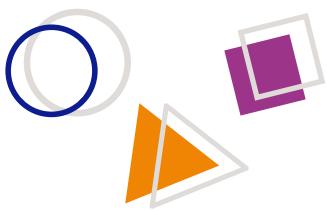




Mundo Tech



PROGRAMAÇÃO FRONT-END

SENAI <LAB365>

SUMÁRIO

Introdução a programação	13
Posicionamento, visibilidade e transparência.....	13
Posicionamento.....	13
Visibilidade	19
Transparência	20
Sombras, gradientes e transformações.....	21
Sombras.....	21
Gradientes.....	23
Transformações.....	24
Box model.....	25
Espaçamento (padding).....	25
Borda (border)	26
Margem (margin).....	29
Propriedade “box-sizing”	30
Propriedade “display”	30
Viewport e media query.....	31
Viewport	31
Media queries	32
Flexbox.....	34
Alinhamento de linhas ou colunas	35
Alinhamento vertical e horizontal.....	37
Referências.....	41



MAS, ANTES DE COMEÇAR...

Para resolver seu desafio, você poderá utilizar qualquer imagem da internet e uma logo com a qual você mais se identifique. Para procurar imagens, acesse o site a seguir ou aproxime seu celular do QR code:

<https://www.istockphoto.com/br>



Para começar, aqui está uma parte do código-fonte. Com isso, basta você compreender o código e finalizá-lo para resolver seu desafio. Confira.

```
/* Usado para importar regras de estilo de outras folhas de estilo. */

@import"https://fonts.googleapis.com/
css2?family=Source+Sans+Pro:wght@200;300;400&display=swap";

* { /* Este seletor escolhe todos os elementos do documento e redefine
todas as configurações do navegador. */

    margin: 0; padding: 0;

    border: 0;

    font-size: 100%;

    font-family: "Source Sans Pro", sans-serif;

    font-weight: 300;

    vertical-align: baseline;

    text-decoration: none;

    list-style: none;

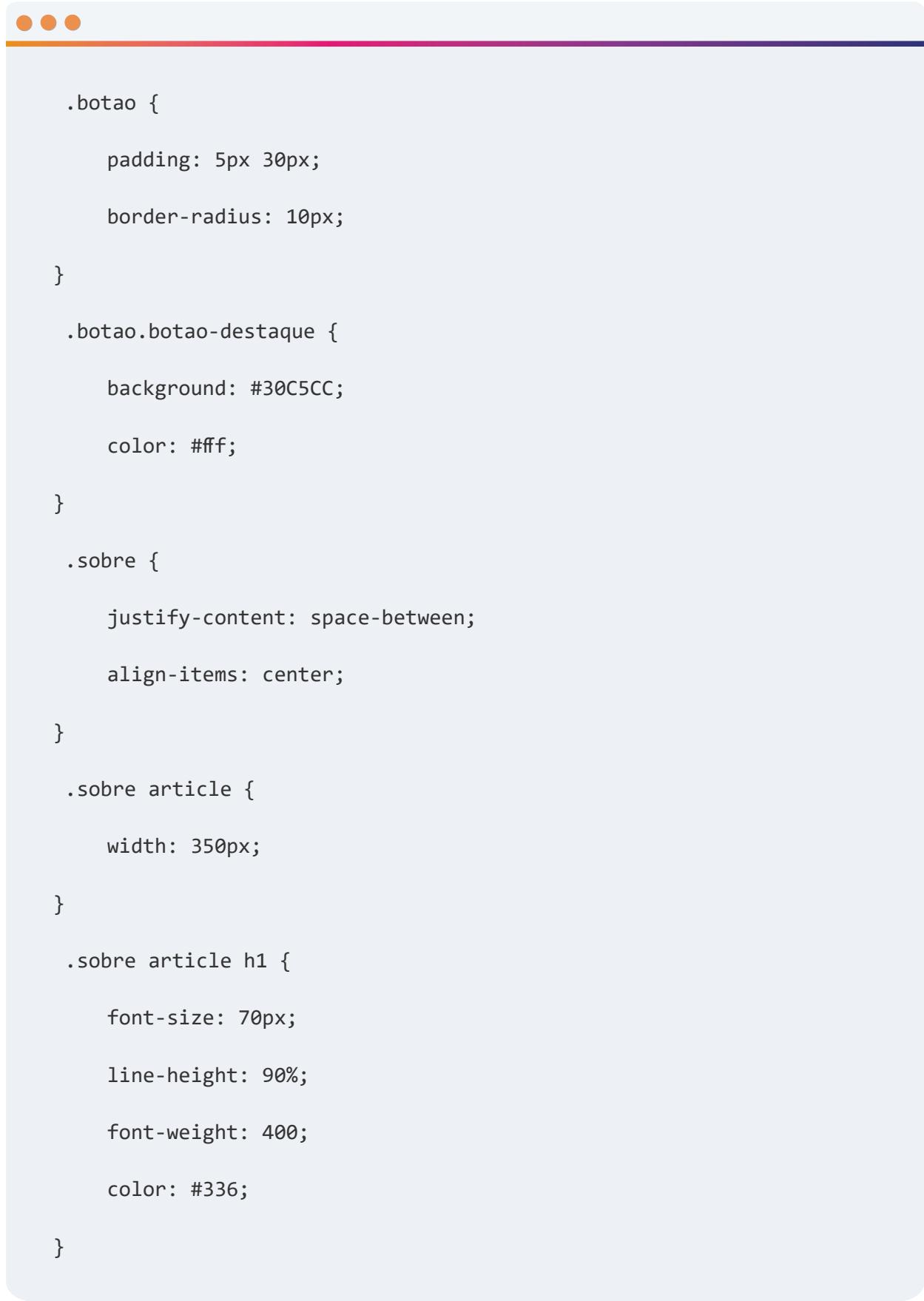
}

.conteudo {

    width: 1024px;

    margin: 30px auto;
```

```
display: flex;  
  
flex-direction: column;  
  
}  
  
.linha {  
  
display: flex;  
  
flex-direction: row;  
  
}  
  
header section {  
  
justify-content: space-between;  
  
align-items: center;  
  
}  
  
header nav ul {  
  
display: flex;  
  
flex-direction: row;  
  
align-items: center;  
  
}  
  
header nav ul li a {  
  
margin: 0px 15px;  
  
color: #39c;  
  
}
```



