



FIAP

Shift



Aula 03

HTML & CSS

Parte 3



O QUE O MATERIAL COBRE

1

ÍCONES OPEN SOURCE: REMIX ICONS

2

PROPRIEDADES CSS: POSITION

3

TAGS HTML: TABELAS

4

LAYOUTS RESPONSIVOS: MEDIA QUERY

O QUE O MATERIAL COBRE

5

METATAGS PARA SEO

6

TAGS HTML: FORMULÁRIOS

7

SUGESTÕES DE EXERCÍCIOS

1

Ícones Open Source

Remix Icons

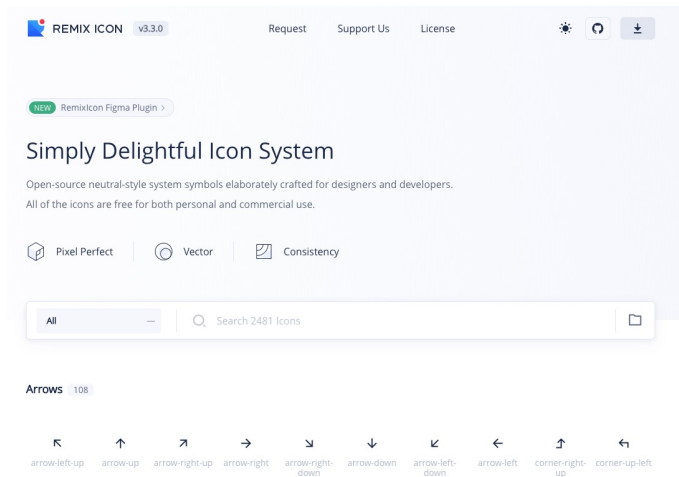
Remix Icons: “Sistema de ícones delicioso”

É isso que o slogan do site diz, e podemos concordar que sim!

Além de extremamente modernos e bonitos, os ícones do remix icons são fáceis de integrar e totalmente open source, você pode usar em qualquer projeto seu.

Procure colocar ícones nos seus projetos para melhorar a experiência do usuário e deixar o design muito bonito.

Site: <https://remixicon.com>



Remix Icons: Instalando via CDN

Existem algumas maneiras de importar a biblioteca Remix Icons no seu projeto, uma das mais simples é via CDN, ou seja, chamando o arquivo CSS principal do Remix Icons diretamente no seu HTML via tag `<link>`.

URL da CDN: <https://cdn.jsdelivr.net/npm/remixicon@3.3.0/fonts/remixicon.css>

```
<link
  href="https://cdn.jsdelivr.net/npm/remixicon@3.3.0/fonts/remixicon.css"
  rel="stylesheet"
/>
```

OBS: a versão pode estar além da 3.3.0, isso não é um problema, pode seguir.

Remix Icons: Usando os ícones



```
<!-- Alguns exemplos -->
<i class="ri-arrow-left-up-line"></i>
<i class="ri-bank-line"></i>
<i class="ri-wifi-line"></i>
```

Para utilizar o ícones, basta ir até o site, escolher o ícone desejado e copiar a tag `<i>` que vai aparecer e colar dentro do seu projeto.

Para aumentar o tamanho e trocar a cor utilize as propriedades **font-size** e **color** do CSS

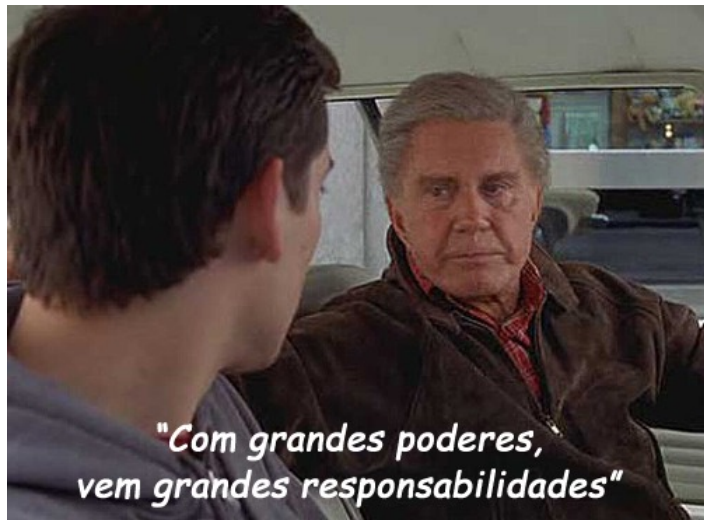
Você também tem as opções de baixar como SVG ou PNG – muito útil para ajudar na criação de layouts no Figma.

