

CSS



Artist Image : Descourtilz, Jean-Théodore | Dates:179?-1855

Guia iniciante: CSS

ABOUT ME



Brunna Croches

Developer Full Stack

Brunna Croches é Dev FullStack, advogada e empreendedora.

Apaixonada por tech, vem adquirido vasto conhecimento na área.

Desenvolveu projetos ricos em diversidade, buscando captar as próximas tendências e necessidades do mercado.

Neste e-book você aprenderá ou recapitulará de forma simplificada e otimizados conceitos de programação feito por ela.

let's share

SUMMARY



COMOCENTRALIZAR DIVS EM HTML E CSS

17

Centralizando divs

PSEUDO CLASSES

18

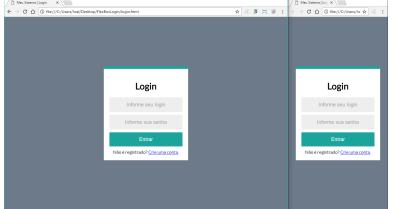
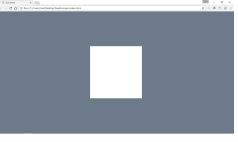
Interagindo a página com o usuário.

MÉDIA QUERIES

19

Aplicando diferentes estilos e adequando a forma do conteúdo.

17- Como centralizar divs em HTML e CSS

<h3>Centralizando uma DIV</h3>	<p>Aqui, trabalharemos basicamente sobre o elemento que atuará como container, de forma que as divs ou outros componentes internos sejam alinhados ao centro.</p>	
<h3>HTML</h3>	<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Document</title> </head> <body> <div class="container"> <div class="box"> </div> </div> </body> </html></pre>	<p>Nessa página, a div com classe container é o elemento principal do nosso layout, pois conterá as formatações via CSS para centralizar o seu conteúdo (nesse caso, a div com classe box, que representará o formulário de login visto anteriormente).</p>
<h3>CSS</h3>	<pre>01 <style> 02 .container { 03 width: 100vw; 04 height: 100vh; 05 background: #6C7A89; 06 display: flex; 07 flex-direction: row; 08 justify-content: center; 09 align-items: center 10} 11 .box { 12 width: 300px; 13 height: 300px; 14 background: #fff; 15} 16 body { 17 margin: 0px; 18} 19 </style></pre>	<ul style="list-style-type: none">• Linhas 3 e 4: definimos as dimensões do container como 100% da largura (width) e altura (height) da página. Para isso, utilizamos as unidades vw e vh das CSS3, que representam, respectivamente, a largura e a altura da viewport;• Linha 5: alteramos a cor do plano de fundo da div container, para melhor simular o resultado visto na Figura 1;• Linha 6: definimos a propriedade display do container como flex. Essa é a configuração que faz com que, de fato, a div utilize o recurso de layout flexível (Flexbox);• Linha 7: com a propriedade flex-direction, fazemos com que os itens internos sejam dispostos horizontalmente, ou seja, em forma de linha (row);• Linha 8: configuramos a disposição dos elementos internos como centralizados na direção definida na propriedade anterior, ou seja, os itens ficarão no centro da linha (horizontalmente);• Linha 9: com a propriedade align-items definida como center, fazemos com que os elementos internos sejam também alinhados na vertical;• Linhas 11 a 15: configuramos a div box com o plano de fundo branco e dimensões fixas.• Linhas 16 a 18: removemos as margens do corpo do documento, fazendo assim com que a div container ocupe toda a página.
<h3>Resultado</h3>		<p>O mesmo resultado visual poderia ser obtido se definíssemos a propriedade flex-direction como column. Nesse caso, seria preciso configurar a disposição dos elementos internos para ocorrer na direção vertical,</p>

enquanto a propriedade align-items passaria
a atuar na direção horizontal.

18 - Pseudo Classes e Pseudo Elementos

Nesta documentação aplicaremos estilo baseado no estado do elemento com as pseudo classes do CSS.

