem
- **height**: altura da imagem
- ***Caminho Relativo***

- ***Caminho Absoluto***

```
<img src="./image/banner.png" alt="Banner" title="Banner do site"
width="200px" height="200px">
```

OBS: Por boas práticas os dimensionamentos de imagens bem como qualquer formatação devem ser feitos através do CSS. No HTML apenas demarcamos e damos significado semântico ao documento.

LINKS

Links são recursos para criar ligações entre partes de seu documento ou documentos externos:

<A></A>: tags responsáveis por gerar um link. O texto do link fica entre as tags. Ex Google.
Atributos:
HREF: página destino ou da âncora. Ex: *rodapé*. **TARGET**: opcional, utilizado para definir se a página será aberta na mesma aba ou em aba diferente (**_self** ou **_blank**).
NAME: utilizado para definir uma âncora.

```
<a href="#fim" name="inicio">Fim da página</a>
<a href="https://www.google.com/">Ir ao Google</a>

<p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. <br> Est
temporibus ipsa harum earum quis amet <br>
    tenetur
    deleniti totam laboriosam voluptatum doloremque facilis sit <br> nam
adipisci optio quibusdam minus, nulla
    delectus?
</p>
<p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. <br> Est
temporibus ipsa harum earum quis amet <br>
    tenetur
    deleniti totam laboriosam voluptatum doloremque facilis sit <br> nam
adipisci optio quibusdam minus, nulla
    delectus?
</p>
<p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. <br> Est
temporibus ipsa harum earum quis amet <br>
    tenetur
    deleniti totam laboriosam voluptatum doloremque facilis sit <br> nam
adipisci optio quibusdam minus, nulla
    delectus?
</p>
```

```
<p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. <br> Est
temporibus ipsa harum earum quis amet <br>
    tenetur
    deleniti totam laboriosam voluptatum doloremque facilis sit <br> nam
adipisci optio quibusdam minus, nulla
    delectus?
</p>
<p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. <br> Est
temporibus ipsa harum earum quis amet <br>
    tenetur
    deleniti totam laboriosam voluptatum doloremque facilis sit <br> nam
adipisci optio quibusdam minus, nulla
    delectus?
</p>

<a href="#inicio" name="fim">topo</a>
```

TABELAS – Estrutura básica

Trabalha com linhas e colunas, como no Excel.

- **<TABLE></TALBE>** - define uma tabela no seu documento.
- **<CAPTION>** - legenda da tabela.
- **<TR></ TR >** - adiciona uma linha na tabela.
- **< TH ></TH>** - adiciona uma célula de cabeçalho, onde os textos aparecerão em negrito e centralizado
  - **<TD></ TD >** - adiciona uma célula de dado comum - alinhado à esquerda

Semântica em tabelas - semântica tem sempre a ver com o significado do documento, aqui, da tabela.

- <THEAD></ THEAD > - cabeçalho da tabela.
- <TBODY> </TBODY> - corpo da tabela.
- <TFOOT> </ TFOOT> - rodapé da tabela.

Exercício Aplicado

Elabore a estrutura HTML referente à tabela abaixo:
Elabore outro HTML referente à imagem abaixo:

Cabeçalho, corpo e rodapé da tabela

Tabela de preços			
Seminovos	Trompete	Trombone	Trompa
	$500	$640	$650
Visite nossa loja			

Span

SPAN: é um elemento de container de linha. Que não significa nada. Utilizado para inserir qualquer tipo de texto.

****texto qualquer envolvido por um container span****

Div

DIV é um container de bloco genérico. Utilizado para agrupar elementos que poderão ser estilizados através de seus ids ou classes.

** **texto qualquer envolvido por um container span ** **

Formulários

FORM

Tag principal. Envolverá todo o seu formulário. Atributos:

- **action** - informa para onde os dados devem ser enviados após o formulário ser submetido.
- **method** - informa qual o método de envio do formulário - POST/GET

INPUT

Define um campo tipo de entrada de dados de um tipo especificado pelo atributo type.

Atributos TYPE:

text

Define um tipo de entrada de texto

```
<input type="text" />
```

radio

Define um campo de rádio.

```
<input type="radio" name="sexo" value="masculino">Masculino
<input type="radio" name="sexo" value="feminino">Feminino
```

checkbox

Define um ou mais campos para checagem de um ou múltiplos valores.

```
<input type="checkbox" name="hobbies[]">Ouvir música <input
type="checkbox" name="hobbies[]">Jogar video game <input
type="checkbox" name="hobbies[]">Tocar um instrumento
```

select

Define um campo para seleção de uma opção dentre várias possíves.

```
<select id="estados">
      <option value="">Selecione</option>
      <option value="acre">Acre</option>
      <option value="mg">Minas Gerais</option>
      <option value="sp">São Paulo</option>
</select>
```

email

Define um campo específico para emails.

```
<input type="email" required id="email">
```

fieldset

Define uma marcação para envolver um grupo de campos de um formulário.

```
<fieldset>
  <legend>Dados Pessoais</legend>
  <label for="nome">Nome</label>
  <input type="text" id="nome" />
  <label for="email">E-mail</label>
  <input type="email" id="email" />
</fieldset>
```

button

define um botão comum, de reset ou de submit

```
<button type="button">Botão Comun</button>
<button type="reset">Limpar Tudo</button>
<button type="submit">Enviar Dados</button>
```

submit

Define um botão type submit. O tipo submit é o que realmente envia o formulário.

```
<input type="submit" id="enviar" value="Enviar Dados" />
```

number(min, max, step, value)

Define um input

range

Define um botão de range

```
<input type="range" name="points" min="0" max="10">
```

Atributos genéricos:

placeholder: dica do campo

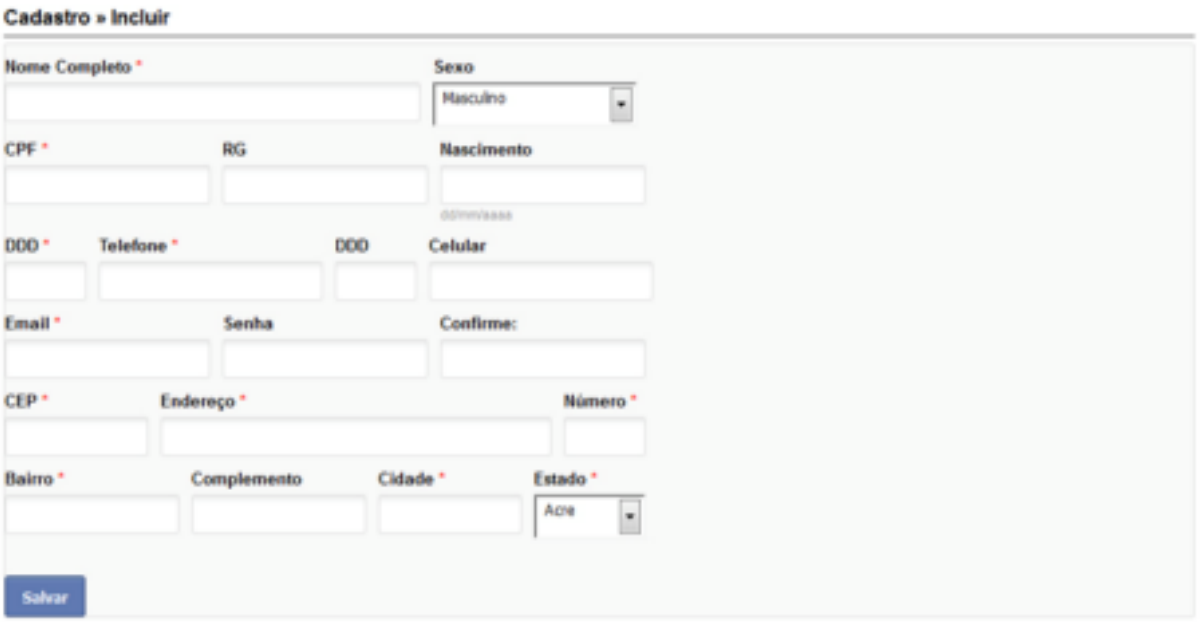
required: campo obrigatório

autofocus: foco automático do campo

disabled: desabilita o campo para edição

Exercício de aplicação 1

Elabore uma marcação HTML para o formulário da imagem abaixo:



Exercício de aplicação 2

Elabore uma marcação HTML para formar um formulário conforme dados abaixo:

DADOS PESSOAIS:

- Nome Completo (txt)
- Sexo (radio)
- Nacionalidade
- CPF
- Nascimento
- Estado Civil (radio)

DADOS PARA CONTATO:

- CEP (text)
- Logradouro (select)
- Endereço (text)
- Número (number)
- Cidade (text)
- Estado (select)
- Email (email)
- Celular(number ou text)
- Fixo(number ou text

Semântica em documentos HTML

Introdução:

A partir da versão 5 do HTML foram introduzidas tags de significado do documento. A semântica tem a ver com o significado que você quer dar às partes de seu documento como veremos nesta seção.

Antigamente as divisões dos sites eram feitas por divs identificadas por ids ou classes. Porém isso dificultava o trabalho dos buscadores e leitores de tela. Sendo assim a versão 5 do HTML trouxe tags específicas para melhor identificar os documentos e dar sentido, significado para o seu documento HTML.

Principais tags semânticas:

HEADER

Utilizada para identificar a cabeça do site.. O próprio cabeçalho. Não confunda com HEAD do documento!!

```
<header>
   <h1>Food's Truck</h1>

   <nav>
       <a href="/Home">Home</a>
```

```
        <a href="/quemsomos">Quem Somos</a>
        <a href="/contato">contato</a>
    </nav>
</header>
```

NAV

Utilizada para referenciar grupos de links importantes como os menus.

```
<nav>
        <a href="/Home">Home</a>
        <a href="/quemsomos">Quem Somos</a>
        <a href="/contato">contato</a>
</nav>
```

MAIN

Utilizada para englobar o conteúdo principal do site.. Seria aqui o corpo do seu documento.

```
<main>
<!-- conteúdo do site -->
</main>
```

SECTION

Utilizada para definir seções genéricas do documento. Define uma seção ou um grupo de um determinado assunto. Agrupa elementos que possui relação entre si, como uma sessão de notícias, por exemplo esportes trânsito, notícias de última hora.

```
<section>
  <h2>Cabeçalho</h2>
  <img src="carneiro.jpg" alt="carneiro">
</section>
```

ARTICLE

Utilizada para agregar conteúdos como artigos, textos redações e que se repetem, etc.

```
<article id="site">
    <h3>CRIAÇÃO DE SITES</h3>
    <img src="./image/icone1.png" alt="criação de sites">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Veniam,
voluptatum labore! Officiis
        exercitationem commodi ex rem necessitatibus at, maxime eos
tempora quas labore cumque! Ipsa
        debitis deleniti quis qui aspernatur.</p>
</article>
```

ASIDE

Utilizada para referenciar uma coluna que envolve geralmente sidebars, barra laterais como anúncios ou links laterais ou conteúdo ligeiramente correlacionados com o conteúdo principal. Pode ser um glossário, links de outros blogs, etc.

```
<aside>
  <p>Conteúdo relacionado ao conteúdo principal do site</p>
</aside>
```