2

Propriedades CSS

Position

Propriedades CSS: position



Ao utilizar flex ou grid no HTML com CSS é possível criar layouts de diversos tipos, porém existe uma outra propriedade complementar, para deixar os layouts ainda melhores.

Com a propriedade **position** você pode controlar a posição de um elemento em relação a outros elementos na página, seja pai, irmãos ou o próprio documento.

Atenção: use com responsabilidade! 😊

Propriedades CSS: position: static;

```
<style>
  .element {
    position: static;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
</div>
```

Utilizando o **position: static;** o elemento segue o fluxo normal, sem nenhuma alteração de posicionamento.

Esse é o valor padrão que é aplicado para todos os elementos.



Propriedades CSS: top, bottom, left, right

```
<style>
  .element:first-child {
    top: 0;
    left: 0;
  }

  .element:last-child {
    bottom: 0;
    right: 0;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
  <div class="element">Hello!</div>
</div>
```

Antes das próximas propriedades do position, que de fato vão influenciar na posição do elemento é preciso saber que existem propriedades que movem esses elementos e são utilizadas em conjunto, que são:

- **top** – define a distância do topo;
- **bottom** – define a distância da base;
- **left** – define a distância da esquerda;
- **right** – define a distância da direita.

Utilize apenas uma propriedade para o eixo x e outra para o eixo y, caso contrário elas se cancelam.

Propriedades CSS: position: relative;

```
<style>
  .element {
    position: relative;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
</div>
```

O **position: relative;** é muito parecido com o static, ele mantém o posicionamento normal, porém habilita o uso das propriedades de movimentação.



Propriedades CSS: position: absolute;

```
<style>
  .wrapper {
    position: relative;
  }

  .element {
    position: absolute;
    top: 10px;
    left: 10px;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
</div>
```

O **position: absolute;** faz com que o elemento seja posicionado em relação ao seu elemento ancestral mais próximo que possua uma propriedade position também. Caso não seja encontrado, ele será posicionado de acordo com o documento (<body>).

Quando aplicado, por padrão, o position absolute fica por cima de todo elemento static ou relative.

Propriedades CSS: position: fixed;

```
<style>
  .element {
    position: fixed;
    top: 10px;
    left: 10px;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
</div>
```

O **position: fixed;** faz com que o elemento seja sempre posicionado fixamente em relação ao viewport (janela).

Sempre que a barra de rolagem for alterada o elemento continuará no mesmo lugar.



Propriedades CSS: position: sticky;



```
<style>
  .element {
    position: sticky;
    top: 10px;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
</div>
```

O **position: sticky;** faz com que o elemento seja posicionado de acordo com a direção atribuída até que ele fique fora da área visível do viewport (janela).

Após perder a visualização ele tem o mesmo comportamento do position fixed, mantendo fixo no viewport independente da rolagem da página.



Propriedades CSS: z-index

```
<style>
  .element {
    position: absolute;
    width: 50px;
    height: 50px;
  }

  .element:first-child {
    top: 10px;
    left: 10px;
    background-color: red;
    z-index: 2;
  }

  .element:last-child {
    top: 20px;
    left: 20px;
    background-color: blue;
    z-index: 1;
  }
</style>

<div class="wrapper">
  <div class="element">Hello!</div>
  <div class="element">World!</div>
</div>
```

O **z-index** é utilizado para controlar a ordem de empilhamento dos elementos que possuem a propriedade position setada. Um z-index maior fará com que o elemento fique por cima de outro que tem um z-index menor.

Para utilizar o z-index defina apenas números inteiros no valor da propriedade.

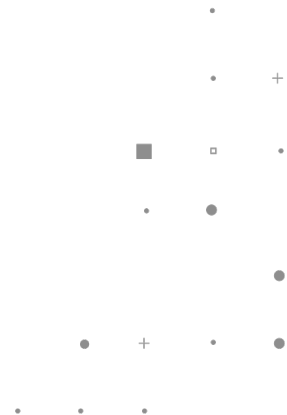
Atenção: use com responsabilidade! 😊



3

Tags HTML

Tabelas



Um pouco sobre tabelas no HTML

No início do HTML, em suas primeiras versões, as páginas eram construídas em sua maior parte utilizando tabelas. Com o tempo o HTML foi evoluindo, e as tabelas continuaram com seu espaço, mas com o seu objetivo mais especificado.