The screenshot shows a Mac OS X window with three orange title bar buttons. The main content area contains a block of CSS code:

```
.botao {  
    padding: 5px 30px;  
    border-radius: 10px;  
}  
  
.botao.botao-destaque {  
    background: #30C5CC;  
    color: #fff;  
}  
  
.sobre {  
    justify-content: space-between;  
    align-items: center;  
}  
  
.sobre article {  
    width: 350px;  
}  
  
.sobre article h1 {  
    font-size: 70px;  
    line-height: 90%;  
    font-weight: 400;  
    color: #336;  
}
```

```
.sobre article h2,  
  
.sobre article p {  
  
    color: #39c;  
  
}  
  
.sobre article h2 {  
  
    font-size: 30px;  
  
    text-transform: uppercase;  
  
    letter-spacing: 2px;  
  
    line-height: 200%;  
  
}  
  
.sobre article p {  
  
    margin: 10px 0 40px;  
  
}  
  
.sobre article figcaption img {  
  
    width: 700px;  
  
}  
  
footer ul li {  
  
    margin: 0 30px 0 0;  
  
}  
  
footer ul li a img {  
  
    width: 30px;  
  
}
```

```
/* Define o tamanho dos elementos em telas de 1150px. */

@media(max-width: 1150px) {

    .conteudo {

        width: 950px;

    }

}

/* Define o tamanho dos elementos em telas de 1024px. */

@media(max-width: 1024px) {

    .conteudo {

        width: 850px;

    }

    .sobre figcaption img {

        width: 600px;

    }

}

/* Define o tamanho dos elementos em telas de 915px. */

@media(max-width: 915px) {

    .conteudo {

        width: 600px;

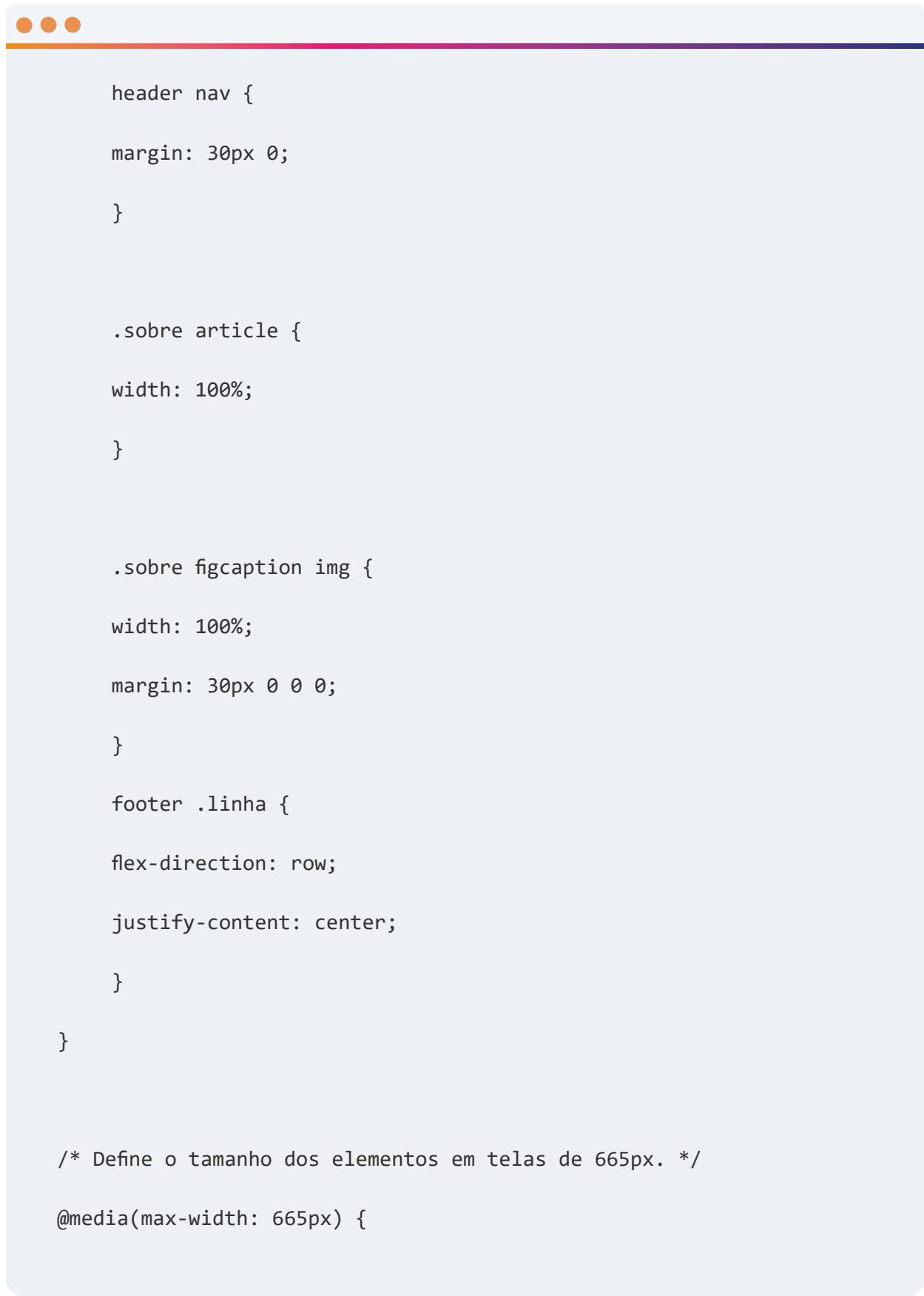
    }

    .linha {

        flex-direction: column;

    }

}
```



```
header nav {  
    margin: 30px 0;  
}  
  
.sobre article {  
    width: 100%;  
}  
  
.sobre figcaption img {  
    width: 100%;  
    margin: 30px 0 0 0;  
}  
  
footer .linha {  
    flex-direction: row;  
    justify-content: center;  
}  
  
/* Define o tamanho dos elementos em telas de 665px. */  
@media(max-width: 665px) {
```

```
.conteudo {  
  
    width: 400px;  
  
}  
  
}  
  
/* Define o tamanho dos elementos em telas de 450px. */  
  
@media(max-width: 450px) {  
  
    .conteudo {  
  
        width: 290px;  
  
    }  
  
  
  
header nav {  
  
    width: 100%;  
  
    padding: 10px 0 20px;  
  
}  
  
header nav ul {  
  
    flex-direction: column;  
  
}  
  
  
  
header nav ul li {  
  
    margin: 10px 0;  
  
}  
  
* {  
  
    box-sizing: border-box;
```