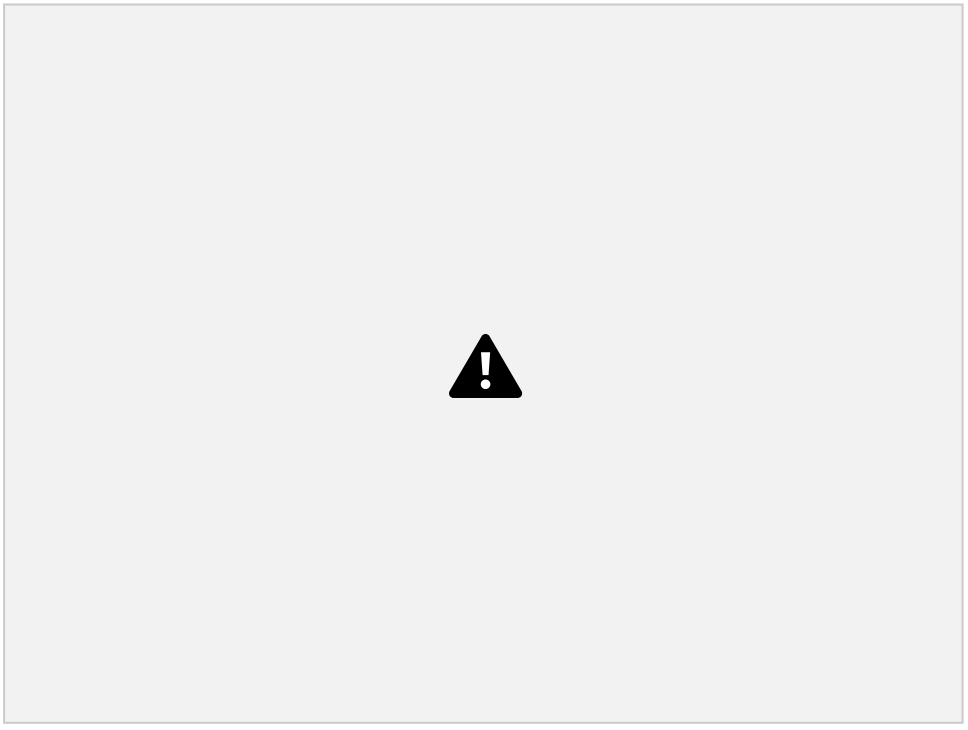
FOOTER

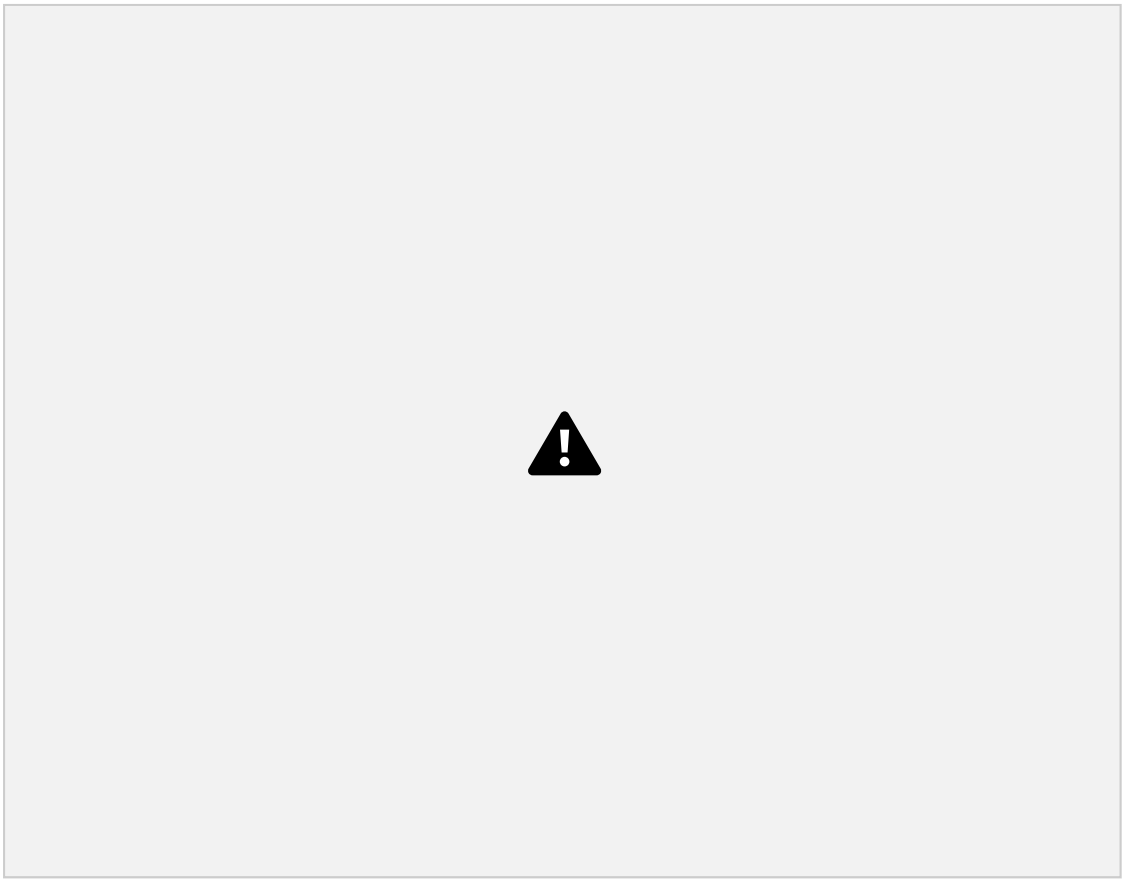
Utilizamos para identificar o rodapé do site.

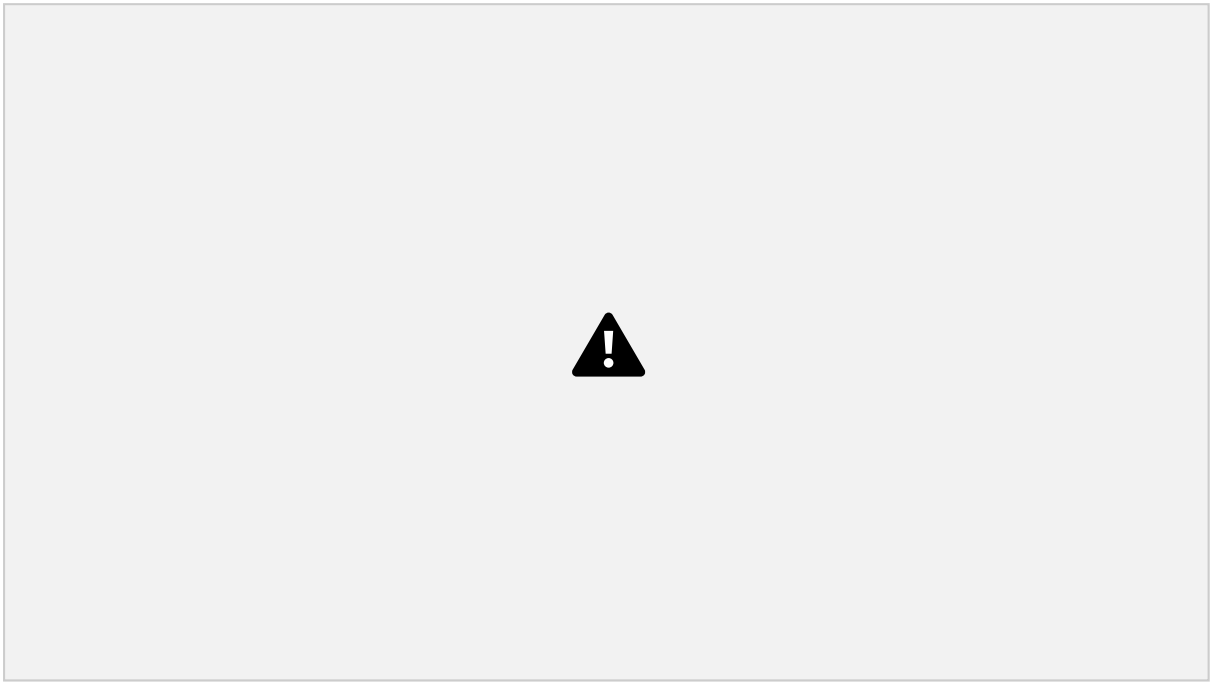
```
<footer><p>Todos os direitos reservados</p></footer>
```

OBS: As DIVS podem ser utilizadas, porém quando você não precisa de um significado, como por exemplo uma div para armazenar conteúdos sem significados.

Exemplos de estruturas semânticas

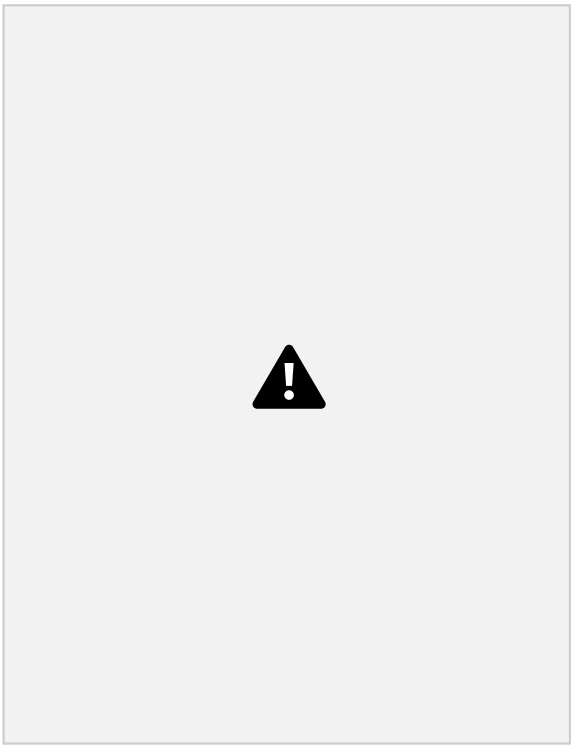






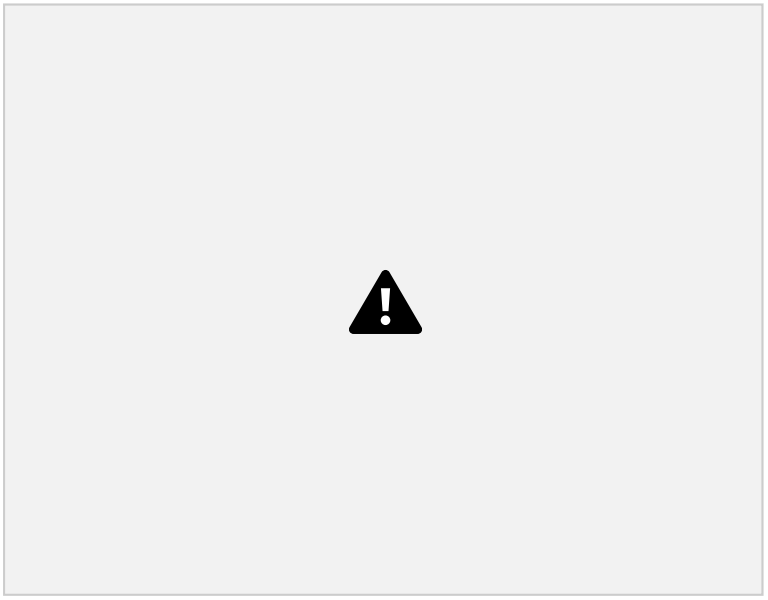
Exercício 01

Agora olhe para a imagem abaixo. Identifique as partes semânticas e elabore o HTML que você identificou:



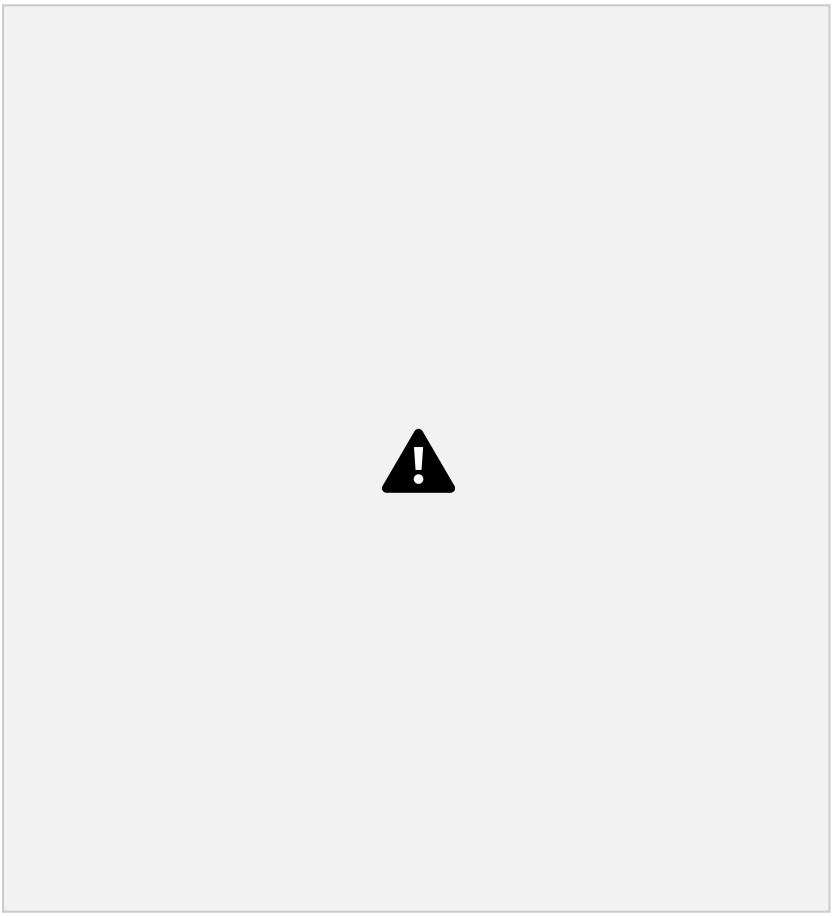
Exercício 02

Faça o HTML para representar o layout abaixo:



Exercício 03

Para finalizar, um último desafio:



CSS - Introdução

CSS é a sigla para Cascading Style Sheets que significa Folhas de Estilos em Cascata. São formadas por regras, geralmente em arquivos separados e servem para aplicar estilos em documentos HTML puros como já pudemos ver anteriormente.

Existem três formas de inserir CSS em seu documento HTML:

● INLINE: Utilizado através do atributo STYLE da própria TAG a ser estilizada. ● INCORPORADO: As regras são escrita entre as TAGs <STYLE> </STYLE> ● EXTERNO: Regras escritas em arquivo externo com a extensão .css e chamado através da TAG <LINK> - <link rel="stylesheet" href="caminho_do_arquivo">

Regras

Para aplicar um estilo em um determinado elemento do documento HTML temos as regras responsáveis por esse fim.

Regra: seletor, declaração e propriedade no seguinte formato:

```
seletor{ propriedade: valor; }
```

Exemplo:

```
p { color : red; }
```

Seletores

Temos os seguintes seletores básicos em CSS:

Seletor universal;

Utilizado para aplicar regras CSS em todos os elementos do documento HTML. Geralmente utilizado para resetar margin e padding dos elementos como veremos na prática no decorrer do curso.

```
* {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }
```

Seletor de tag ou tipo

Aplica uma regra a todos os elementos do html daquela tag.

```
p {color: red; font-family: Verdana Sans-Serif;}
```

Seletor de atributo

Utilizado para aplicar uma regra a todos os elementos que contenham determinado atributo.

```
input[type="number"]
{
      background-color: grey;
color: white;
font-size: 0.8rem;
}
```

Seletor de id

```
#foo {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
            font-size: 16px;
            color: grey;
}
```

Seletor de class

```
.sessao-principal {
      display: flex;
      flex-direction: row-reverse;
}
```

Comentários

Os comentários servem para colocar lembretes, documentar trechos de regras ou simplesmente para esconder do navegador a interpretação de alguma regra ou trecho de regra específico:

```
H2 {color: blue;}
/* P {font-size: 15px} */
```

No trecho acima somente o elemento h2 será estilizado. Já o parágrafo não será interpretado pelo navegador, pois está comentado.

Dimensões

width

Propriedade para dimensionar a largura dos elementos estilizados. Por padrão os elementos recebem width com tamanho de 100%, dentro do elemento pai.

```
div {width: 500px;} ou div {width: 70%;}
```

height

propriedade para dimensionar a altura dos elementos estilizados. Por padrão segue-se o tamanho do conteúdo;

```
div {height: 100px}
```

max-width

define a largura máxima do elemento

min-width

define a largura mínima do elemento

max-height

define a altura máxima do elemento

min-height

define a altura mínima do elemento

```
.container {
      width: 100%;
      max-width : 1200px;
      margin: 0 auto;
}
```

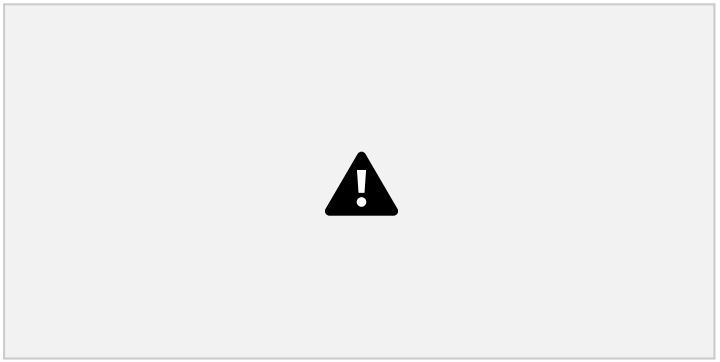
Margin e Padding

Margins e paddings são utilizados para definir uma margem para o elemento em relação aos elementos externos (margin) ou elementos internos (padding), veja:

```
header {
   margin-top: 20px;
   margin-right: 20px;
   margin-botton: 20px;
   margin-left: 20px;
}
```

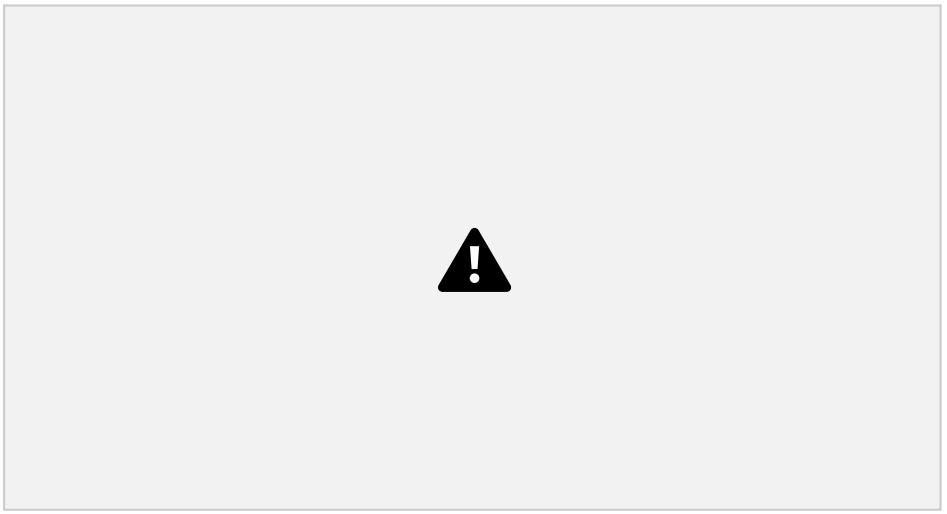
```
footer {
   padding-top: 20px;
   padding-right: 20px;
   padding-botton: 20px;
   padding-left: 20px;
}
```

Margin e padding são somados ao tamanho (width) do elemento



Border

Seguindo o mesmo conceito de de margin e padding, BORDER também está relacionado ao elemento e segue as 4 posições sentido horário;

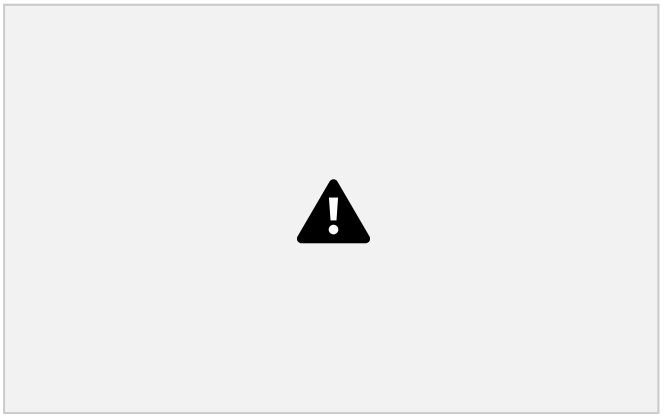


```
div {
    margin-top: 20px;
    margin-right: 20px;
    margin-botton: 20px;
    margin-left: 20px;
}
```

Box Model

Todos os elementos html podem ser considerados como caixas que contém margin, padding, borda e o conteúdo interno.

O conceito box model está relacionado ao layout. Falamos de box model para pensar na estrutura, disposição dos elementos na página.

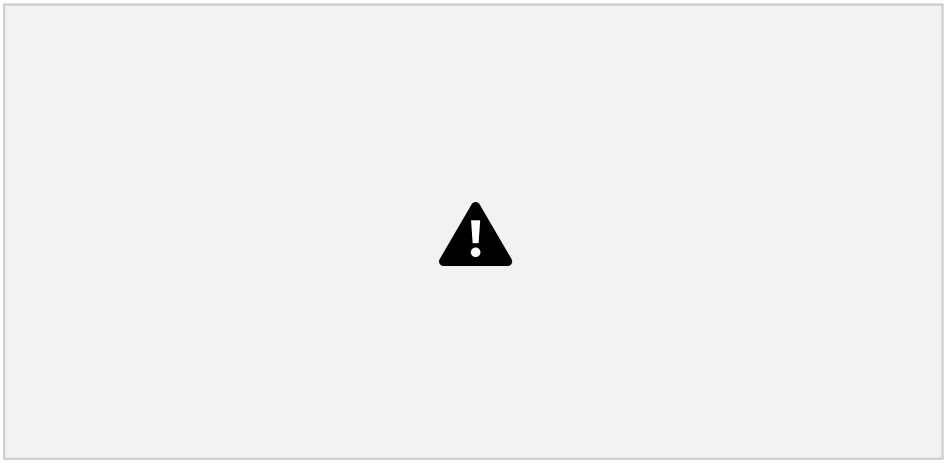


Box sizing

A propriedade de dimensionamento de caixa (box-sizing) é usada para informar ao navegador que as propriedades de dimensionamento (width e height) devem ser incluídas, de acordo com o **valor** da propriedade:

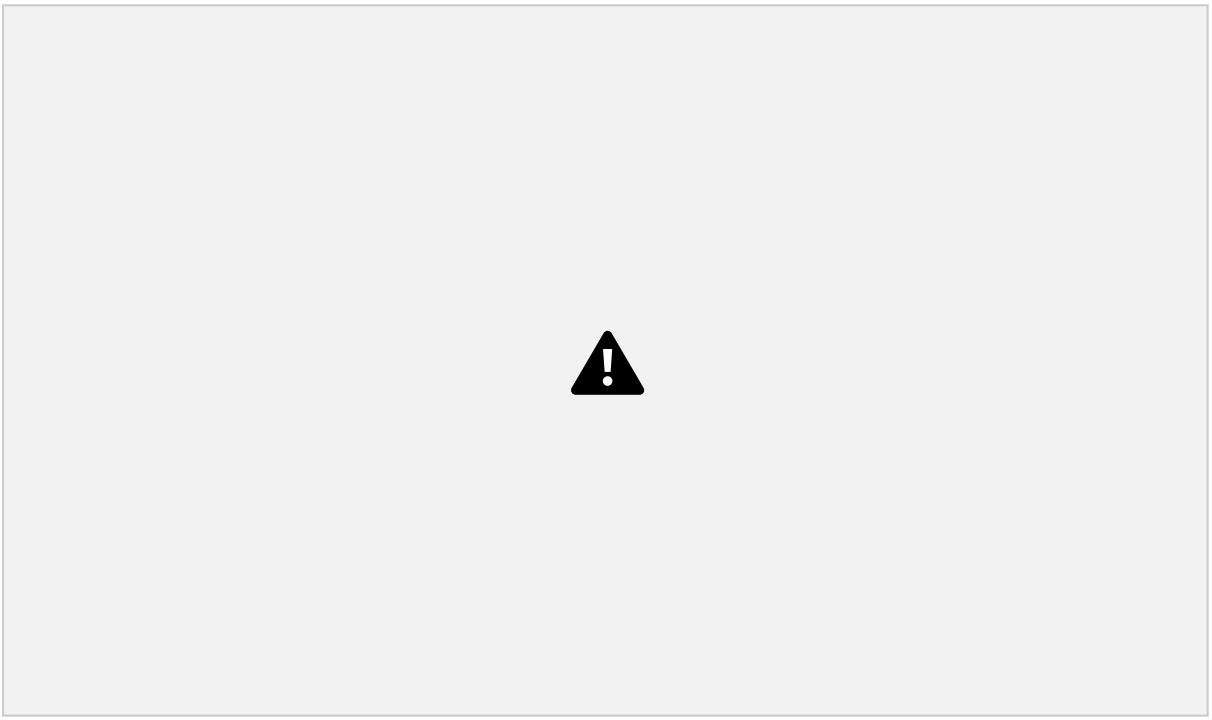
Valores:

- content-box (*default*) – width e height só são calculados considerando apenas o conteúdo da caixa;
- **border-box** – width e height incluem o paddin e a borda para o seu cálculo, ● padding-box - width e height incluem o tamanho padding size, mas não incluem o tamanho da borda ou margem



Unidades de Medida

As unidades de medida em CSS servem para medir o comprimento de um elemento.
Algumas surgiram dos estudos de tipografia como pt (ponto) e pc (paica).
Existem diversas unidades de medidas em CSS. Aqui estudaremos durante nossos projetos as principais para o desenvolvimento web como px, %, em, rem, vh, vw, vmin, vmax..



Fonte: (https://www.w3.org/Style/Examples/007/units.pt_BR.html)

px

O Pixel é uma unidade de medida fixa. Ela só existe para monitores de computadores e equivale a 1 ponto na tela. Se você definir a largura de um elemento como 200px teremos uma largura de 200 pontos na tela.

Ex:

```
div { width: 200px; background-color:purple; }
```

em

A unidade relativa EM tem seu padrão de 16px em todos os navegadores, quando não especificada. Você pode fazer uma comparação da seguinte forma:

Imagine que você gostou da fonte em tamanho 40px. Você pode calcular 40/16 = 2,5. Ou seja pode usar 2.5em.

```
.meu-texto { font-size : 2.5em }
```

As fontes são herdáveis através do seu elemento pai e dessa forma deve-se tomar o seguinte cuidado: Imagine que você não definiu um tamanho de fonte para o elemento root - o body. Então repare no exemplo abaixo:

HTML

```
<h1>Utilizando a medida relativa EM e <i>tes<strong>tan</strong>o</i>
```

```
geral</h1>
```

CSS

```
h1 {
    font-size:2.2em;
}

h1 i {
    font-size:2.2em;
}

h1 i strong {
    font-size:2.2em;
}
```

Resultado:

rem

Diferente da unidade de medida *em* a unidade **rem** utiliza como o tamanho da fonte padrão exibida no elemento root, o body. Nesse caso não sofreremos do efeito cascata como na unidade *em*.

HTML

```
<h1>Texto com o tamanho padrão de 10px para
<i>tes<strong>tes</strong></i> em geral</h1>
```

CSS

```
body {
    font-size: 10px;
}
h1 {

    font-size:1.5rem; /* equivale a 15px, ou seja, 10px * 1.5 */ }

h1 i {
    color:red;
    font-size:2rem; /* equivale a 20px, ou seja, 10px * 2 */ }

h1 i strong {
    color:purple;
    font-size:3rem; /* equivale a 30px, ou seja, 10px * 3 */ }
```

Resultado:

vh

A unidade vh, viewport height, toma como base a altura útil disponível do dispositivo, independente da altura do elemento pai.

Em vh e vw ambas equivalem a 1/100 (como em porcentagem, 1%) da altura ou largura da viewport. Ambas são medidas responsivas e relativas.

HTML

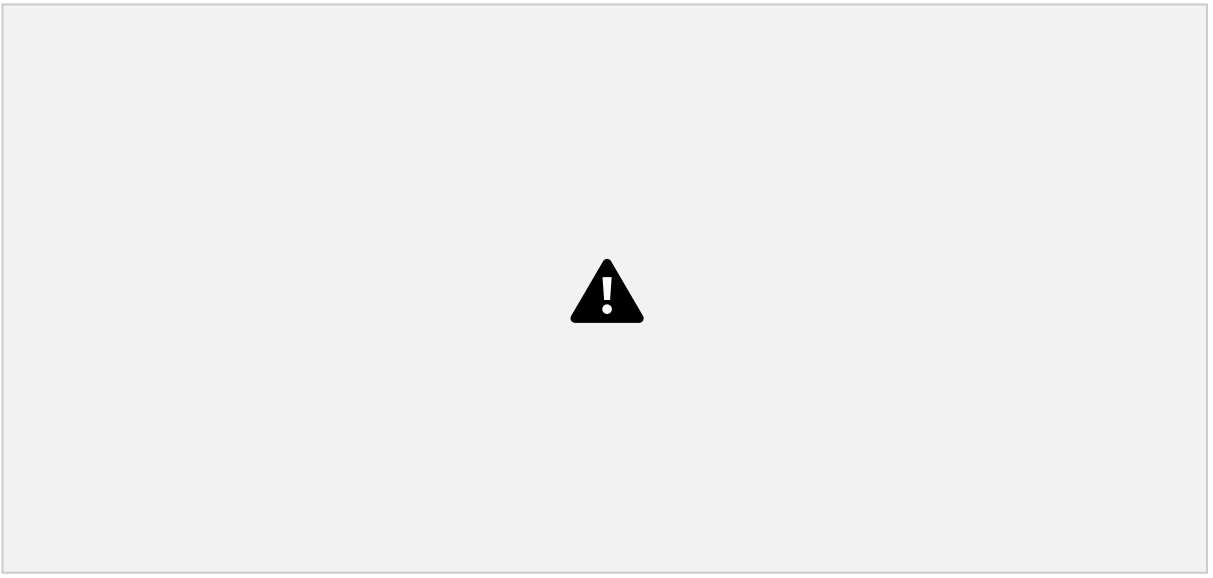
```
<body>
<header>
    <h1>CABEÇALHO DO DOCUMENTO - 20% da altura da viewport</h1>
</header>

<main>
    <h2>CORPO DO DOCUMENTO - 60% da altura da viewport</h2>
</main>

<footer><h2>RODAPÉ DO DOCUMENTO - 20% da altura da
viewport</h2></footer>
</body>
```

CSS

```
* {
    margin:0;
    padding:0;
    box-sizing: border-box;
}

header {
    background-color: black;
    height: 20vh;
    color: white;
}

main{
    background-color: purple;
    height: 60vh;
    color: white;
}

footer {
    background-color: black;
    height: 20vh;
    color: white;
```

```
}
```

Resultado:



VW

Da mesma forma que a unidade de medida vh, temos a unidade vw, viewport width, que leva em consideração a largura da viewport do elemento sem se importar com a altura do elemento pai.

HTML

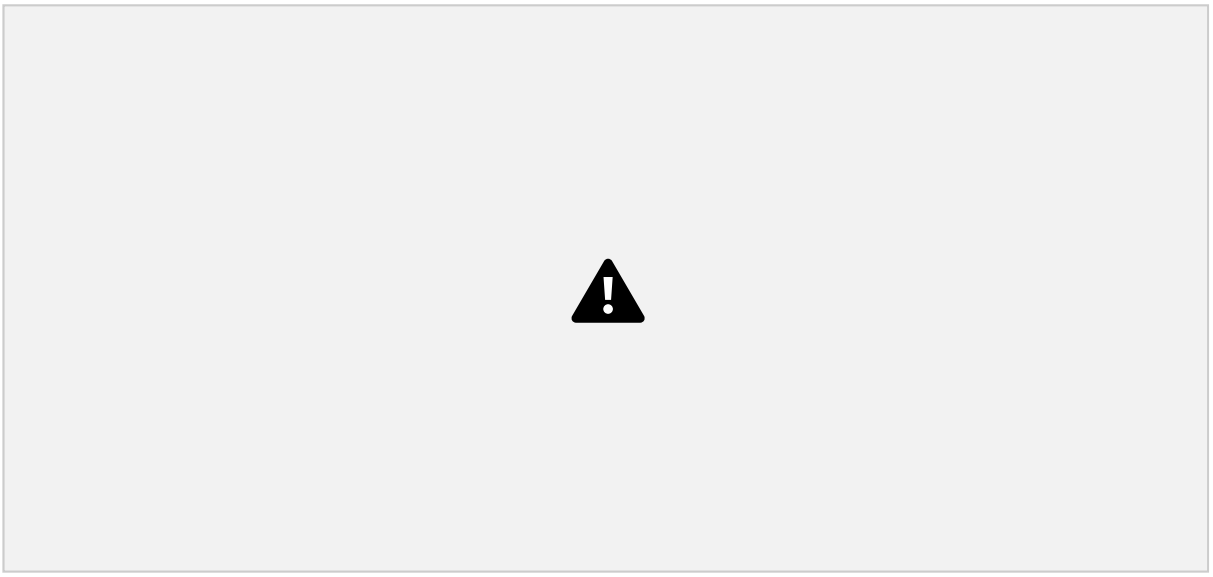
```
<header>
    <h1>CABEÇALHO DO DOCUMENTO - 20% da altura da viewport</h1>
</header>

<main>
    <h2>CORPO DO DOCUMENTO - 60% da altura da viewport</h2>
</main>

<footer><h2>RODAPÉ DO DOCUMENTO - 20% da altura da
viewport</h2></footer>
```

CSS

```
* {
    margin:0;
    padding:0;
    box-sizing: border-box;
}

header {
    background-color: black;
```

```
    width: 50vw;
    height: 20vh;
    margin: 0 auto;
    color: white;
}

main{
    background-color: purple;
    width: 50vw;
    height: 60vh;
    margin: 0 auto;
    color: white;
}

footer {
    background-color: black;
    width: 50vw;
    height: 20vh;
    margin: 0 auto;
    color: white;
}
```

Resultado:



%

A unidade % equivale a medida em percentual. Diferentemente de vh e vw que tomam como base a viewport, a unidade em porcentagem tem como referência o tamanho da do elemento pai.