Elas são utilizadas para exibir dados de forma organizada e tabular através de linhas e colunas, mostrando os elementos em grade, como no exemplo abaixo.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

Tags HTML: <table>

<table></table>

A tag **<table>** define uma tabela dentro do documento HTML e precisa ser fechada com **</table>**. É o elemento pai que envolve todas as outras tags relacionadas como as linhas, colunas e outras.



Tags HTML: <thead>

<thead></thead>

A tag **<thead>** define a sessão de cabeçalho da tabela onde ficará a linha principal da tabela com o título de cada coluna e deve ser fechada com **</thead>**. Essa tag é opcional.



Tags HTML: **<tbody>**

<tbody></tbody>

A tag **<tbody>** define a sessão do corpo da tabela, onde ficarão as linhas com o conteúdo e deve ser fechada com **</tbody>**. Essa tag é opcional.



Tags HTML: <tfoot>

<tfoot></tfoot>

A tag **<tfoot>** define a sessão do rodapé da tabela, onde ficarão as linhas com conteúdos finais e deve ser fechada com **</tfoot>**. Essa tag é opcional.



Tags HTML: <tr>

<tr></tr>

A tag **<tr>** define uma linha na tabela onde ficarão as colunas e deve ser fechada com **</tr>**.



Tags HTML: <th>

<th></th>

A tag **<th>** define uma coluna de cabeçalho na linha (<tr>) e deve ser fechada com **</th>**. Essa tag é opcional.



Tags HTML: <td>

<td></td>

A tag **<td>** define uma coluna de dados na linha (<tr>) e deve ser fechada com **</td>**.



Tags HTML: propriedades rowspan e colspan

<td rowspan="2"></td>

<td colspan="2"></td>

As propriedades **rowspan** e **colspan** podem ser utilizadas em qualquer <td> para fazer mesclagem de linhas (row) ou colunas (col) para prevenir que a tabela quebre.

Utilize a propriedade seguido de um número inteiro para fazer a mesclagem na tabela.



Tabelas e CSS: algumas tricks

```
<style>
  table tr:nth-child(odd) {
    background: #cbd5e1;
  }

  table tr:nth-child(even) {
    background: #f1f5f9;
  }

  table tr:hover {
    background: #64748b;
    color: #fff;
  }
</style>

<table>
  <tr>
    <td>Olá</td>
    <td>Mundo</td>
  </tr>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
  <tr>
    <td>Hola</td>
    <td>Mundo</td>
  </tr>
  <tr>
    <td>Bonjur</td>
    <td>le Monde</td>
  </tr>
</table>
```

É possível mesclar o CSS junto as tabelas para criar linhas e colunas mais dinâmicas. Podemos estilizar os elementos e também adicionar algumas pseudo-classes, como no exemplo ao lado.



4

Layouts Responsivos

Media Query

Mobile first, medidas relativas e media query

Hoje em dia temos uma alta gama de dispositivos no mundo todo e as pessoas utilizam mais o celular do que o computador, por isso precisamos programar seguindo o conceito de **mobile first**, que nada mais é que começar pelo dispositivo menor (mobile) e ir adiante até outros dispositivos maiores (computadores, TVs, etc).

Esses dispositivos podem ter tamanhos completamente diferentes e por isso precisamos de algo que cubra bem todo os cenários.

É possível utilizar as **media queries** juntamente com **medidas relativas como porcentagem**.



Propriedades CSS: @media (media query)

```
div {  
  width: 100%;  
  height: 10rem;  
  background: red;  
}  
  
@media (min-width: 768px) {  
  /* Estilos aplicados em telas  
    maiores ou iguais a 768px */  
  div {  
    background: blue;  
  }  
}  
  
@media (min-width: 1200px) {  
  /* Estilos aplicados em telas  
    maiores ou iguais a 1200px */  
  div {  
    background: green;  
  }  
}
```

Com as media queries podemos aplicar diferentes estilos com base nas características da tela ou do dispositivo, chamamos isso de breakpoint (ponto de quebra).