Pseudo Classes	As pseudo-classes são um recurso poderoso, que quando utilizadas em conjunto com os outros recursos do CSS, como combinadores, nos permitem criar diversos efeitos.	
:visited	Lembre-se da seguinte regra: <code>a:visited{color:purple;}</code> Dessa forma, após o link ter sido clicado, a cor de fundo aplicada pelo navegador seria purple e não vermelho, como é feito na maioria dos casos.	<code>a:visited{color:purple;}</code>
:active	para trabalhar com links. :active é o estado do link quando este está ativo, mas ainda não foi clicado, geralmente quando chegamos até ele utilizando a tecla tab do teclado.	<code>a:active{color:green;}a:hover{color:cyan;}</code>
:hover	é o estado de um elemento quando o ponteiro do mouse está sobre ele. Vale notar que :hover pode ser aplicado a qualquer elemento, embora esse comportamento possa não ser válido em todos os navegadores.	<code>a:active{color:green;}a:hover{color:cyan;}</code>

Pseudo classes do CSS

Nesta documentação aplicaremos estilo baseado no estado do elemento com as pseudo classes do CSS.

:first-of-type	Com a pseudo-classe <code>:first-of-type</code> podemos selecionar o primeiro elemento, dentro de um determinado elemento pai.	<div> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p> <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p> <p>Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.</p> </div>	Podemos alterar o estilo do primeiro dentre eles colocando-o em itálico da seguinte forma: <code>p:first-of-type{font-style: italic;}</code>
:last-of-type	Se desejamos selecionar o último parágrafo na div em lugar do primeiro, podemos utilizar a pseudo-classe :last-of-type da seguinte forma	" "	<code>p:last-of-type{font-style: italic;}</code> Nesse caso, apenas o último parágrafo na div será selecionado.

Pseudo Classes do CSS

A pseudo-classe :nth-of-type **corresponde a um elemento em uma determinada posição, desde que o mesmo seja de um certo tipo.**

SELECIONAR Apenas o primeiro parágrafo em cada elemento sera selecionado :nth-of-type	Vejamos um exemplo: <code>p:nth-of-type(1){color:blue;}</code> <code>p:nth-of-type(1) = apenas o</code> <code>primeiro parágrafo em cada</code> <code>elemento pai será</code> <code>selecionado.</code>	No exemplo ao lado, apenas o primeiro parágrafo em cada elemento pai será selecionado. Podemos utilizar a pseudo-classe :nth-of-type passando diferentes tipos de argumentos, de forma idêntica a que fazemos com :nth-child.
SELECIONAR Apenas os parágrafos que correspondem a posições do elemento pai :nth-of-type	Abaixo, vemos um novo exemplo no qual selecionamos apenas os parágrafos que correspondam a posições pares dentro do elemento pai: <code>p:nth-of-type(2n){color:blue;}</code> <code>p:nth-of-type(2n)=</code> selecionamos apenas os parágrafos que correspondam a posições pares dentro do elemento pai	
SELECIONAR Inverter a ordem na qual as posições serão consideradas :nth-last-of-type (2n+1)	Para inverter a ordem na qual as posições serão consideradas, mantendo o comportamento dessa pseudo-classe, podemos utilizar :nth-last-of-type da seguinte forma: <code>p:nth-last-of-type(2n+1)</code> <code>{color:blue;}</code>	Nesse caso, apenas os elementos ímpares serão selecionados, contando de baixo para cima.

:first-child	:first-child seleciona o primeiro elemento filho em um elemento pai. Essa pseudo-classe não recebe parâmetros e pode ser utilizada da seguinte forma: <code>span:first-child{color:gray;}</code> apenas o primeiro span em um parágrafo, div ou qualquer outro elemento pai será selecionado.
:last-child	Podemos inverter a ordem em que o elemento será selecionado de forma simples, alterando a pseudo-classe para last-child <code>span:last-child{color:gray;}</code> Desta forma, o último elemento filho em um elemento pai qualquer que corresponda ao tipo span será selecionado.