HTM

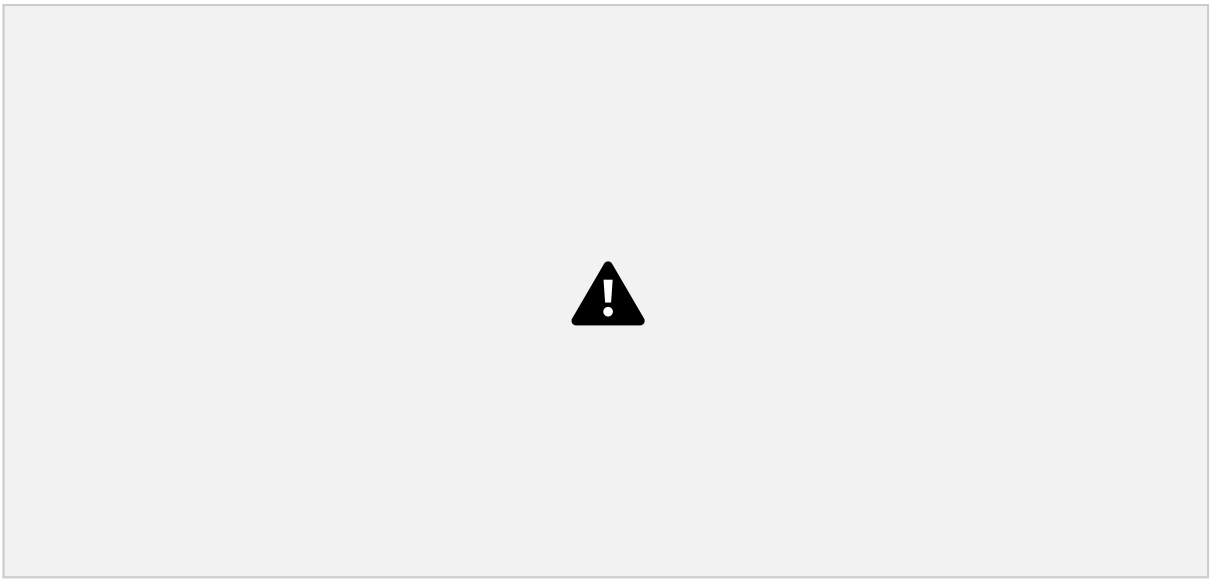
```
<header>
    <h1>CABEÇALHO DO DOCUMENTO</h1>
</header>
```

CSS

```
* {
    margin:0;
    padding:0;
    box-sizing: border-box;
}
body {
    background-color: bisque;
}
header {
    width: 800px;
    height: 300px;
    background-color: black;
    color: white;
}

h1 {
    width: 90%;
    height: 30%;
    background-color: transparent;
    color:orangered;
```

```
    border: 1px solid orangered;
}
```

Resultado:



vmin

A unidade vmin utiliza como base o menor tamanho da viewport. imaginemos que você tem uma viewport ajustada em 320px x 568px, como exemplo um iphone 5s. nesse caso o menor tamanho seria 320 e 1vmin, nesse caso, seria 3.2px (). 1vmin significa 1/100 do tamanho, como em porcentagem, porém o menor tamanho da viewport.

Em um outro exemplo, imaginemos uma viewport ajustada para 1200px x 768px. O menor tamanho, neste caso, é 768. Dessa forma 1vmin seria 76.8px.

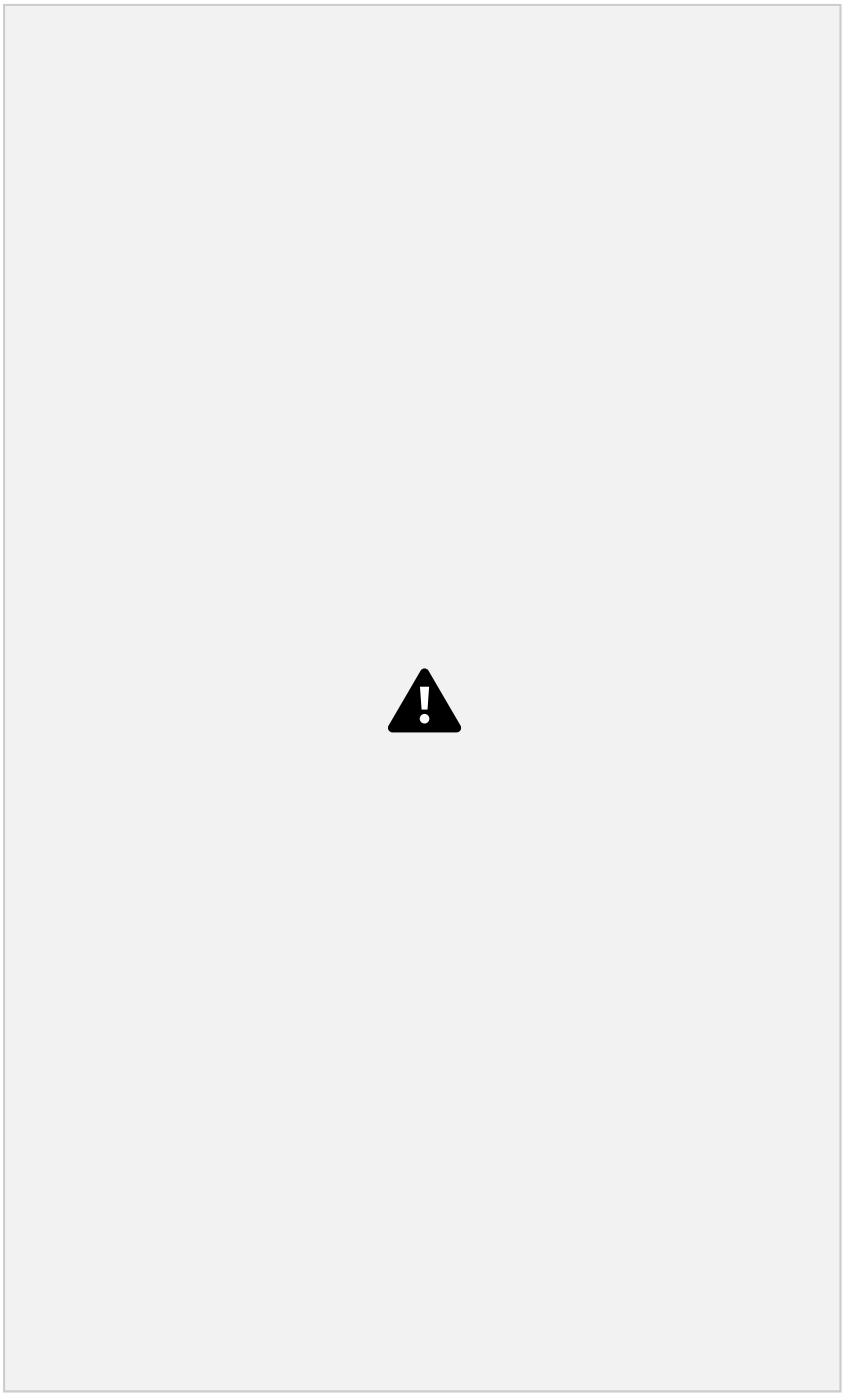
HTML

```
<body>
    <header></header>
</body>
```

CSS

```css
* {
    margin:0;
    padding:0;
    box-sizing: border-box;
}

body {
    background-color: bisque;
}

header {
    width: 100vmin;
    height: 100vmin;
    background-color: purple;
    color: white;
}
```

Resultado:



vmax

Em vmax, temos o mesmo funcionamento de vmin, porém é levado em consideração o maior tamanho da viewport.
Uma viewport redimensionada em 320px x 568px teremos 1vmax equivalente a 56.8px.

Propriedades de texto

Podemos estilizar os textos de nosso documento de diversas formas. Veremos a seguir as principais propriedades.

color

Define cor do texto: ex `color: grey;`

text-align

Define o alinhamento do texto com os possíveis valores: text-align: center;

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

text-decoration

Define a decoração do texto. Podemos alterar, por exemplo um link, retirando o seu underscore.

```
a { text-decoration: none; }
```

text-transform

Utilizamos para formatar um texto com letras maiúsculas, minúsculas ou capitalizado.

```
input[type="text"] { text-transform: uppercase; }
```

text-indent

Define a indentação ou recuo da primeira linha.

```
p {text-indent: 10px;} /* primeira linha com deslocamento de 10px */
```

line-height

Define a altura da linha, independente do tamanho da fonte.

```
p { font-size: 10px; line-height: 15px } /*tamanho da fonte 10px, porém tamanho da linha com 15px;*/
```

letter-spacing

Altera o espaçamento padrão entre as letras.

```
p { letter-spacing: 2px; }
```

word-spacing

Altera o espaçamento padrão entre as palavras.

```
p { word-spacing: 5px; }
```

text-shadow

Define uma sombra para o texto a seguinte ordem: vertical horizontal **desfoque** cor;

```
p { text-shadow: 3px 3px 4px red; }
```

Overflow

A propriedade overflow controla o conteúdo dentro do container. Seja esse container um parágrafo, div, header ou qualquer outro box de um elemento, como vimos em **box-model**.

Se uma div tiver um tamanho limitado, por exemplo 200px x 200px, e o seu conteúdo for maior que ela, você pode controlar se a div terá barra de rolagem ou não.

```
p {
      width: 200px;
height: 200px;
overflow: auto;/* quando o conteúdo ultrapassar o limite do parágrafo,
automaticamente aparecerá uma scrolbar*/
}
```

Para maiores informações acesse: https://www.w3schools.com/cssref/pr_pos_overflow.asp

Opacidade

A opacidade ou nível de transparência de um elemento pode ser controlada pela propriedade opacity.

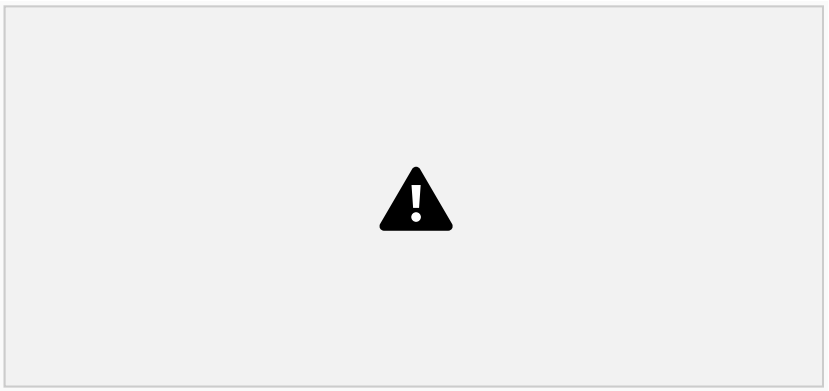
CSS

```
body {
    background-color: bisque;
    background-color: orange;
    font-size: 3em;
}
```

```css
.opaco {
    width: 800px;
    height: 200px;
    opacity: 0.5; /* 50% de opacidade */
    margin: auto auto;
    text-align: center;
    background-color: cyan;
    border: 1px dashed red;
}
```

HTML

```html
<body>
    <div class="opaco">div com nível de opacidade alterado</div>
</body>
```



DISPLAY

A propriedade display é utilizada para controlar como é feita a renderização da caixa do elemento HTML. É aplicada a todos os elementos e a maioria deles, por padrão é block. Os principais valores para a propriedade display são: none, inline, block, inline-block e flex.

flex

Renderiza a caixa do elemento de forma flexível e pode ser conjugada com outras propriedades que veremos quando estudarmos Flexbox.

none

O valor none, omite a renderização da caixa do elemento na tela.

