

Cascading Style Sheets

Seletores, Formulários, Tabelas etc...

Mais sobre pseudoclasses e pseudo-elementos

Arquivos de apoio:

seletor1.html

seletor2.html

seletor3.html

seletor4.html

seletor5.html

:focus

:checked

:first-child

:last-child

:nth-child(n)

::before

::after

::first-letter

— — —

Sintaxe ligeiramente diferente

:pseudoclasse

::pseudo-elemento

Note que há uma pequena diferença entre a sintaxe das pseudoclasses (:) e dos pseudo-elementos (::). Nas versões anteriores (CSS1 e CSS2) ambos utilizavam apenas dois pontos. O CSS3 optou em diferenciá-los através dessa nova sintaxe.

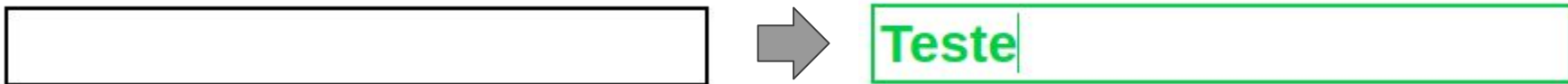
Vamos ver alguns casos. Para ver outros casos, acesse:

https://www.w3schools.com/cssref/css_selectors.asp

Pseudoclasses *focus* e *checked* (seletor1.html)

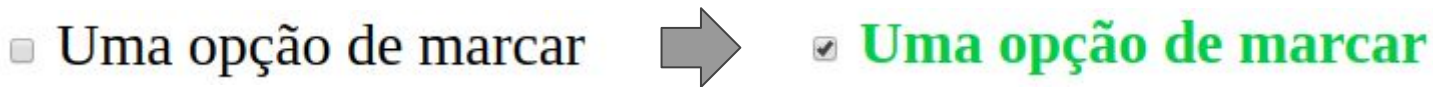
:focus

Geralmente utilizado para estilizar elementos de formulário que estiverem em foco no navegador. Melhora usabilidade pois mostra exatamente o campo em que o usuário se encontra.



:checked

Geralmente utilizado para estilizar formulários, aplicando estilos em componentes cujo usuário precisa marcar uma opção (checkbox, por exemplo).



Pseudoclasses *first-child* e *last-child* (*seletor2.html*)

Útil para estilizar o primeiro e último elementos de uma tag pai que possua vários elementos (tabelas, listas etc). **Atenção:** deve ser aplicado aos elementos filhos, não aos elementos pais (por exemplo, ao , não ao).

:first-child	Aponta para o primeiro elemento de um conjunto de elementos.
:last-child	Aponta para o último elemento de um conjunto de elementos.

✓ Primeiro elemento

✓ Segundo elemento

✓ Terceiro elemento

✓ Quarto elemento

✓ Quinto elemento

Pseudoclasse *nth-child()* ([seletor3.html](#))

A pseudoclasse *nth-child()* é útil para aplicar estilos em elementos específicos de uma lista de elementos (por exemplo, linhas ou colunas de tabelas).

even	Utilizado para estilizar apenas as linhas pares.
odd	Utilizado para estilizar apenas as linhas ímpares.
an+b	Onde a indica de quantas em quantas linhas será realizado o salto e o valor de b é a linha inicial.

[illegible]

Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3
Coluna 1	Coluna 2	Coluna 3

[illegible]

Pseudo-elemento *first-letter* (*seletor4.html*)

::first-letter

Através deste pseudo-elemento é possível aplicar um estilo para a primeira letra de um conteúdo de texto. Veja o exemplo a seguir.

T

odo documento HTML possui marcadores, palavras entre parênteses angulares (sinais de menor e maior). Esses marcadores são os comandos de formatação da linguagem. Um elemento é formado por um nome de marcador (tag), atributos, valores e filhos (que podem ser outros elementos ou texto). Os atributos modificam os resultados padrões dos elementos e os valores caracterizam essa mudança.

Pseudo-elementos *before* e *after* (*seletor5.html*)

::before - Utilizado para definir um conteúdo **antes** do elemento.

::after - Utilizado para definir um conteúdo **depois** do elemento.

