

# 3 formas de centralizar qualquer elemento em CSS

Para centralizar em CSS, existem três cenários clássicos que você enfrenta. Veja como chegar na interface desejada usando `text-align`, `display flex`, `display grid`, `position relative` e `position absolute`.

Publicado em 26/05/2022 Atualizado em 17/07/2022

- centralizar
- css
- alinhar
- vertical
- horizontal
- imagem
- display
- position
- grid

Front-end

## Shortcut

deMenezes

- Home
- Sobre
- Contato
- Referências

```
.elemento-pai {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.elemento-filho {  
    /* nothing */  
}
```

Se não resolver (e para entender o porquê das coisas), segue a leitura ;)

Para centralizar em CSS, você deve estar em um desses três cenários:

- Centralizar elementos de texto
- Centralizar imagens ou blocos (como divs)
- Ou alinhar no sentido horizontal e/ou vertical

Veja agora como tratar cada um dos casos.

## Centralizar em CSS: elementos de texto

Esse é o caso mais simples, basta você usar a propriedade `text-align: center`.

Apesar da propriedade começar com a palavra **text**, os elementos inline, como `span` ou `strong`, não sofrem nenhuma alteração. Essa propriedade muda apenas o conteúdo de elementos que possuem o `display` padrão como `block`, tal como `p`, `div` e `section`.

Para alinhar ao centro uma imagem usando `text-align`, coloque essa propriedade em um elemento `block`, como uma `div`, e coloque a imagem dentro dela.

```
<div class="container">
  <h1>Centralizar em CSS</h1>
  <p>Uma hora a gente aprende...</p>
  
</div>
```

```
.container {
  text-align: center;

  border: 1px solid;
}
```

Nesse exemplo, o título, o parágrafo e a imagem ficam centralizados, pois eles herdam o estilo do elemento pai `div.container`.

# Centralizar em CSS

Uma hora a gente aprende...



- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

Você pode usar outra forma para centralizar imagens e também elementos de bloco. Veja abaixo:

## Como centralizar imagens, divs e outros elementos em CSS

Existem duas formas de centralizar elementos de bloco:

1. Aplicando estilo no próprio elemento
2. Aplicando estilo no elemento pai

Cada método tem suas vantagens e desvantagens, e você precisa analisar para entender qual é melhor na sua situação.

Uma regra que você sempre deve ter em mente é que, para que um elemento seja centralizado na horizontal, a sua largura deve ser menor que a largura do elemento pai. Veja o exemplo abaixo:

```
<div class="gallery">
  
</div>
```

```
.gallery {
  width: 100%; /* isso é padrão para elementos div */
```

**deMenezes**

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

```
width: 80%; /* usei 80% como exemplo, porém vale qualquer valor abaixo de 100% */
}
```

Veja agora como centralizar o elemento `.image` aplicando propriedades CSS apenas nele:

## Aplicando estilo no próprio elemento

Nesse exemplo de uma galeria de imagens, vou centralizar a única imagem que existe.

Se você prestou atenção à regra da largura mencionada acima, apenas coloque `margin: auto` nas laterais da imagem:

```
.gallery {
  width: 100%;
}

.gallery .image {
  width: 80%;
  display: block; /* Necessário para o margin funcionar */
  margin-left: auto; /* aqui */
  margin-right: auto; /* e aqui */
}
```

Ou use um shorthand:

```
.gallery {
  width: 100%;
```

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

**deMenezes**

```
margin: 0 auto; /* zero para top e bottom, e auto para as laterais */  
}
```

Esse shorthand só funcionará, caso a margem superior e inferior do elemento seja 0. Caso contrário, coloque outro valor no lugar de zero, ou defina as margens laterais em propriedades separadas.



Centralizar imagem usando margin auto

Mas existe uma forma de centralizar elementos sem aplicar nenhuma propriedade neles:

## Aplicando estilo no elemento pai

Para isso, você precisa do recurso CSS mais amado dos últimos anos (quem não gosta dele, está errado): o **Flexbox**

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

**deMenezes**

```
.gallery {  
    display: flex;  
    flex-direction: row; /* row é padrão */  
    justify-content: center;  
}
```

Na maioria dos casos isso basta, e nada precisa ser feito no elemento filho `.image`.

Nesse exemplo, a propriedade `justify-content: center` está centralizando a imagem, porque o `flex-direction` padrão é `row`.