inline

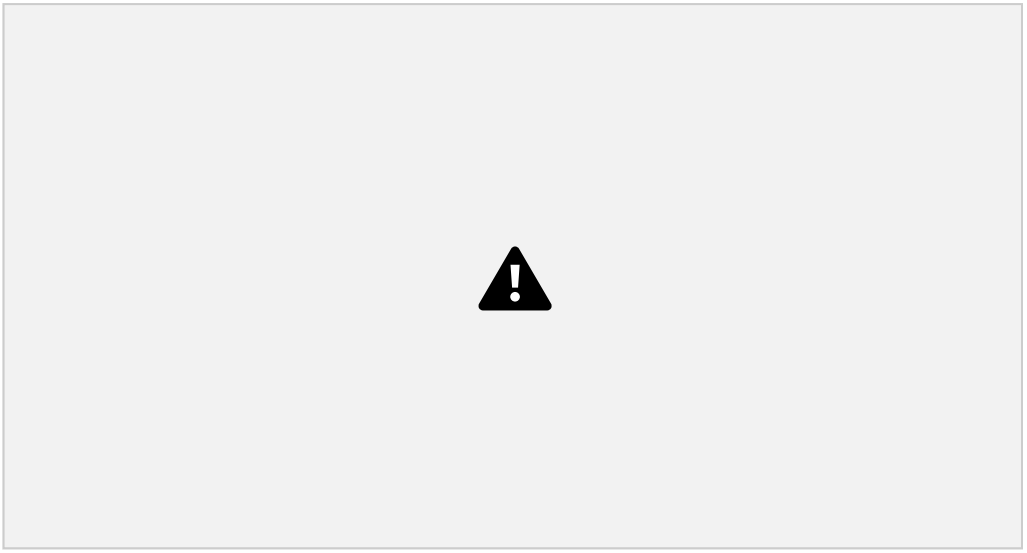
Renderiza a caixa do elemento de em volta do mesmo, na mesma linha, e sendo assim podemos ter vários elementos inline, ou seja, na mesma linha, um ao lado do outro. Esta propriedade não aceita redimensionamento da caixa e margin's.

block

Block já renderiza a caixa do elemento em 100% da largura da linha, mesmo que o conteúdo não ocupe todo este espaço. Outra característica marcante é que mesmo aceitando redimensionamento da caixa, um elemento com display block não aceitará outro elemento ao seu lado, fazendo com que o próximo elemento ou o elemento anterior fique nas linhas de baixo ou de cima respectivamente.

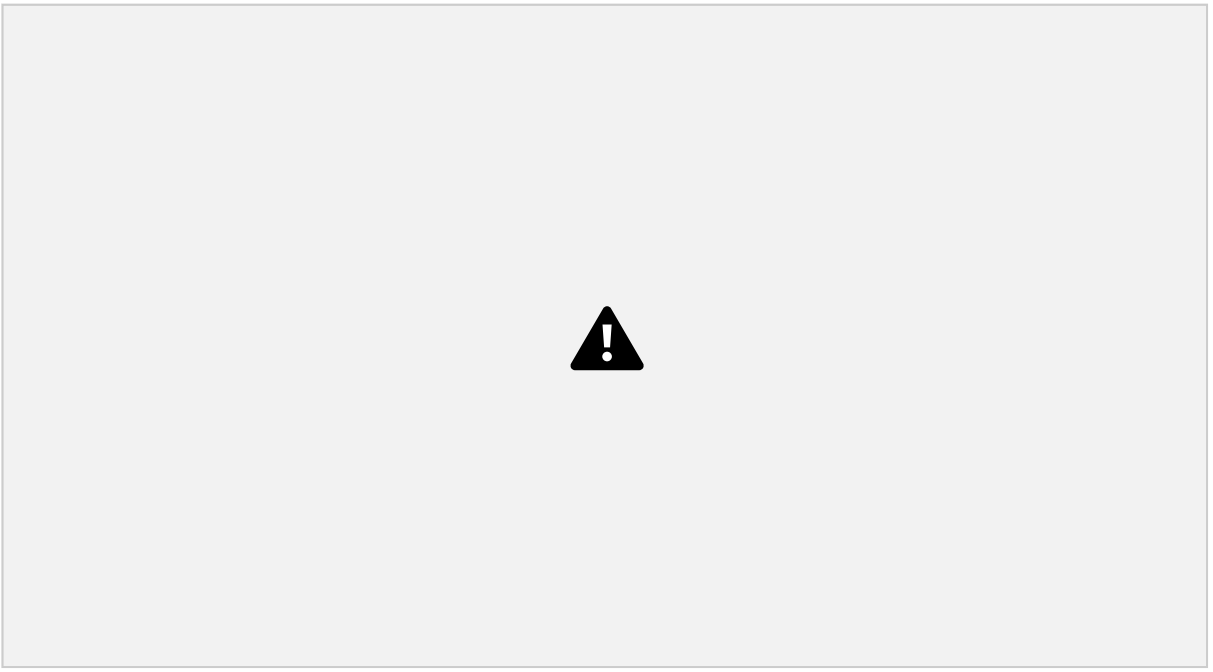
inline-block

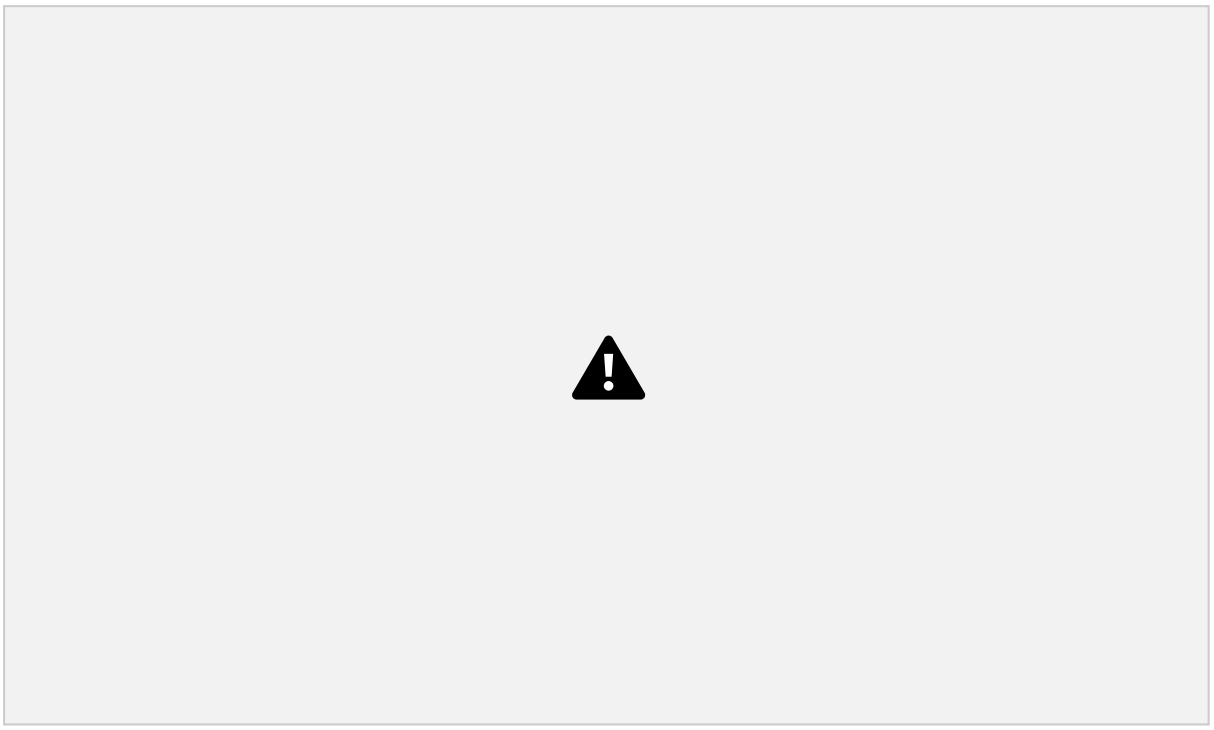
O valor inline-block mescla as características de inline e block. Um elemento inline-block aceita o redimensionamento e admite que outros elementos fiquem ao seu lado, caso este outro elemento caiba no espaço da linha.

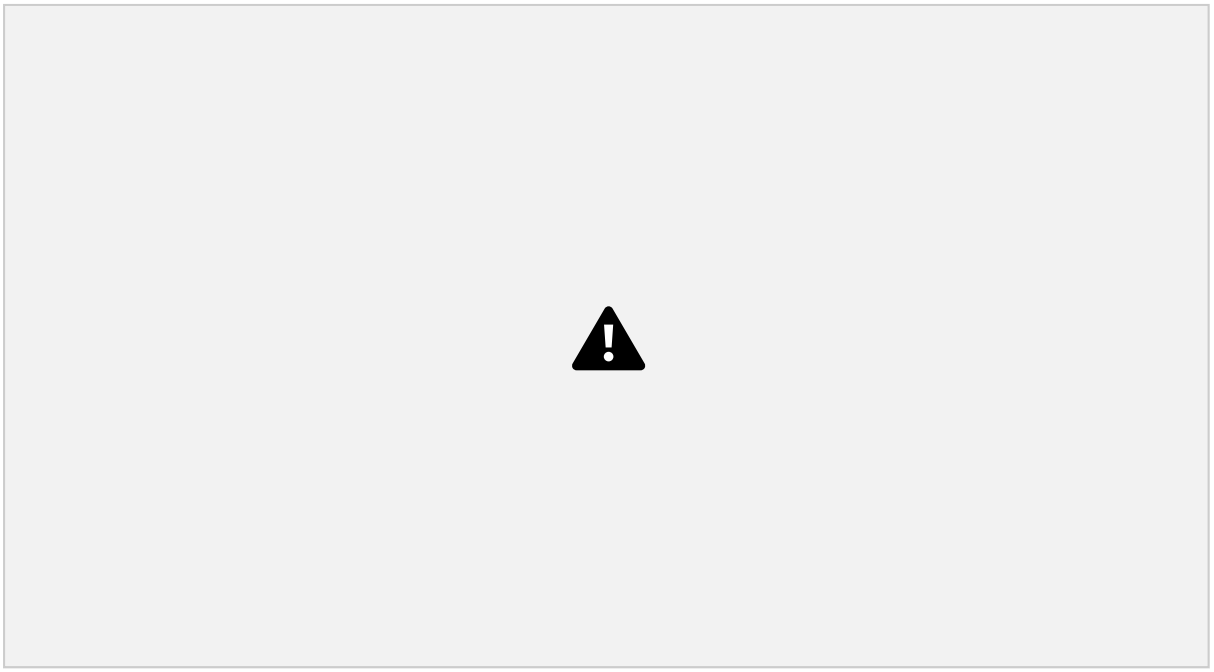


Em especial vamos trabalhar com conceito de box model e inline block em um exercício prático para validar sua aplicação em desenvolvimento de layouts. Porém hoje em dia o mais indicado é trabalhar com flexbox. Mas há situações onde o cliente precisa de manutenção em projetos legados onde conterá layouts feitos com float ou inline-bolck.

Vamos desenvolver os seguintes layouts:







Position

Controla o posicionamento dos elementos na tela. contém as propriedades de posicionamento top left right e bottom para se posicionar na tela, sempre em relação a algum outro elemento pai ou a si mesmo. Por padrão o position dos elementos é *static* e, esta não permite alterar as propriedades de posicionamento ditas acima como veremos. Os principais valores são:

static

A propriedade static é a propriedade padrão dos elementos. Faz com que o posicionamento deste elemento seja padrão, no fluxo dos elementos da página. Não possibilita alterações das propriedades top, left, right ou bottom.

relative

A propriedade relative toma como base para posicionamento o seu próprio box, porém seu espaço continua reservado após as alterações das coordenadas de posicionamentos. O box com este valor de position faz com que seu posicionamento seja inicialmente padrão, no fluxo dos elementos da página. Veremos mais detalhes em sala de aula.