Em ambos os casos, deve-se obrigatoriamente utilizar o atributo **content** para indicar o conteúdo a ser exibido. Este atributo recebe um texto ou uma imagem. Se precisarmos uma estilização diferente, deve-se deixá-lo com um texto vazio.

content: ";



Formulários

Arquivos de apoio:
form1.html
form2.html

`<form>`

`<input>`

`<button>`

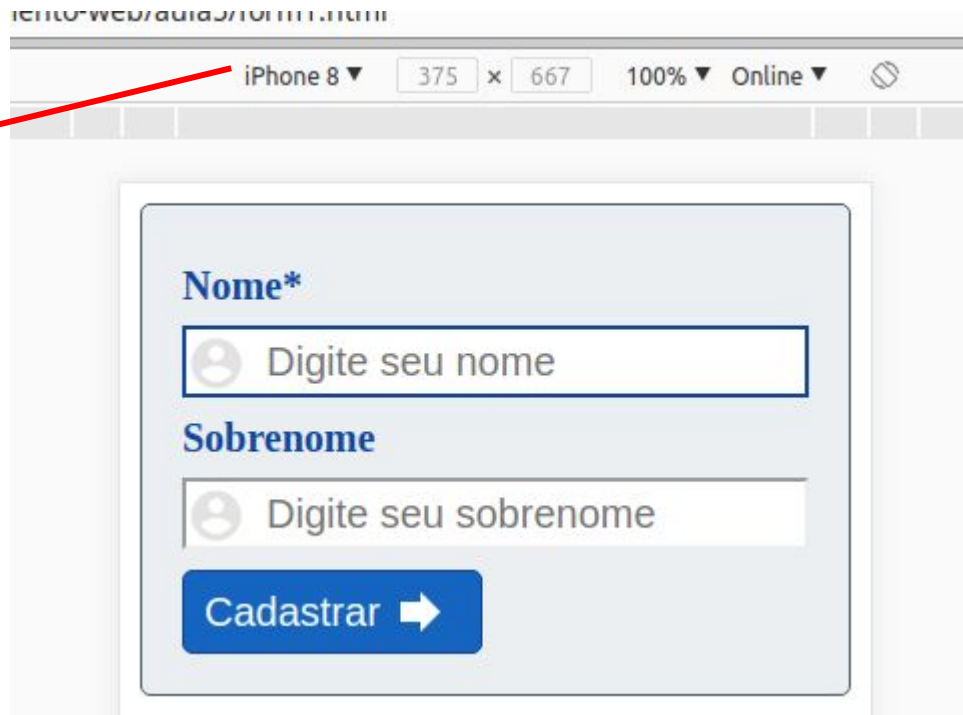
`<select>`

`<textarea>`

— — —

Vamos começar montando um formulário simples (*form1.html*)

Simulando um iPhone 8 no DevTools do Google Chrome



<form> : a tag <form> padrão ocupará 100% da tela e possui comportamento *block*. É uma boa prática mobile permitir que ela ocupe 100% da viewport, porém fica mais agradável trabalhar com propriedades como *padding* e *margin*.

Vamos começar montando um formulário simples (*form1.html*)

<label> : Sempre use tags <label> para associar textos aos campos de entrada (tag <input>, por exemplo) da página. O atributo *for* do <label> deve apontar para o *id* do <input>.

```
<label  
for="nome">Nome</label>  
<input type="text" id="nome"  
>
```

Você pode estilizá-lo como qualquer outra tag, porém lembre-se que ele possui um comportamento *inline* (não pega *width*, *height*, entre outros, por padrão).

```
label {  
    font-size: 20px;  
    line-height: 40px;  
    width: 150px;  
    color: #0d47a1;  
    font-weight: bold;  
    margin-top: 5px;  
}
```

Vamos começar montando um formulário simples (*form1.html*)

<input> : Há várias formas legais de estilizar um <input> do tipo texto. Vejamos algumas dicas:

- Apesar de *type="text"* ser default no <input>, podemos defini-lo explicitamente para trabalhar com seletores por atributo.

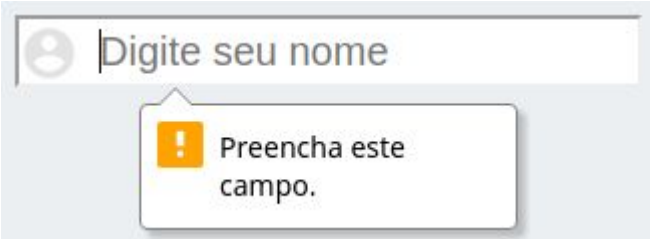
`input[type=text] {}`

- Visando trabalhar com *mobile-first* (técnica de estilizar pensando primeiramente em mobile), é comum definir o *width* do <input> como 100% para forçá-lo a pular para a próxima linha (ele tem comportamento *inline*). Neste caso, como geralmente estilizamos com *padding*, *border* e outros efeitos, deve-se sempre trabalhar com:

`box-sizing: border-box;`

Vamos começar montando um formulário simples (*form1.html*)