Caso a direção seja mudada para `column`, você precisará centralizar os elementos usando `align-items`, e não mais `justify-content`, veja:

```
.gallery {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}
```

Como você deve ter desconfiado, `justify-content` e `align-items` alinham tanto na vertical, quanto na horizontal. Tudo depende da direção (`flex-direction`) do Flexbox.

## Centralizar em CSS: sentido vertical e horizontal

A regra que falei no capítulo anterior sobre o sentido horizontal continua valendo. Mas agora na vertical: para que um elemento seja centralizado verticalmente, a sua altura deve ser menor que a altura do elemento pai.

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

```
.gallery {  
    display: flex;  
    justify-content: center; /* Alinhamento horizontal */  
    align-items: center; /* Alinhamento vertical */  
    width: 100%;  
    height: 300px;  
}  
  
.gallery .image {  
    width: 40%;  
    height: 150px;  
}
```



Centralizar imagem na vertical e horizontal usando display flex

As propriedades aplicadas em `.image` servem apenas para que ela seja menor do que o elemento pai.

Aqui, caso o `flex-direction` fosse `column`, nada mais precisaria mudar. A única diferença é

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

**deMenezes**

```
.gallery {  
    display: flex;  
    flex-direction: column; /* Mudou a direção */  
    justify-content: center; /* Agora, alinhamento vertical */  
    align-items: center; /* e alinhamento horizontal */  
    width: 100%;  
    height: 300px;  
}  
  
.gallery .image {  
    width: 40%;  
    height: 150px;  
}
```

## Display Grid

Muito semelhante ao exemplo anterior:

```
.gallery {  
    display: grid;  
    place-items: center; /* Alinhamento horizontal e vertical */  
    width: 100%;  
    height: 300px;  
}  
  
.gallery .image {  
    width: 10%;  
    height: 30px;  
}
```

```
<div class="container">
  
  <p class="text">Café da manhã</p>
</div>
```

Para começar, aplique a propriedade position:

```
.container {
  position: relative;
}

.container .image {
  display: block;
  width: 100%;
}

.container .text {
  position: absolute;
  z-index: 1;
}
```

Simples assim:

- relative no pai

**deMenezes**

- Home
- Sobre
- Contato
- Referências

As propriedades colocadas na imagem servem apenas para ela ocupar todo o espaço do container.

Mas o texto ainda não está por cima da imagem. O que falta? Falta informar qual será sua posição:

- Para o sentido vertical, use `top` ou `bottom`;
- Para o sentido horizontal, use `left` ou `right`;

```
.container {  
    position: relative;  
}  
  
.image {  
    display: block;  
    width: 100%;  
}  
  
.text {  
    position: absolute;  
    z-index: 1;  
    top: 50px; /* 50px de distância do topo */  
    left: 25px; /* 25px de distância da esquerda */  
  
    background-color: white;  
    padding: 1rem;  
}
```

A leitura padrão do HTML é **um elemento após o outro**. Mas dessa forma que expliquei, o elemento posicionado (no caso o texto) ficará **por cima** dos outros elementos irmãos.

**deMenezes**

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)



Posicionar texto sobre imagem usando position

Além disso, ele ficará "preso" dentro do elemento com `position: relative` (nesse exemplo, o `.container`).

Para que o texto fique realmente centralizado, faltam 2 etapas.

Primeiro, troque os valores de `top` e `left` para `50%` (isso quer dizer a metade do container).

Assim você vai perceber que o texto está "quase centralizado". Isso acontece porque o que está centralizado não é o ponto central do texto, e sim o ponto superior esquerdo dele.

**deMenezes**

- Home
- Sobre
- Contato
- Referências



Posicionar texto sobre imagem usando position (está quase)