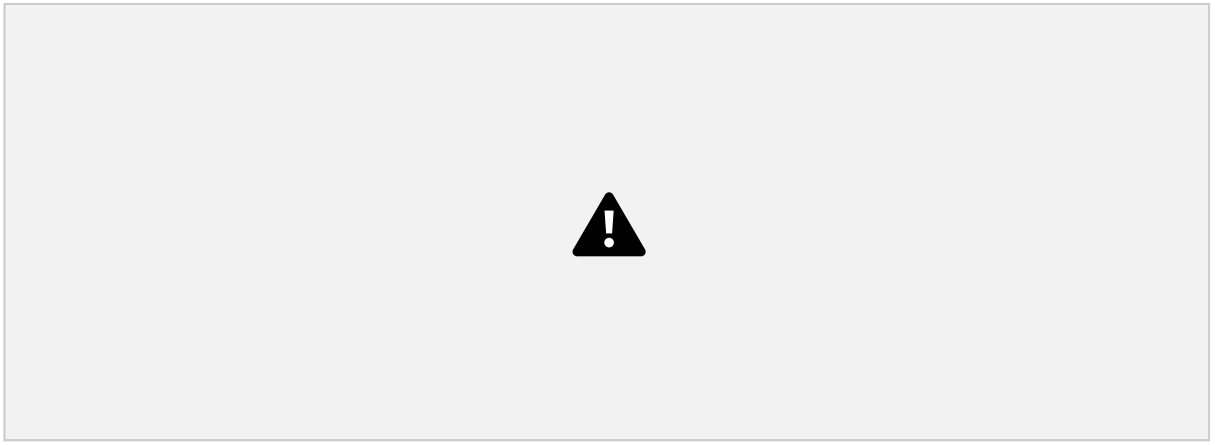
absolute

Position com o valor absoluto toma como base o elemento ancestral mais próximo que contenha algum position explicitamente definido, desde que não seja static :) outra característica marcante é que o conteúdo é retirado do fluxo padrão dos elementos, como se estivesse uma camada acima. Veja a imagem na próxima página.

fixed

Position com o valor fixed funciona parecido com o absoluto. Seu posicionamento tem como

referência a janela do browser, ou seja, a viewport, sempre, o que torna seu box fixo na tela como podemos ver em sala.



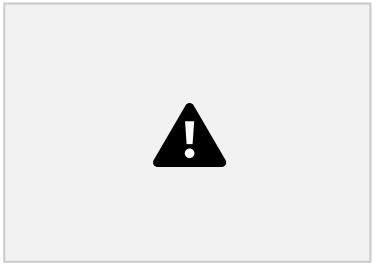
z-index

Controla a posição da camada do elemento. A cada elemento criado é gerado um número de camada (z-index), a partir do zero, automaticamente. Desta forma o valor padrão de position é auto.

Repare no exemplo abaixo:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

#verde, #amarelo {
    width: 300px;
    height: 300px;
    position: fixed;
}

#verde {
    margin-left: 20px;
    background-color: #144556;
}

#amarelo {
    background-color: #dcd754;
}
<body>
<div id="verde"></div>
<div id="amarelo"></div>
</body>
```

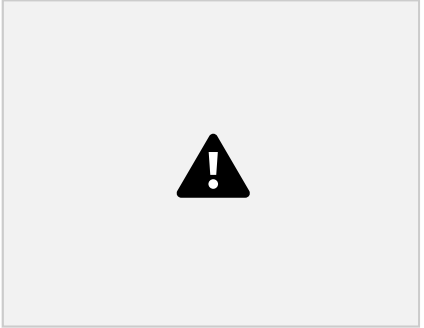
Ambos os box receberam position fixed e seu
top e left



automaticamente estão com o valor 0. Desta forma o box amarelo ficou por cima, pois ele foi criado por último e, como z-index não foi definido, seu valor ficou **1**, enquanto o z-index do box verde ficou 0 por ser o primeiro a ser criado no documento.

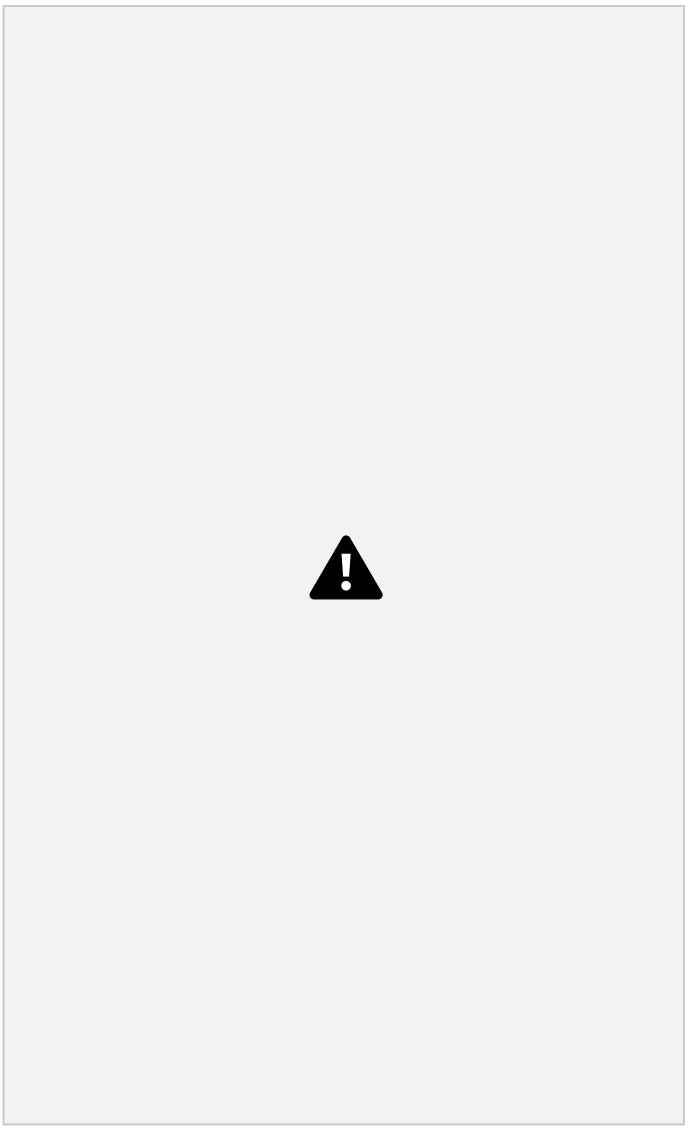
Veja outro exemplo, com a inversão na ordem das camadas:

```
#verde {
    margin-left: 20px;
    background-color: #144556;
    z-index: 1;
}

#amarelo {
    background-color: #dcd754;
    z-index: 0;
}
```



Agora o box verde ficou por cima, mesmo sendo criado antes do box amarelo. Com z-index podemos controlar a camada do elemento como bem quisermos.

Vejamos um exemplo prático:



```
<!DOCTYPE html>
<html lang="pt-br">

<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.5.0/css/all.css">

    <style>
        @import
url('https://fonts.googleapis.com/css?family=Roboto+Condensed');
        @import
url('https://fonts.googleapis.com/css?family=Montserrat');
        /*************** mobiel first ********************/
        * {
            margin: 0;
            padding: 0;
```

```
    box-sizing: border-box;
}

body {
    position: relative;
    width: 100vw;
    height: 100vh;
    color: white;
    font-size: 16px;
    font-family: 'Roboto Condensed', sans-serif;

    background: linear-gradient(to bottom, #144556, #144556);
}

h1 {
    position: absolute;
    top: 20%;
    width: 80%;
    margin-left: 10%;
    font-size: 6rem;
    text-align: center;
}

form { color: #144556; }

form input { color: #144556;}

#login {
    position: absolute;
    width: 80%;
    margin-left: 10%;
    height: 8vh;
    top: 35%;
}

#login i {
    position: absolute;
    top: 1.4vh;
    left: 25px;
    z-index: 2;
    color: #144556;
    font-size: 5rem;
}

#login input {
    position: absolute;
    width: 100%;
    font-size: 4rem;
    height: 8vh;
    z-index: 1;
    padding-left: 10rem;
```

```
    padding-right: 5rem;
    border-radius: 15px;
}

 #password {
    position: absolute;
    top: 47%;
    width: 80%;
    height: 8vh;
    font-size: 6rem;
    z-index: 1;
    margin-left: 10%;
    padding-left: 10rem;
    padding-right: 5rem;
    border-radius: 15px;
}

#enter {
    position: absolute;
    top: 60%;
    width: 80%;
    height: 8vh;
    font-size: 6rem;
    z-index: 1;
    margin-left: 10%;
    padding-left: 7rem;
    padding-right: 5rem;
    font-size: 4rem;
    background-color: #75c3b5;
    color: white;
    border: none;
    border-radius: 15px;
}
#forgot {
    position: absolute;
    top: 72%;
    width: 80%;
    height: 8vh;
    font-size: 3.5rem;
    z-index: 1;
    margin-left: 10%;
    padding-left: 7rem;
    padding-right: 5rem;
    font-family: 'Montserrat', sans-serif;
    font-size: 3rem;
    background-color: transparent;
    color: white;
    border: 5px dashed white;
    border-radius: 15px;
}
```

```
    </style>
</head>

<body>

    <h1>LOGIN</h1>
    <form action="">
        <div id="login">
            <i class="fas fa-user"></i>
            <input type="email" id="email" placeholder="your name">
        </div>