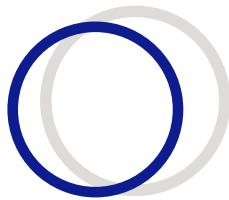
```
margin: 0;  
  
padding: 0;  
  
}  
  
}  
  
/* Itens do formulário */  
  
fieldset {  
  
width: 60%;  
  
padding: 10px 0 20px;  
  
margin-left: 10px;  
  
border: 2px solid lightblue;  
  
border-radius: 10px;  
  
padding: 20px;  
  
}  
  
.agrupamento {  
  
display: flex;  
  
flex-direction: column; flex-direction: column;  
  
}
```

```
.agrupamento-botao {  
    padding-top: 10px;  
}  
  
.entrada {  
    font-family: inherit;  
    line-height: 1.15;  
    border: 2px solid lightblue;  
    border-radius: 10px;  
    padding-bottom: 10px;  
    padding-left: 15px;  
    padding-right: 15px;  
    padding-top: 10px;  
}  
  
.entrada:focus {  
    outline: none;  
    box-shadow: 0 0 0 0.2rem rgb(173, 216, 230, 0.5);  
}  
  
.entrada.mensagem {  
    resize: none;  
}
```

```
.rotulo {  
  
    padding-bottom: 2.5px;  
  
    padding-top: 20px;  
  
    font-weight: 600;  
  
}
```