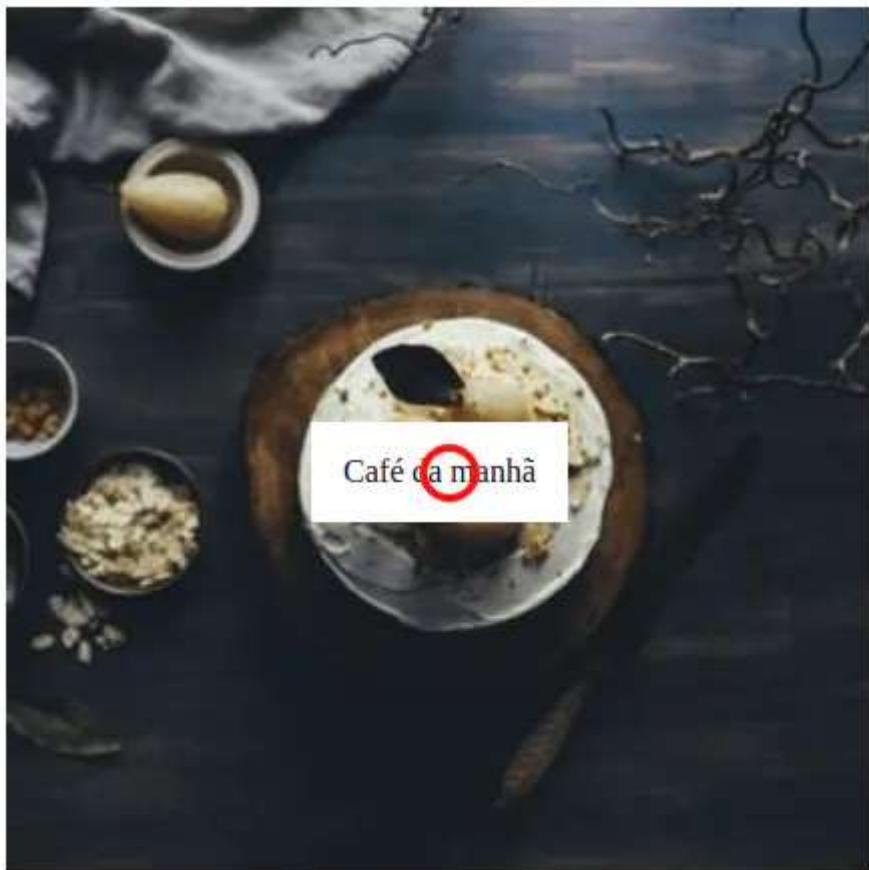
E assim você acaba de chegar à última etapa que é mover o ponto do elemento que realmente é posicionado:

```
.container {  
    position: relative;  
}  
  
.image {  
    display: block;  
    width: 100%;  
}  
  
.text {
```

- Home
- Sobre
- Contato
- Referências

**deMenezes**

```
background-color: white;  
padding: 1rem;  
}
```



Posicionar texto sobre imagem usando position (agora sim!)

O transform movimenta esse ponto focal para onde você quiser. Agora, nada mais arranca seu texto do centro 😊

Uma alternativa ao position: absolute é usar display: grid.

## Display Grid

Olha ele aí de novo.

**deMenezes**

- Home
- Sobre
- Contato
- Referências

```
<div class="container">
  
  <p class="text">Café da manhã</p>
</div>
```

Para começar a sobrepor os elementos, veja esta magia:

```
.container {
  display: grid;
}

.container > * {
  grid-area: 1 / -1;
}

.container .text {
  background-color: white;
  padding: 1rem;
}
```

Isso transforma o container em um grid. Em seguida, o `grid-area: 1 / -1` faz os filhos do primeiro nível do container começarem na primeira linha e primeira coluna (1), e terminarem na última linha e última coluna (-1) desse grid.

E como posicionar o texto sobre a imagem agora? Ele deve receber duas propriedades:

- Alinhamento horizontal: `justify-self` com os valores `start`, `center` ou `end`
- Alinhamento vertical: `align-self` também com os mesmos valores `start`, `center`, ou `end`

Por exemplo, para posicionar o texto à direita e ao meio da imagem, o código fica assim:

```
.container .text {  
    justify-self: end;  
    align-self: center;  
  
    background-color: white;  
    padding: 1rem;  
}
```



Posicionar texto sobre imagem usando display grid. Texto alinhado à direita e ao meio da imagem

**deMenezes**

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)

```
.container .text {  
    justify-self: left; /* opcional, pois é o valor padrão */  
    align-self: end;  
  
    background-color: white;  
    padding: 1rem;  
}
```



Posicionar texto sobre imagem usando display grid. Texto alinhado à esquerda e no bottom da imagem

O que achou dessas formas de centralizar em CSS?

## Callback

Em resumo, veja que tipo de conteúdo você precisa centralizar:

- Elementos textuais
- Imagens, divs e outros elementos de bloco

- Home
- Sobre
- Contato
- Referências

**deMenezes**

- `display: flex`
- `display: grid;`
- `position: relative e position: absolute`

Dúvidas? Deixe nos comentários, e vamos codar.