        <input type="password" id="password" placeholder="********">

        <button type="submit" id="enter" id="">ENTER</button>
        <button type="button" id="forgot" id="">FORGOT PASSWORD</button>
    </form>
</body>
</html>
```

Poderíamos fazer este exemplo de várias formas diferentes porém resolvi fazer com position e controle de camadas para deixar um exemplo completo para discutirmos em aula.

Flexbox

A seguir aprenderemos as propriedades e técnicas referentes ao conceito moderno de flexbox. De forma complementar, você poderá estudar o flexbox também no site do joguinho Flex Froggy em https://flexboxfroggy.com/
O assunto flexbox será aplicado em formato handson, dentro da sala de aula, para melhor entendimento e prática de suas propriedades, bem como a utilização do flexfroggy. Espero você lá ;)

Flexbox trabalha com a disposição dos elementos de forma flexível para adaptá-los em diferentes dispositivos. É uma técnica que facilita o desenvolvimento de layouts simples sem a complexidade das técnicas que utilizam float. Para se utilizar o felxbox basta trabalhamos com o conceito de conteiner e itens, ou seja, pai e filhos. Basta eleger qualquer elemento pai como container e conter um ou mais elementos filhos.

Principais propriedades de flexbox:

display

A propriedade display deve ser setada para flex para haver a flexibilidade do layout.

Flex container

Propriedades para o conteúdo pai. Reflete no comportamento dos seus filho, os flex items

flex-direction

Controla a direção dos flex-items dentro de flex-container.

- row: os box filhos (flex-items) seguirão um fluxo em linha.
- column: os itens seguirão o fluxo em coluna.
- row-reverse: os itens seguirão um fluxo em linha, com sua ordem reversa. ● column-reverse: os itens seguirão um fluxo em coluna, com sua ordem reversa.

flex-wrap

Controla a quebra dos flex-items dentro de flex-container.

- wrap: permite a quebra.
- wrap-reverse permite a quebra de forma reversa.
- no-wrap: não permite quebra, deixando todos os elementos na mesma linha ou coluna. Mesmo que haja um tamanho definido para o elemento, esta não será obedecida caso a quantidade de elementos chegue ao limite da linha ou coluna.

justify-content

Controla o alinhamento dos filhos na horizontal.

- flex-start: flex-items se alinham com base no início do container.
- flex-end: flex-items se alinham com base no final do container
- space-between: se alinham com base no início do container, porém divide o espaço que sobra, entre dos flex-items.
- space-around: se alinham com base no início do container, porém divide o espaço que sobra, ao redor dos flex-items.

align-items

Controla o alinhamento dos filhos na horizontal.

- flex-start
- flex-end
- center
- stretch
- baseline

flex-flow

Short line (atalho) para *flex-wrap* e *flax-direction* respectivamente.

Projeto Prático

Para dominarmos HTML e CSS temos que praticar. Não aprendemos apenas na teoria. É

preciso praticar para fixar os conceitos e tirar as dúvidas.
Para desenvolver qualquer projeto você deverá se basear nas documentações de apoios no MDN e W3 Schools até que se atinja um domínio ao ponto de não precisar mais utilizá-la.

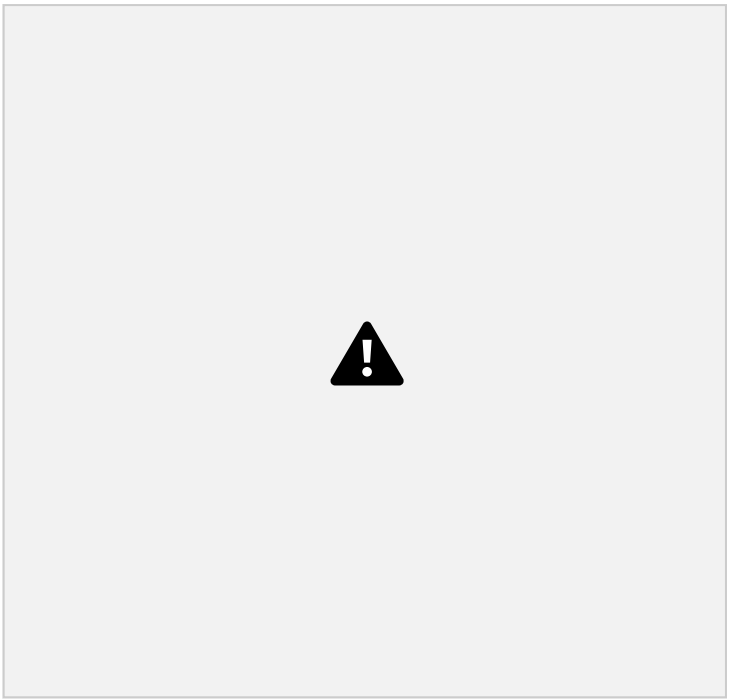
Bons treino!

Web Designer

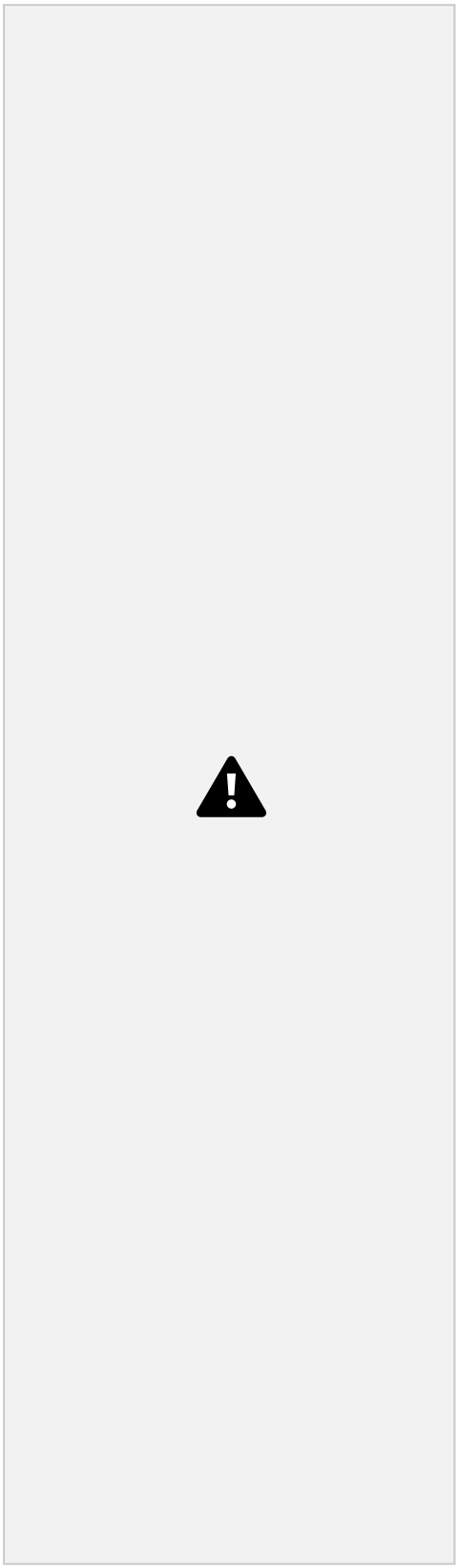
A empresa Web Designer LTDA contratou você como freelancer para desenvolver o layout HTML E CSS do template entregue pela equipe de designers.

Este projeto consiste em um site institucional simples e, como requisitos do projeto, deve ser desenvolvido utilizando as técnicas inline-block e float, bem como conceitos de box-model, position e media-queries.

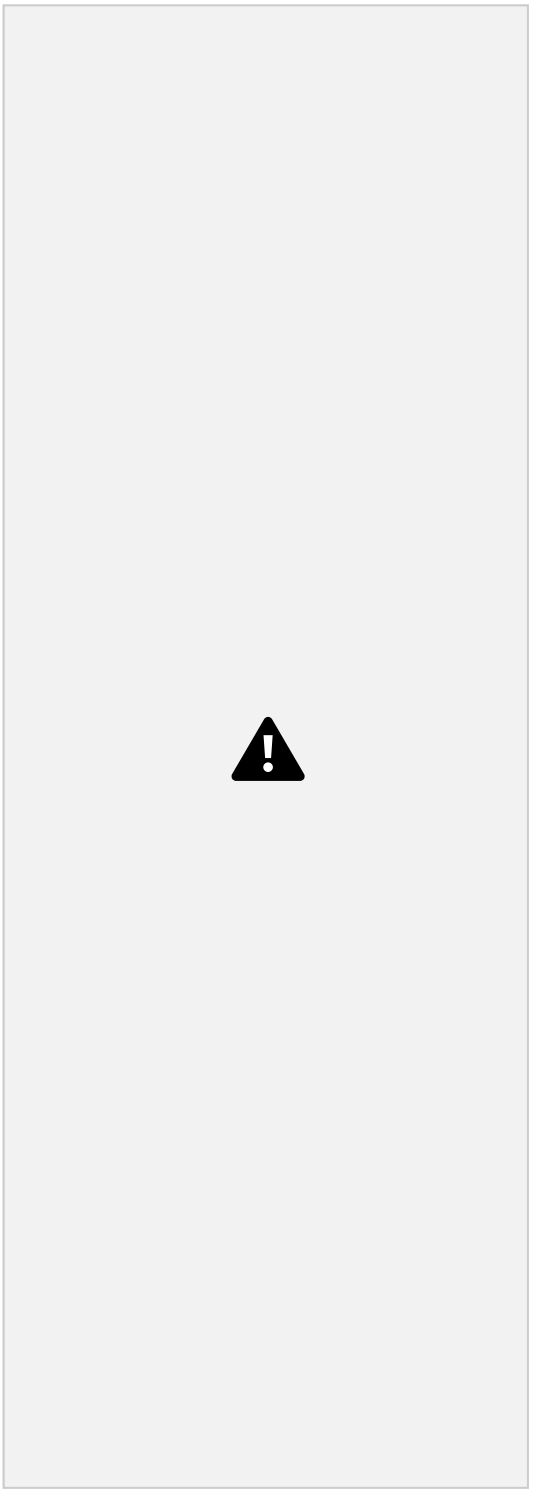
Para o menu em dispositivos móveis deve ser utilizada a técnica de position relative no header e absolute na **nav**bar, reproduzindo o efeito do template abaixo:



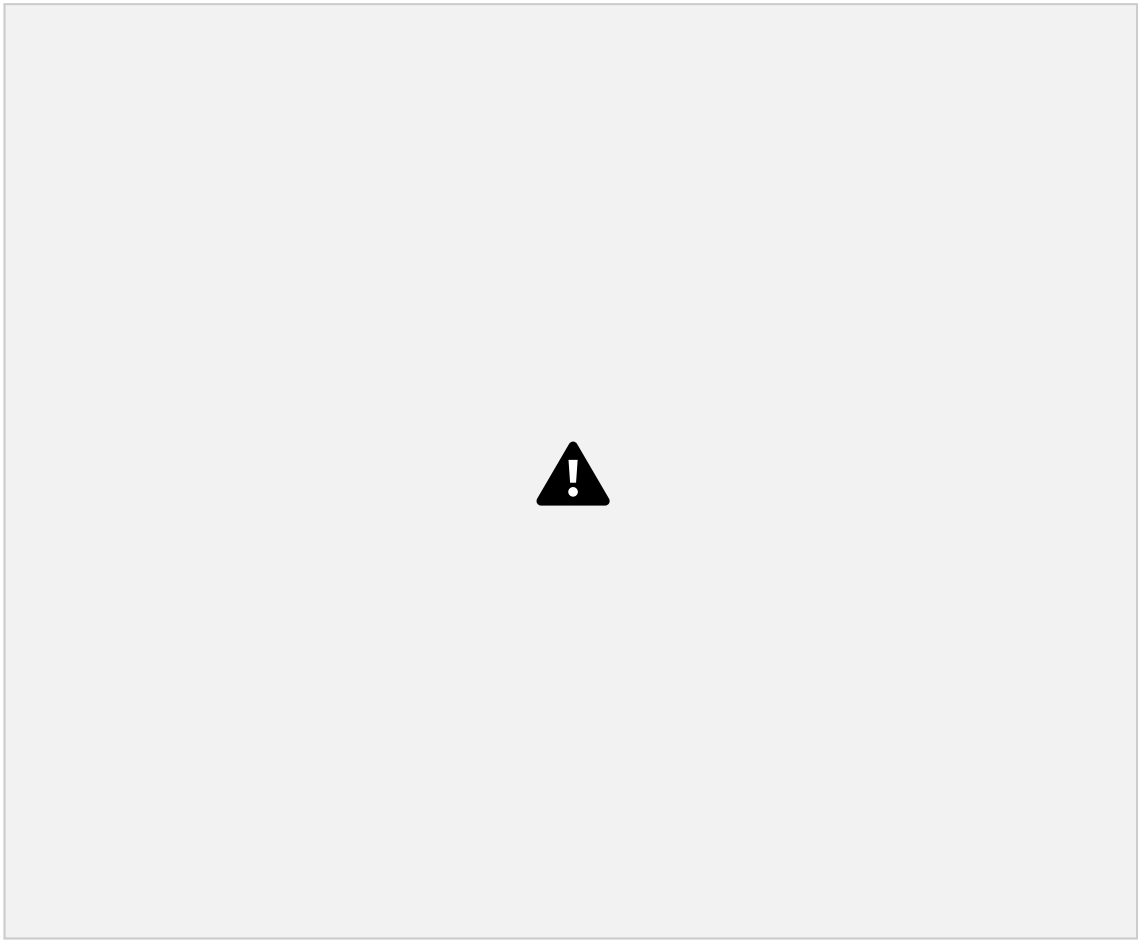
Mobile: Todo o CSS deve ser feito utilizando o conceito de mobile first para facilitar o trabalho com a responsividade.



Tablet: Teráumavisualização ligeiramentealteradacomotamanhodefontesepaddingem geral.



Desktop: Neste modelo temos um reajuste do layout, deixando de utilizar o position no cabeçalho para então utilizar as técnicas de inline-block abordadas durante o curso.



Tiny Love

Você recebeu um layout diretamente da equipe de designers da empresa Tiny Love. Como requisito para este projeto responsivo você deverá utilizar as técnicas de flexbox e design responsivo com media queries. Este projeto poderá ser feito em até 3 desenvolvedores front-end.