Existe uma gama de propriedades que podemos usar dentro do **@media** como **min-width**, **max-width**, **orientation**, **media** e outros.

O mais comum é criar a regra para o mobile, e depois ir colocando variações com **@media** de **min-width** para alcançar os outros tamanhos, assim cobrindo a maior parte dos dispositivos conhecidos.

Qualquer regra CSS pode ser aplicada dentro do media query.

Breakpoints mais usados

Depois de muito estudo e bastante teste alguns padrões de tela foram definidos, que ajudam bastante na hora de criar nosso código com media query.

Podemos utilizar a tabela de breakpoints do **TailwindCSS** para definir os **min-width** corretamente.

Tabela de breakpoints: <https://tailwindcss.com/docs/responsive-design>

Breakpoint prefix	Minimum width	CSS
<code>`sm`</code>	640px	<code>`@media (min-width: 640px) { ... }`</code>
<code>`md`</code>	768px	<code>`@media (min-width: 768px) { ... }`</code>
<code>`lg`</code>	1024px	<code>`@media (min-width: 1024px) { ... }`</code>
<code>`xl`</code>	1280px	<code>`@media (min-width: 1280px) { ... }`</code>
<code>`2xl`</code>	1536px	<code>`@media (min-width: 1536px) { ... }`</code>

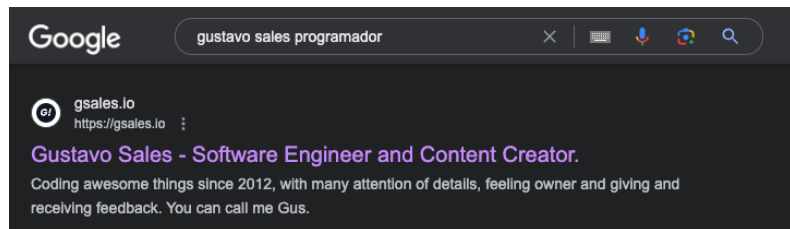
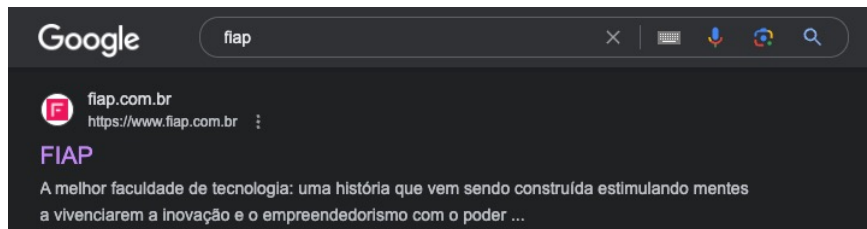
5

Metatags para **SEO**

Algoritmos de ranqueamento, metatags e SEO

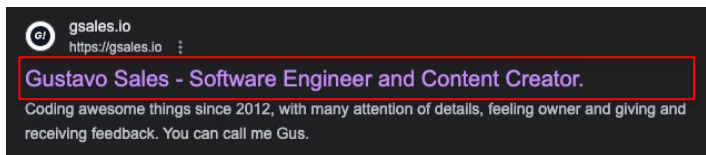
Todo dia os algoritmos de ranqueamento de sistemas de busca como Google lêem novos sites e páginas na internet a fim de mostrarem links relevantes quando alguma pessoa realiza buscas a procura de resolver seu problema ou achar um produto para comprar ou algo relacionado.

Existe uma tecnica muito famosa que se chama SEO (Search Engine Optimization) que faz com que esses algoritmos encontrem os sites e parte fundamental dessa tecnica é aplicar corretamente as metatags no seu site para você ser visto pelo Google.



Metatags SEO: <title>

```
<meta>Título da Página</title>
```



A tag **<title>** que já foi vista antes é referente ao título da página e extremamente importante para aparecer no Google, sem ela o algoritmo já te exclui da lista. Além do SEO ela ajuda muito em experiência do usuário, por isso escreva bons títulos.