## Deixe um Comentário

 Entrar ▼



Iniciar a discussão...

FAZER LOGIN COM

OU REGISTRE-SE NO DISQUS 

Nome



Compartilhar

Mais votados Mais recentes Mais antigos

Seja o primeiro a comentar.

## Veja outros posts sobre Front-end

deMenezes

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)



Front-end

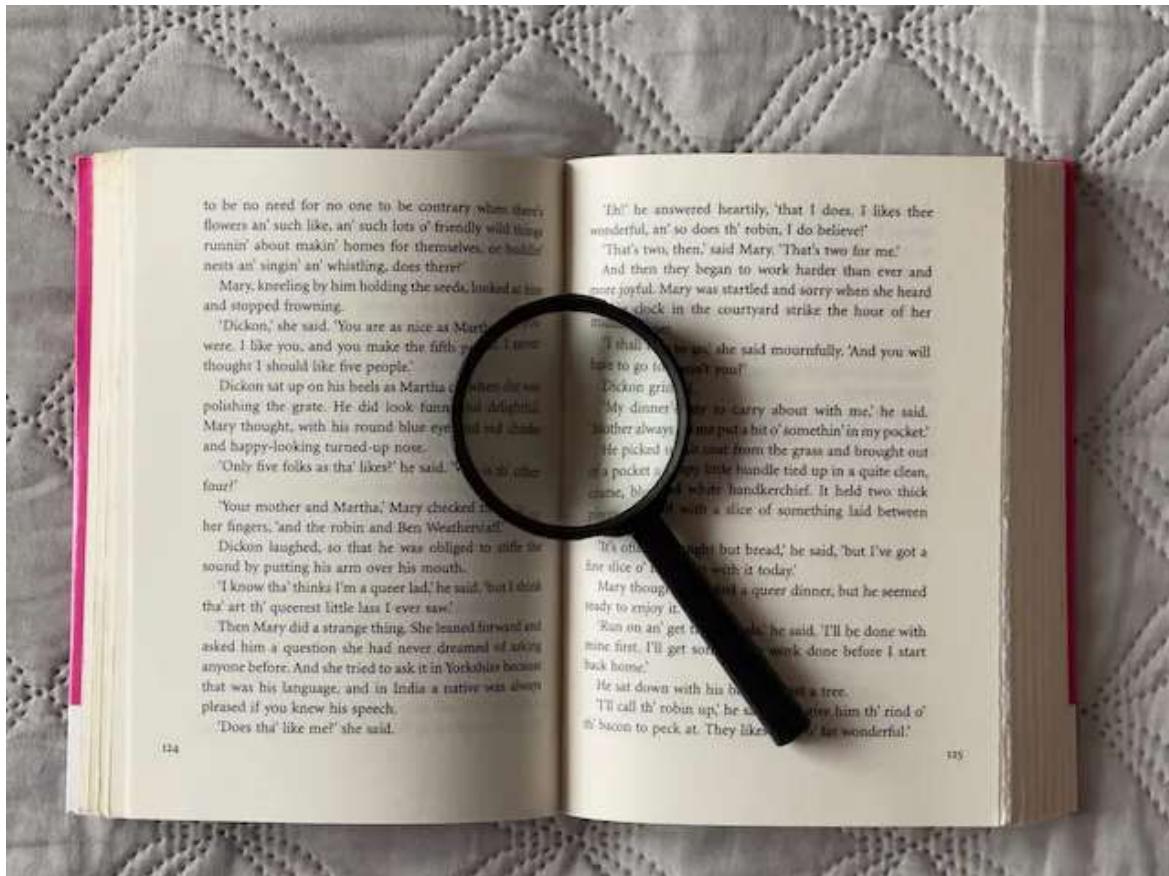
**Quem tem medo do reduce? Entenda esse método agora ou seu dinheiro de volta**



- Home
- Sobre
- Contato
- Referências

**deMenezes**

## Every e some: os métodos de array em Javascript que respondem "sim" ou "não"



**Front-end**

### O método find e a busca ao modo CTRL + F em JavaScript

**deMenezes**

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)