Lembre-se de que você já tem os arquivos de início e o contato do estudo anterior.
Então, pode usar os mesmos!

Boa sorte com seu último desafio!



INTRODUÇÃO A PROGRAMAÇÃO

Em continuidade aos seus estudos, agora você confere alguns tópicos acerca do CSS, responsáveis por incluir recursos mais avançados em uma página web. Na sequência, haverá uma introdução aos recursos de posicionamento, visibilidade e transparência. Você terá contato com recursos de personalização avançada envolvendo sombras, gradientes e transformações. Conhecerá os fundamentos do box model. Posteriormente, você fará a utilização de viewport e media query. E, por fim, entenderá como realizar a organização de elementos utilizando flexbox.

Posicionamento, visibilidade e transparência

A propriedade “position” pode ajudá-lo a manipular a localização de um elemento. A propriedade “visibility” em CSS tem duas funções diferentes. Ela oculta linhas e colunas de uma tabela e também oculta um elemento sem alterar o layout. Já a propriedade “opacity” é responsável por permitir a alteração da opacidade de um elemento. Por padrão, todos os elementos têm o valor de 1. Alterando esse valor para mais próximo de 0, o elemento ficará cada vez mais transparente. Veja na sequência suas principais definições.

Posicionamento

O posicionamento é o que nos faz determinar onde os elementos aparecem na tela e como eles aparecem. Você pode mover os elementos e posicioná-los exatamente onde quiser. Nesta seção, também veremos como as coisas mudam em uma página com base em como os elementos com diferentes posições interagem entre si. Temos uma propriedade CSS principal: “position”. Essa propriedade pode ter esses valores: “static”, “relative”, “absolute” e “fixed”.

Posicionamento estático

Esse é o valor padrão para um elemento. Elementos estáticos posicionados são exibidos no fluxo de página normal.

Posicionamento relativo

Se você definir a posição de um elemento como “relative”, poderá posicioná-lo com um deslocamento, usando as propriedades: “top”, “right”, “bottom” e “left”, que são chamadas de propriedades de deslocamento. Ele aceita um valor de comprimento ou uma porcentagem.