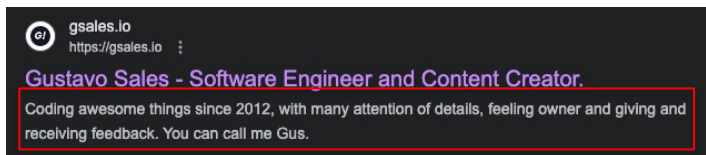
Metatags SEO: <meta name="description" />

```
<meta  
  name="description"  
  content="Descrição da página"  
/>
```

Para todas as outras metatags de SEO precisamos utilizar a tag **<meta />** dentro da sessão **<head>** do nosso site.

A **description** é referente ao pequeno texto abaixo do título da página no Google.

A recomendação é definir uma description com até 160 caracteres.



Metatags SEO: <meta name="robots" />

```
<!-- Para garantir a ativação -->
<meta
  name="robots"
  content="index, follow"
/>

<!-- Para desativar -->
<meta
  name="robots"
  content="noindex, nofollow"
/>
```

A metatag **robots** informa se o Google e outros mecanismos devem indexar sua página no algoritmo deles ou não.

Para sistemas e aplicações internas muitas das vezes não é necessário que apareça no Google, então você pode desativá-la utilizando o valor **noindex, nofollow**.



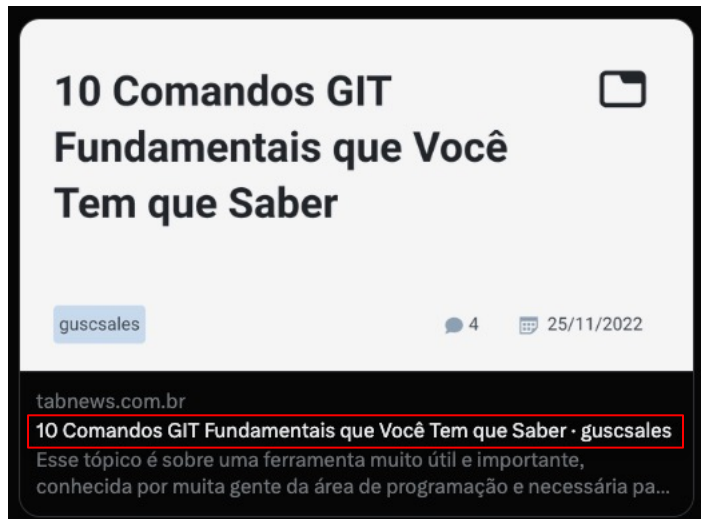
Metatags SEO: <meta name="og:title" />

```
<meta
  name="og:title"
  content="Título da página"
/>
```

As metatags que tem início “og” são as que aparecem em pre-visualização de redes sociais.

Você pode usar o mesmo valor das outras metatags já definidas ou alterar caso precise.

Para o título utilize **og:title**.



Metatags SEO: <meta name="og:description" />

```
<meta
  name="og:description"
  content="Descrição da página"
/>
```

Para a descrição você pode usar o mesmo valor da metatag description comum seguindo a mesma regra dos 160 caracteres. Para definir utilize **og:description**.



Metatags SEO: <meta name="og:image" />

```
<meta
  name="og:image"
  content="URL da imagem referente a página"
/>
```

Define a imagem que irá aparecer na pre-visualização do post. Normalmente caso nenhuma imagem seja definida as redes sociais procuram uma imagem no seu site, se não encontrar nada ficará vazia.

Defina utilizando **og:image**.

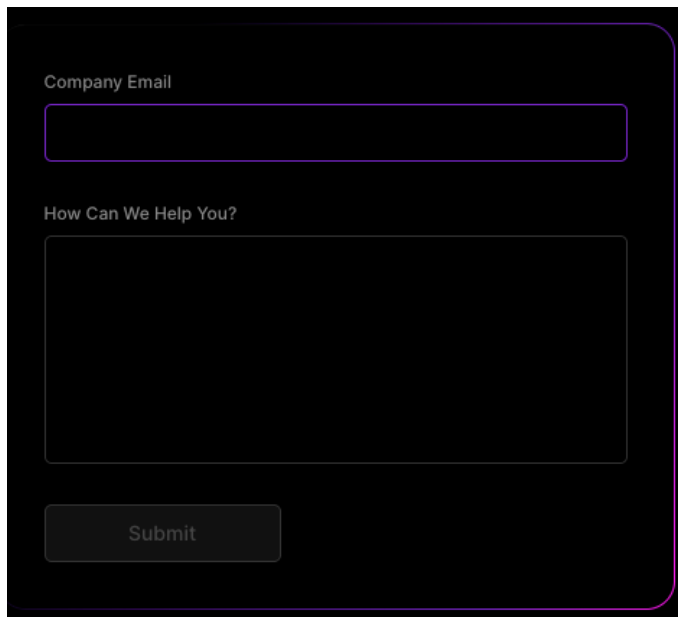


6

Tags HTML

Formulários

Formulários no HTML



Company Email

How Can We Help You?