**Front-end**

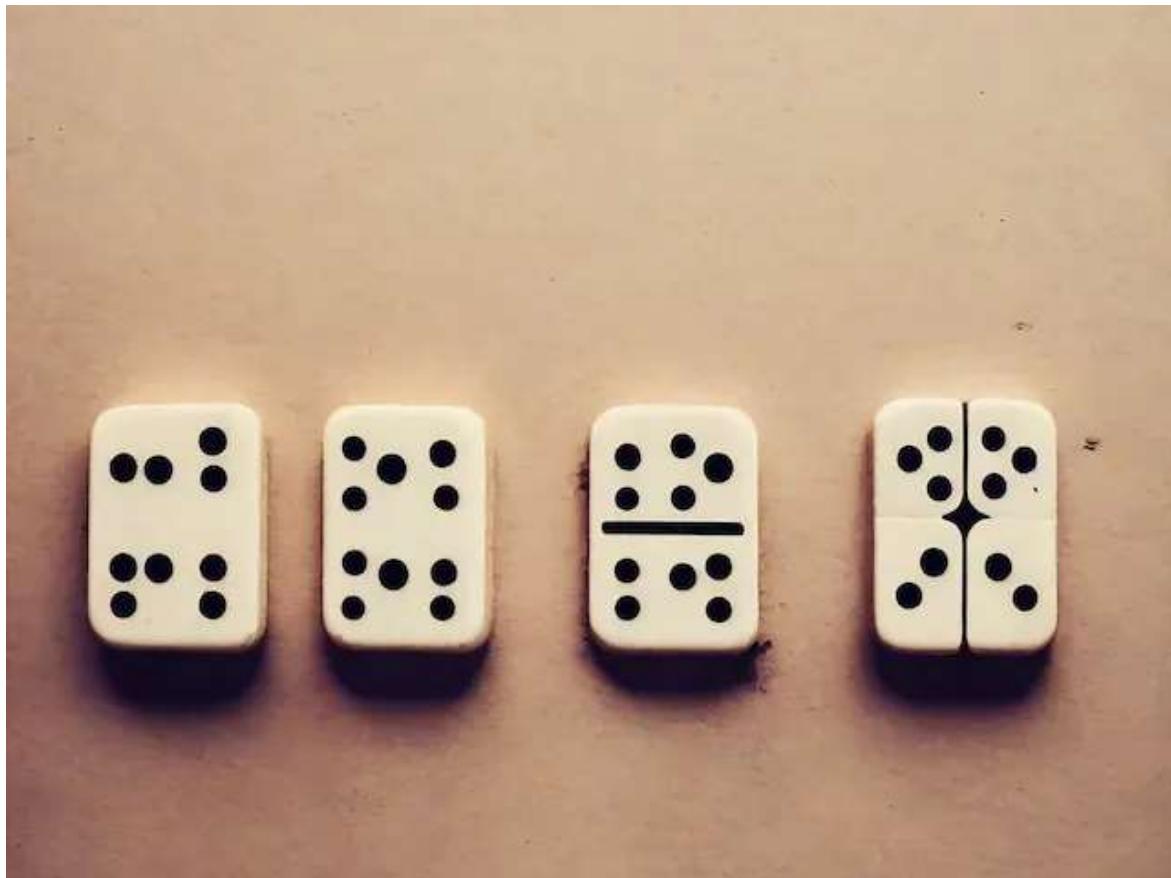
**Domine o método Filter e simplifique sua lógica  
com arrays em Javascript**