Entenda o exemplo seguinte. Nele são criados um contêiner pai, um contêiner filho e uma caixa interna com a palavra “Testando”.

```
Untitled - Captain Anonymous
```

HTML

```
1 <div class="pai">
2   <div class="filho">
3     <div class="caixa">
4       <p>Testando</p>
5     </div>
6   </div>
7 </div>
```

CSS 100+ unsaved changes

```
1 .pai{
2   background-color: #8aff00;
3   padding: 30px;
4   width: 300px;
5 }
6
7 .filho{
8   background-color: #f30472;
9   padding: 30px;
10}
11
12 .caixa{
13   background-color: #42124c;
14   padding: 30px;
15   border: 2px solid #ffffff;
16   border-style: dotted;
17   text-align: center;
18   font-size: 20px;
19   color: #ffffff;
20 }
```

Figura 1 - Exemplo de um contêiner

Fonte: do Autor (2022)

Você pode observar que o CSS contém algumas instruções para mudar cores e preenchimento, mas não há nenhuma regra que afeta o posicionamento. Agora, se definirmos “position: relative” para o seletor “.caixa”, a princípio, aparentemente, nada muda. Mas, o elemento agora é capaz de se mover usando as propriedades “top”, “right”, “bottom” e “left”, e agora é possível alterar a posição da caixa em relação ao elemento que o contém. Por exemplo:

The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML:

```
<div class="pai">
  <div class="filho">
    <div class="caixa">
      <p>Testando</p>
    </div>
  </div>
</div>
```

CSS:

```
.pai{
  background-color: #8aff00;
  padding: 30px;
  width: 300px;
}

.filho{
  background-color: #f30472;
  padding: 30px;
}

.caixa{
  background-color: #42124c;
  padding: 30px;
  border: 2px solid #ffffff;
  border-style: dotted;
  text-align: center;
  font-size: 20px;
  color: #ffffff;
  position: relative;
  top: -60px;
}
```

The preview window shows a dark purple box with the text "Testando" inside, centered on a red rectangular background, which is itself centered on a green square background.

Figura 2 - Movimentação da caixa utilizando o posicionamento relativo

Fonte: do Autor (2022)

Posicionamento absoluto

A configuração “position: absolute” removerá um elemento do fluxo do documento. Com o posicionamento absoluto, assim que definimos “position: absolute” no seletor “.caixa”, seu espaço original agora é recolhido e apenas a origem (coordenadas x, y) permanece a mesma. Agora podemos mover a caixa como quisermos, usando as propriedades “top”, “right”, “bottom” e “left”. Confira um exemplo:

The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML:

```
1 <div class="pai">
2   <div class="filho">
3     <div class="caixa">
4       <p>Testando</p>
5     </div>
6   </div>
7 </div>
```

CSS:

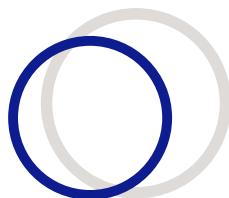
```
1 .pai{
2   background-color: #8aff00;
3   padding: 30px;
4   width: 300px;
5 }
6
7 .filho{
8   background-color: #f30472;
9   padding: 30px;
10 }
11
12 .caixa{
13   background-color: #42124c;
14   padding: 30px;
15   border: 2px solid #ffffff;
16   border-style: dotted;
17   text-align: center;
18   font-size: 20px;
19   color: #ffffff;
20   position: absolute;
21   top: 130px;
22   left: 60px;
23 }
```

The preview window on the right displays three nested divs. The outermost div ("pai") has a yellow-green background and a width of 300px. The middle div ("filho") has a dark red background and a padding of 30px. The innermost div ("caixa") has a dark purple background, a padding of 30px, and a dotted border. It contains the text "Testando" and is positioned absolutely at top: 130px and left: 60px relative to its parent.

Figura 3 - Movimentação da caixa utilizando o posicionamento absoluto

Fonte: do Autor (2022)

Isso significa que, se adicionarmos “position: relative” ao elemento “.filho” e definirmos “top” e “left” para 0 no seletor “.caixa”, a caixa não será posicionada na margem superior esquerda da janela, mas será posicionada nas coordenadas 0, 0 de “.filho”:



The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML:

```
<div class="pai">
  <div class="filho">
    <div class="caixa">
      <p>Testando</p>
    </div>
  </div>
</div>
```

CSS:

```
.pai{
  background-color: #8aff00;
  padding: 30px;
  width: 300px;
}

.filho{
  background-color: #f30472;
  padding: 30px;
  position: relative;
}

.caixa{
  background-color: #42124c;
  padding: 30px;
  border: 2px solid #ffffff;
  border-style: dotted;
  text-align: center;
  font-size: 20px;
  color: #ffffff;
  position: absolute;
  top: 0px;
  left: 0px;
}
```

The preview area on the right displays a layout with three main sections: a green header, a pink middle section, and a purple bottom section. The purple section contains the text "Testando". A dotted border surrounds the purple section, indicating its absolute position relative to the page's top-left corner.