Submit

São elementos utilizados para coletar informações e valores dos usuários por meio de campos interativos na tela. Com esses campos é possível inserir dados como textos, números, seleções, opções de escolha e mais. Esses dados normalmente são enviados para algum servidor backend.

Existem vários tipos de formulários e eles são criados de acordo com a situação.

Alguns exemplos comuns são: formulários de cadastro, login e também envio de contato.

Tags HTML: <form>

```
<form  
  method="GET"  
  action="URL_DE_ENVIO">  
  
</form>
```

A tag **<form>** define a sessão geral do formulário. Todos os elementos devem ficar dentro dela.

Essa tag conta também com duas propriedades principais:

- **action** – especifica a URL que o formulário irá chamar e enviar os dados;
- **method** – define o tipo de envio de dados para o servidor, sendo GET ou POST:
 - **GET** – os dados vão de maneira visível na URL;
 - **POST** – os dados vão de maneira escondida para o servidor.

Tags HTML: <input>

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

A tag **<input>** representa vários tipos de elementos de formulário, ela pode ser específica para texto, email, número, data, arquivo e outros tipos. Ela conta com algumas propriedades principais (* = requeridas):

- **type*** - especifica o tipo de entrada, como **text, password, email, number, checkbox, radio, submit** e outros;
- **name*** - define o nome do campo que é usado para enviar os dados para o servidor.

Tags Input: value

```
<form>
  <input
    type="text"
    name="name"
    value="Meu valor inicial"
  />
</form>
```

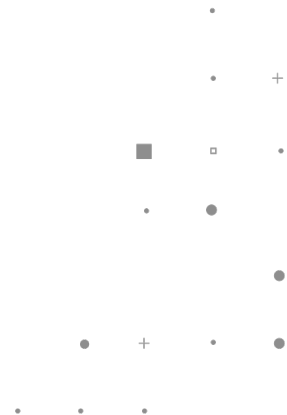
A propriedade **value** é utilizada para definir o valor inicial do campo caso exista.



Tags Input: placeholder

```
<form>
  <input
    type="text"
    name="name"
    placeholder="Digite seu nome"
  />
</form>
```

A propriedade **placeholder** é utilizada para orientar o campo antes de ser preenchido.



Tags Input: required

```
<form>
  <input
    type="text"
    name="name"
    required
  />
</form>
```

A propriedade **required** é utilizada para deixar o campo obrigatório, não permitindo a submissão do formulário para o servidor em caso de não preenchimento.



Tags Input: disabled

```
<style>
  input[disabled] {
    opacity: 0.2;
  }
</style>

<form>
  <input
    type="text"
    name="name"
    disabled
  />
</form>
```

A propriedade **disabled** é utilizada para desabilitar o campo não permitindo a interação do usuário. Normalmente ela pode ser estilizada com CSS para melhor experiência.



Tags Input: readonly

```
<form>
  <input
    type="text"
    name="name"
    readonly
  />
</form>
```

A propriedade **readonly** é utilizada para manter o campo somente leitura. É muito parecido com o disabled, mas tem o objetivo final diferente. Somente leitura é diferente de desabilitado.



Tags Input: maxlength

```
<form>
  <input
    type="text"
    name="name"
    maxlength="10"
  />
</form>
```

A propriedade **maxlength** é utilizada para definir um tamanho máximo de caracteres para aquele campo.



Tags Input: min e max

```
<form>
  <input
    type="number"
    name="name"
    min="3"
    max="8"
  />
</form>
```

As propriedades **min** e **max** são utilizadas para definir o valor numérico mínimo e máximo do campo.

Pode ser utilizada apenas com o type number.