Impares e Primos

	A primeira regra da família :nth que veremos é a :nth-	um exemplo com palavras-chave:
--	--	--------------------------------

:nth-child	child, que corresponde a um elemento que está em uma posição específica em um grupo de elementos primos.	[code]li:nth-child(odd) { background-color: gray; }[/code] os elementos nas posições ímpares, 1, 3, 5 e assim por diante, serão selecionados. Se utilizássemos even no lugar de odd, os elementos nas posições pares, 2, 4, 6 e assim por diante, seriam selecionados.
Na forma funcional :nth-child	Na forma funcional, passamos como parâmetro para :nth-child uma fórmula $a + b$, onde a e b são números inteiros.	Vejamos um exemplo: [code]li:nth-child(5n) { background-color: gray; }[/code] Os elementos cujas posições forem múltiplos de 5 terão suas cores de fundo alteradas para cinza.
:nth-last-child	A pseudo-classe :nth-last-child() seleciona um elemento baseado em sua posição, contando a partir do último elemento encontrado.	Na prática, utilizamos essa pseudo-classe da mesma forma que :nth-child(), porém contanto os itens de trás para frente.

:not	A pseudo-classe :not seleciona um elemento que não corresponda ao seletor passado como argumento.	Sua sintaxe é a seguinte: seletor:not(seletor)
Exemplo :not	Para selecionar todos os parágrafos que não possuam a classe .resumo podemos escrever uma regra como esta ao lado	p:not(.resumo){color:gray;}
Exemplo todos os paragrafos exceto p.resumo receberão cor cinza	<pre><!DOCTYPE html> <html> <head> <title></title> <link rel="stylesheet" type="text/css" href="stylesheet.css"> </head> <body> <div> <h1>Cork returns to where it started for Milestone match</h1> <p>Burnley midfielder makes 200th PL appearance at Craven Cottage, having made his debut at the same ground</p> <p>There was symmetry for Jack Cork as he celebrated his 200th appearance in the Premier League with a Milestone award.</p> <p>The Burnley midfielder reached the landmark in the 4-2 defeat at Fulham last weekend.</p></pre>	<pre>p { font: 14px/22px "Helvetica Neue", arial, sans-serif; } p:nth-of-type(2n+1) { background: #eee; } tr:nth-of-type(2n+1) { background: #eee; } input:disabled, input:read-only { background: #eee; border: 1px solid #222; cursor: not-allowed; } input:checked + label { color: green; } input:disabled {</pre>

```

<p>Coincidentally, the 29-year-old made his Premier League debut for Burnley at Craven Cottage in February 2010, while on loan from Chelsea.</p>

<p>Since then he has reached the double century thanks also to spells at Southampton and Swansea City.</p>
</div>

<table>
  <tbody>
    <tr>
      <td>LEI</td>
      <td>0</td>
      <td>x</td>
      <td>2</td>
      <td>LIV</td>
    </tr>
    <tr>
      <td>BHA</td>
      <td>1</td>
      <td>x</td>
      <td>0</td>
      <td>FUL</td>
    </tr>
    <tr>
      <td>CHE</td>
      <td>1</td>
      <td>x</td>
      <td>0</td>
      <td>BOU</td>
    </tr>
    <tr>
      <td>CRY</td>
      <td>3</td>
      <td>x</td>
      <td>0</td>
      <td>SOU</td>
    </tr>
    <tr>
      <td>EVE</td>
      <td>0</td>
      <td>x</td>
      <td>1</td>
      <td>HUD</td>
    </tr>
  </tbody>
</table>
</body>
</html>

```

```

border: 1px solid red;
}

input:not(:disabled) {
  border: 1px solid green;
}

input:last-of-type {
  font-style: italic;
}

p:nth-child(even) {
  background: #eee;
}

p:nth-last-child(odd) {
  background: #eee;
}

p:nth-last-of-type(2n+1) {
  background: #eee;
}

p:nth-last-of-type(-n+2) {
  background: #eee;
}

```

Pseudo Elementos

Pseudo Elemento

Os pseudo-elementos nos permitem **selecionar algumas áreas internas de um elemento HTML e customizá-las através de propriedades**. Selecionar a área que antecede o conteúdo de um elemento adicionando a ela texto ou imagens, ou estilizar apenas a primeira linha de um elemento de texto são apenas algumas das possibilidades criadas por esse recurso.