<input> : Use e abuse de atributos como:

placeholder	Define um texto no fundo do <input> que auxilia no preenchimento do campo. Quando o usuário começa a digitar, este texto é sobrescrito.
autofocus	Se definido, o <i>focus</i> será automaticamente aplicado ao campo quando a página for carregada.
required	<p>Diz que o campo é obrigatório e já mostra uma mensagem (no idioma do navegador do usuário) informando quando o campo não estiver preenchido.</p>  <p>Cuidado com a compatibilidade desta mensagem no IE: https://www.w3schools.com/tags/att_input_required.asp</p>

Vamos começar montando um formulário simples (*form1.html*)

:focus : podemos trabalhar com a pseudoclassee *focus* para estilizar o <input> que estiver com o focus.

```
input[type=text]:focus {}
```

A screenshot of a web form input field. The field is a rectangular box with a thin blue border. Inside the box, on the left, is a small, light gray circular icon containing a person silhouette. To the right of the icon, the text "Digite seu nome" is written in a light gray font. The entire input field is highlighted with a thick blue border, indicating it is the active element with focus.

Um último recurso interessante é atribuir uma imagem de fundo. Se distanciarmos o conteúdo do <input> com um *padding-left*, podemos aplicar um pequeno ícone para melhorar a aparência do formulário.

```
background-image: url(img/nome.png);
```

```
background-size: 30px;
```

```
background-repeat: no-repeat;
```

```
background-position-y: 2px;
```

```
padding-left: 40px;
```

A screenshot of a web form input field. The field is a rectangular box with a thin gray border. Inside the box, on the left, is a small, light gray circular icon containing a person silhouette. To the right of the icon, the text "Digite seu sobrenome" is written in a light gray font. The input field has a light gray background and is not highlighted with a thick border, indicating it is not the active element.

<input type="submit"> ou <button>? (*form1.html*)

Há duas formas de criar botões de submissão no HTML:

```
<input type="submit" value="Cadastrar" />
```

Tag tradicional que deve ter seu valor definido no atributo *value*. Vantagem: se houver algum estilo aplicado à tag geral <input>, esta tag também será estilizada.

```
<button type="submit">Cadastrar</button>
```

Tag que possui elementos de abertura e fechamento segregados, sendo que o valor é mantido entre elas. Vantagens: fácil de estilizar com ícones e possui uma semântica melhor que a versão com <input>.



Outras tags de formulário (*form2.html*)

<select> : útil para selecionar valores a partir de uma lista. Se for utilizado *required*, o primeiro **<option>** deve ter *value=""*.

<textarea> : útil para digitar texto com várias linhas. **Atenção:** Neste exemplo, foi definido para entrada do endereço do usuário. Na prática, os campos associados ao endereço devem ser digitados separadamente (logradouro, cidade, CEP etc).

Nome*

Sobrenome

Área*

Endereço*

☐ Masculino ☐ Feminino *

☐ Estudante?

Limpar 

Cadastrar 

Outras tags de formulário (*form2.html*)

<input type="radio"> : componentes do tipo *radio* devem ter o campo *name* definido com o mesmo valor para que eles possam ser mutuamente exclusivos.

<input type="checkbox"> : útil para valores booleanos. Não faz sentido colocar campos desse tipo como *required*.



Nome*

Digite seu nome

Sobrenome

Digite seu sobrenome

Área*

Selecione

Endereço*

Digite seu endereço

☐ Masculino ☐ Feminino *

☐ Estudante?

Limpar

Cadastrar

Dicas gerais (*form2.html*)

- **Estilos:** as tags <label>, <input>, <button>, <form>, etc, podem ser estilizadas como qualquer outra tag HTML.
- **Atributo *name*:** para fins de envio do formulário ao back-end, deve-se obrigatoriamente definir o atributo *name*.
- **Posicionamento e cores para os botões:** é muito comum manter cores e posicionamento específicos para tipos de botões. Botões de cancelamento ou limpeza de formulário devem ser colocados à esquerda, com cores claras. Botões primários, que servem para enviar conteúdo do formulário, devem ser posicionados à direita, com cores mais escuras.
- **Sempre coloque * nos campos obrigatórios**
- **Use divs com 100% para organizar o layout, se necessário**

Tabelas

Arquivos de apoio:
table1.html

<table>

<thead>

<tbody>

<tfoot>

<tr>

<th>

<td>

— — —

Aplicando estilos em tabelas (*table1.html*)

ID	Produto	Quantidade	Preço
#1	Notebook Gamer Acer	5	R\$ 3.500,00
#2	Notebook Samsung	3	R\$ 2.800,00
#3	Notebook Asus	10	R\$ 4.100,00
#4	Notebook Apple	2	R\$ 6.000,00
#5	Notebook Dell	14	R\$ 3.000,00
#6	Notebook HP	0	R\$ 1.900,00
#7	Notebook Toshiba	1	R\$ 1.500,00
#8	Notebook Lenovo	5	R\$ 3.000,00
#9	Notebook Dell Inspiron	2	R\$ 2.300,00
Total			R\$ 45.000,00

Tags <thead>, <tbody> e <tfoot>

Além das tags tradicionais (<table>, <tr> e <td>), existem as tags <thead>, <tbody> e <tfoot> para flexibilizar estilos específicos em tabelas e ao mesmo tempo aplicar semântica.