Tags HTML: <label>

```
<form>
  <label
    for="name-field">
    Digite seu nome
  </label>
  <input
    type="text"
    name="name"
    id="name-field"
  />
</form>
```

A tag **<label>** é utilizada para definir o título do campo. Para atrelar totalmente o campo a label pode-se utilizar as propriedades **for (na <label>)** e **id (no <input>)** com o mesmo nome.



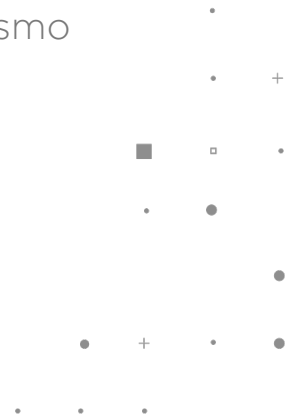
Tags HTML: <input type="checkbox">

```
<form>
  <input
    type="checkbox"
    name="option"
    value="value1"
    id="opt-1-field"
  />
  <label
    for="opt-1-field">
    Valor 1
  </label>

  <input
    type="checkbox"
    name="option"
    value="value2"
    id="opt-2-field"
  />
  <label
    for="opt-2-field">
    Valor 2
  </label>
</form>
```

A tag **<input>** com o **type checkbox** permite criar caixas de seleção onde é possível selecionar uma ou mais opções.

Note que todos os inputs precisam ter o mesmo valor na propriedade **name** e também a propriedade **value** definida.



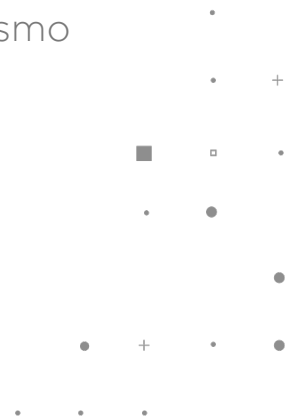
Tags HTML: <input type="radio">

```
<form>
  <input
    type="radio"
    name="option"
    value="value1"
    id="opt-1-field"
  />
  <label
    for="opt-1-field">
    Valor 1
  </label>

  <input
    type="radio"
    name="option"
    value="value2"
    id="opt-2-field"
  />
  <label
    for="opt-2-field">
    Valor 2
  </label>
</form>
```

A tag **<input>** com o **type radio** permite criar caixas de seleção onde é possível selecionar apenas uma opção.

Note que todos os inputs precisam ter o mesmo valor na propriedade **name** e também a propriedade **value** definida.



Tags HTML: <input type="file">

```
<form>
  <input
    type="file"
    name="files"
    multiple
  />
</form>
```

A tag **<input>** com o **type file** permite selecionar e enviar arquivos para o servidor.

Para fazer upload de múltiplos arquivos utilize a propriedade **multiple** – que é opcional.



Tags HTML: <input type="submit">

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

  <input
    type="submit"
    value="Enviar"
  />

  <!-- Outra opção -->
  <button type="submit">
    Enviar
  </button>
</form>
```

A tag **<input>** com o **type submit** faz com que o formulário seja enviado para a URL definida no action.

É possível também utilizar com outra tag, chamada **<button type="submit">** com apenas uma pequena mudança de sintaxe.

Tags HTML: <input type="reset">

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

  <input
    type="reset"
    value="Limpar campos"
  />

  <!-- Outra opção -->
  <button type="reset">
    Limpar campos
  </button>
</form>
```

A tag **<input>** com o **type reset** faz com que o formulário retornado para seus valores iniciais. É possível também utilizar com outra tag, chamada **<button type="reset">** com apenas uma pequena mudança de sintaxe.



7

Sugestões de **Exercícios**

Exercícios para treinar CSS, Tableas, Media Query, Forms e mais

Com o site criado nas últimas aulas

- Atualize o header do seu site para ficar fixo na tela, assim possibilitando uma melhor experiência para o usuário;
- Utilize as inspirações e monte seu site com uma versão mobile;
- Aplique as metatags SEO para o seu site;
- Crie um formulário de contato;

Para se inspirar

- Lee Robinson Personal Website: <https://leerob.io/>
- Zeno Rocha Personal Website: <https://zenorocha.com/>
- Gustavo Sales Personal Website: <https://gsales.io/>

Bons estudos! 😊



OBRIGADO

@guscsales - gsales.io

FIAP

Copyright © 2023 | Professor Gustavo Campos Sales

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