**deMenezes**

- Home
- Sobre
- Contato
- Referências

## Transforme arrays em ouro: aprenda a manipular dados com o map() em Javascript

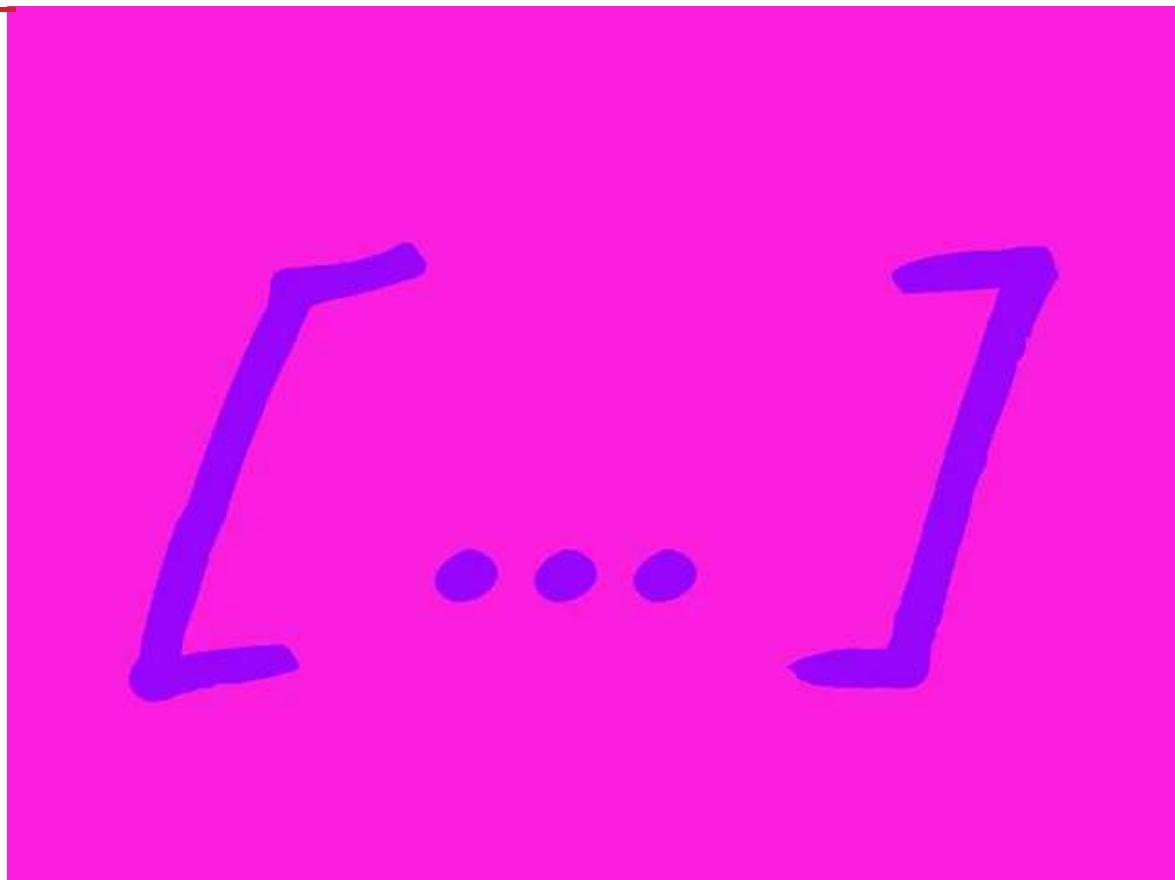


Front-end

## Como usar o forEach em Javascript: exemplos detalhados contados em uma história

deMenezes

- [Home](#)
- [Sobre](#)
- [Contato](#)
- [Referências](#)



Front-end

O que diferencia array, NodeList, HTMLCollection e outros iteráveis em Javascript

```
(function (text) {  
    console.log(text)  
    // resultado: 'front-end'  
} )('front-end' )
```

- Home
- Sobre
- Contato
- Referências

deMenezes

## Como usar uma IIFE (função autoinvocada) para criar um código modular e sem conflitos de escopo



Front-end

## Imagens quadradas e redondas em CSS: o guia que eu queria ter lido



deMenezes

- Home
- Sobre
- Contato
- Referências

## Front-end

### Guia básico para adicionar eventos em Javascript e despertar a interatividade



## Front-end

### Atributo HTML contenteditable: o que é e como debugar um site com ele

- Home
- Sobre
- Contato
- Referências



Front-end

## O perigo da propriedade CSS `list-style-type: none` para listas acessíveis na web



deMenezes

- Home
- Sobre
- Contato
- Referências

## Boas práticas para evitar problemas com margin collapsing em CSS



Front-end

`event.target` vs `event.currentTarget`: qual usar?



**deMenezes**

- Home
- Sobre
- Contato
- Referências

## Front-end

### 7 maneiras de selecionar elementos com Javascript sem querySelectorAll, querySelector e getElementById



## Front-end

### 6 motivos para o focus não funcionar (com e sem Javascript)

**deMenezes**

- Home
- Sobre
- Contato
- Referências



Front-end

Como trabalhar com links e botões em HTML da forma correta, com exemplos



deMenezes

- Home
- Sobre
- Contato
- Referências

## O mínimo sobre focus de elementos em HTML, CSS e Javascript (bônus: tabindex)



Front-end

Como fazer um lazyload compatível com qualquer browser + Placeholder

deMenezes

- Home
- Sobre
- Contato
- Referências



**Front-end**

**Talvez você não deva usar o atributo  
loading="lazy"**

- github
- linkedin
- twitter
- medium
- dev.to

**deMenezes**