<h2>Sintaxe</h2>	<p>Para utilizar um pseudo-elemento basta declarar o seletor seguido pelo pseudo-elemento desejado. Logo após, da mesma forma que é feito com qualquer regra CSS, podemos definir as propriedades a ele relacionadas.</p>	<p>Sintaxe de declaração de um pseudo-elemento:</p> <pre>[seletor]::pseudo-elemento { propriedade : valor; }</pre>
<h2>::first-line</h2>	<p>O pseudo-elemento ::first-line é utilizado quando desejamos customizar o conteúdo apresentado na primeira linha de um elemento. Assim, podemos estilizar esse fragmento de texto de forma independente, aplicando, por exemplo, um tamanho de fonte diferente.</p>	<pre>p::first-line { font-size: 16px; }</pre> <p>Nota: É importante ressaltar que não há uma maneira trivial de definir até onde o texto será estilizado, principalmente quando lidamos com páginas responsivas. Ao redimensionar uma página, a primeira linha pode se tornar maior ou menor, e isso também depende de outros fatores, como o tamanho da fonte, espaçamento entre palavras, entre outros.</p>
<h2>::first-letter</h2>	<p>O pseudo-elemento ::first-letter nos permite selecionar a primeira letra do conteúdo de um elemento. Usamos esse recurso quando desejamos destacar essa letra alterando seu tamanho e sua cor, por exemplo.</p>	<pre><p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. </p></pre> <pre>p::first-letter { color: red; font-size: 32px; }</pre>
<h2>::before</h2>	<p>A declaração do pseudo-elemento ::before em um seletor cria um falso elemento que nos permite adicionar conteúdo antes do conteúdo do elemento selecionado.</p>	<pre><div> Texto dentro do elemento. </div></pre> <pre>div::before { content: "pseudoelemento ::before"; color: blue; }</pre> <p>Nesse exemplo o texto “pseudoelemento ::before” será apresentado antes do conteúdo “Texto dentro do elemento” e na cor azul. Para isso declaramos as propriedades content e color como regras do pseudo-elemento.</p>
<h2>::after</h2>	<p>Assim como ::before, o pseudo-elemento ::after também cria um falso elemento que nos permite adicionar conteúdo ao elemento selecionado, mas desta vez no final dele.</p>	<pre><div> Texto dentro do elemento. </div></pre> <pre>div::after { content: "pseudoelemento ::after"; color: red; }</pre> <p>Nesse caso, todo texto declarado nos elementos <p> de uma página HTML terá a cor vermelha.</p>
<h2>::selection</h2>	<p>O pseudo-elemento ::selection é utilizado para selecionar o conteúdo de um elemento e, a partir disso, customizar algumas propriedades dessa área, como a cor de fundo e do texto.</p>	<pre>p::selection { color: red; }</pre> <p>Nesse caso, todo texto declarado nos elementos <p> de uma página HTML terá a cor vermelha.</p> <p>Nota: É importante ressaltar que apenas um pequeno conjunto de propriedades do CSS pode ser utilizado com esse pseudo-elemento. Entre essas propriedades podemos destacar: color, background-color, cursor e text-decoration. Em alguns navegadores ainda podemos utilizar: outline e text-shadow. Quando</p>

Exemplo Prático

uma propriedade não puder ser renderizada ela será ignorada.

```
<blockquote>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod  
tempor incididunt ut labore et dolore magna aliqua.  