<thead> : aplicada ao cabeçalho da tabela.

<tbody> : aplicada ao corpo da tabela.

<tfoot> : aplicada ao rodapé da tabela.

```
<table>
  <thead>
    <tr>
      <th>...</th><th>...</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>...</td><td>...</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>...</td><td>...</td>
    </tr>
  </tfoot>
</table>
```

Outros recursos (*table1.html*)

É comum usar seletores universais para indicar filhos de forma independente. Por exemplo, para aplicar *border* e *padding* a todos `<td>`, independente de estarem na `<thead>`, `<tbody>` ou `<tfoot>`, deve-se utilizar:

```
table tr * {  
    padding: 4px 8px;  
    border: 1px solid #333333;  
}
```

Também é comum o uso de pseudoclasses:

<code>first-child</code>	Aponta para o primeiro filho (linha ou coluna).
<code>last-child</code>	Aponta para o último filho (linha ou coluna).

CSS Avançado

Arquivos de apoio:

[transition.html](#)

[transform.html](#)

[animation.html](#)

transition

transform

animation

— — —

Transição entre os estilos (*transition.html*)

Serve para aplicar tempos de transição entre os efeitos. Por exemplo, um link no qual o mouse hover aumenta o tamanho da tag <a>. Poderíamos aplicar um tempo máximo para a transação através da propriedade *transition*. Veja alguns exemplos de uso:

```
transition: 2s;
```

É possível separar por vírgula tempos diferentes por propriedade CSS e definir um delay para começar a transição.

```
transition: width 2s, line-height 2s, background-color 500ms;
```

```
transition-delay: 1s, 2s, 0s;
```

Mais detalhes: https://www.w3schools.com/css/css3_transitions.asp

Transformando elementos (*transform.html*)

Através da propriedade *transform* é possível rotacionar e ajustar o tamanho dos elementos.

```
transform: scale(1.5) rotate(-45deg);
```

Especificação 2D:

https://www.w3schools.com/css/css3_2dtransforms.asp

Especificação 3D:

https://www.w3schools.com/css/css3_3dtransforms.asp

Animação (*animation.html*)

A propriedade *animation* permite aplicar efeitos de animação automáticos.

Passo 1: definir uma *@keyframes* para especificar a animação. Para cada bloco, pode-se trabalhar com percentuais ou com as palavras chaves *from* e *to*, indicando o estilo inicial e o estilo final, respectivamente.

```
0% { opacity: 0; }
```

```
100% { opacity: 1; }
```

```
from { transform: rotate(0deg); }
```

```
to { transform: rotate(360deg); }
```


Passo 2: definir a propriedade *animation* no elemento indicando o nome da *keyframes* criada e tempo da animação (pode ser infinito).

```
animation: meu-h1 3s;
```

Animação (*animation-v2.html*)

Atenção: essas propriedades foram suportadas pelos navegadores mais recentes. Para garantir a compatibilidade, deve-se trabalhar com os prefixos dos navegadores conforme orientação da W3C:

https://www.w3schools.com/css/css3_animations.asp

Property					
@keyframes	43.0 4.0 -webkit-	10.0	16.0 5.0 -moz-	9.0 4.0 -webkit-	30.0 15.0 -webkit- 12.0 -o-
animation	43.0 4.0 -webkit-	10.0	16.0 5.0 -moz-	9.0 4.0 -webkit-	30.0 15.0 -webkit- 12.0 -o-

Desenhos Vetoriais Avançados

Arquivos de apoio:
[canvas.html](#)

<canvas>

— — —

O poderoso <canvas> (*canvas.html*)

Através da tag <canvas> é possível desenhar elementos geográficos e imagens no corpo do HTML de forma estática ou dinâmica. Exige uso da linguagem JavaScript e é muito utilizado para criar gráficos, jogos, e outros recursos mais sofisticados.

Não vamos explorar todo o poder do <canvas> neste curso. Para maiores detalhes, veja a documentação oficial em:

https://www.w3schools.com/html/html5_canvas.asp

Exemplo 2D:
canvas.html

Exemplo 3D:

https://threejs.org/examples/?q=canvas#canvas_geometry_birds

Obrigado!