```

```
blockquote {  
    display: inline;  
}  
  
blockquote::before {  

```

19- Media Queries

Queries	<p>Isso permite, por exemplo, a aplicação de estilos diferentes para telas maiores (desktops) e menores (tablets e smartphones), adequando a forma como o conteúdo é exibido para o usuário, a fim de oferecer sempre a melhor experiência.</p>	
Como era	<pre>@media [media type] { estilos }</pre>	<p>Para o argumento [media type], os valores suportados são: screen, para visualização em telas quaisquer; print, para telas de impressão; speech para leitores de tela; ou all, para todos os tipos.</p>
Usando o media para alterar a cor do texto	<p>uso do @media que altera a cor do texto dependendo do tipo de mídia. Dessa forma, quando a página for acessada de um browser, visualizaremos a cor azul; já em modo de impressão, visualizaremos a cor vermelha.</p> <pre>@media screen { body { color: blue; } } @media print { body { color: red; } }</pre>	 <p>Enquanto na parte superior vemos o texto na cor padrão ao acessar a página pelo browser, na parte inferior vemos o mesmo texto em vermelho, por utilizarmos o comando de impressão.</p>
Sintaxe	<pre>@media [not only] [media type] and ([query]) { estilos }</pre>	
Sintaxe not e only	<p>Os operadores not e only permitem negar um tipo de mídia, ou especificar que apenas determinado tipo seja atendido, respectivamente.</p>	<p>Por exemplo, a expressão not speech faria com que as regras não fossem aplicadas a leitores de tela, enquanto que only screen faria com que apenas telas fossem atendidas.</p>
Sintaxe and	<p>Por sua vez, o operador and faz a ligação entre o tipo de mídia e o filtro adicional que será especificado entre parênteses, no lugar de query.</p>	
Exemplo	<p>Com esse código, aplicamos o plano de fundo azul para páginas com até 640px de largura e vermelho para aquelas com até 480px de largura. Acima de 640px, o comportamento padrão será assumido, ou seja, será mantida a página branca</p> <pre>@media screen and (max-width: 640px){ body { background-color: blue; } }</pre>	

	<pre> } @media screen and (max-width: 480px){ body { background-color: red; } } </pre>	
Conclusão	Entre as várias propriedades que podem ser usadas, as mais comuns são: min-width , min-height , max-width , max-height , width e height .	A partir delas podemos, por exemplo, adaptar as páginas para apresentar o comportamento responsivo, ajustando-as adequadamente conforme os diferentes tamanhos de tela.

O que é?

O que é?	criar páginas responsivas, usando apenas HTML5 e CSS3 sem ter de recorrer a frameworks de componentes.	
Motivo	Quando precisamos misturar os componentes de duas bibliotecas diferentes numa só aplicação para atender a demanda, aumentando consideravelmente a quantidade de código e arquivos (muitos deles sequer usados).	Uma solução mais adequada seria a criação da sua própria biblioteca de componentes responsivos, com seu próprio estilo. Assim, evitamos o excesso de implementação e, acima de tudo, temos o total controle de todo o website.
Objetivo	construir seus frameworks web	
A qualidade de um site	a responsividade é o conceito que define a qualidade de um site, aliada a conceitos como confiabilidade, segurança, performance e preocupação dos criadores para com seu público.	
Mobile Friendly	O próprio Google, bem como outras ferramentas de busca na web, ranqueiam melhor sites que apresentam seu conteúdo de forma <i>mobile friendly</i> , isto é, preparam seus conteúdos para serem exibidos em dispositivos de diversos tamanhos, a incluir smartphones, tablets, notebooks, etc.	

Sass

	O Sass permite que elementos sejam aninhados para flexibilizar a forma como as regras serão geradas no CSS final.	
	Veja como o Sass simplifica a criação de regras de estilo ao evitar que tenhamos de duplicar código sempre que quisermos uma regra em herança.	<pre> // Antes .barra-navegacao { display: flex; li { padding: 5px 10px; } } // Depois .barra-navegacao { display: flex; } .barra-navegacao li { padding: 5px 10px; } </pre>

Estrutura de diretórios
do projeto.

```
---responsive-web
+---- html
+---- img
+---- js
+---- style
    +---- CSS
    +---- SCSS
```

CONTACT



Brunna Croches

Developer Full Stack



brunnacroches.dev



linkedin.com/brunnacroches



github.com/brunnacroches



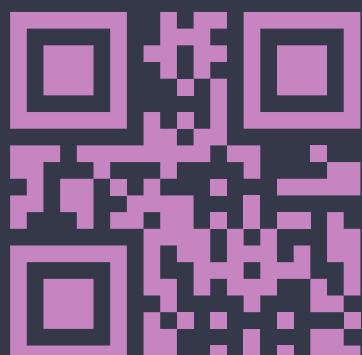
@brunnacroches.dev



discord.com/brunnacroches



brunnacroches@gmail.com



let's share