Figura 4 - Movimentação da caixa utilizando o posicionamento absoluto com coordenadas 0, 0

Fonte: do Autor (2022)

Posicionamento fixo

Assim como no posicionamento absoluto, quando um elemento é atribuído como “position: fixed”, ele é removido do fluxo da página. A diferença com o posicionamento absoluto é esta: os elementos agora são sempre posicionados em relação à janela, em vez do primeiro contêiner não estático.



Dica

Outra grande diferença é que os elementos não são afetados pela rolagem. Depois de colocar um elemento fixo em algum lugar, rolar a página não o remove da parte visível do site.



```
Untitled ↗
Captain Anonymous
```

HTML

```
1 <div class="pai">
2   <div class="filho">
3     <div class="caixa">
4       <p>Testando</p>
5     </div>
6   </div>
7 </div>
```

CSS 100+ unsaved changes X

```
1 .pai{
2   background-color: #8aff00;
3   padding: 30px;
4   width: 300px;
5 }
6
7 .filho{
8   background-color: #f30472;
9   padding: 30px;
10 }
11
12 .caixa{
13   background-color: #42124c;
14   padding: 30px;
15   border: 2px solid #ffffff;
16   border-style: dotted;
17   text-align: center;
18   font-size: 20px;
19   color: #ffffff;
20   position: fixed;
21 }
```

Figura 5 - Movimentação da caixa utilizando o posicionamento fixo

Fonte: do Autor (2022)



Visibilidade

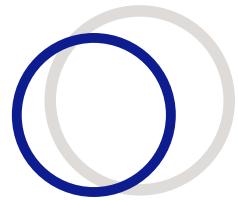
Você pode usar a propriedade “visibility” para controlar se um elemento é visível ou não. Essa propriedade pode ter um dos seguintes valores listados na tabela abaixo (CSS..., 2021; REFERÊNCIA..., [s.d.]):

Valor	Descrição
visible	Valor padrão. A caixa e seu conteúdo são visíveis.
hidden	A caixa e seu conteúdo são invisíveis, mas ainda afetam o layout da página.
collapse	Esse valor faz com que toda a linha ou coluna seja removida da exibição. Esse valor é usado para elementos de linha, grupo de linhas, coluna e grupo de colunas.
inherit	Especifica que o valor da propriedade de visibilidade deve ser herdado do elemento pai, ou seja, assume o mesmo valor de visibilidade especificado para seu pai.

Quadro 1 - Visibilidade

Fonte: do Autor (2022)

Confira um exemplo:



```
Untitled
Captain Anonymous

HTML
1 <h1>Testando a visibilidade</h1>
2 <h2 class="visivel">Este título está visível...</h2>
3 <h2 class="invisivel">Este título está invisível...</h2>
4 <p>Observe que o título oculto ainda ocupa espaço na página.</p>

CSS
81 unsaved changes

1 h2.visivel {
2   visibility: visible;
3 }
4
5 h2.invisivel {
6   visibility: hidden;
7 }
```

Testando a visibilidade

Este título está visível...

Observe que o título oculto ainda ocupa espaço na página.

Figura 6 - Trabalhando com visibilidade e invisibilidade de textos

Fonte: do Autor (2022)

Transparência

A opacidade e seu oposto, a transparência, podem ser usadas no design do site para criar contraste e reforçar a identidade de um elemento. Para definir a opacidade de um plano de fundo, imagem, texto ou outro elemento, você pode usar a propriedade de opacidade do CSS. Os valores dessa propriedade variam de 0 a 1. Se você definir a propriedade como 0, o elemento com estilo será completamente transparente (ou seja, invisível). Se você definir a propriedade como 1, o elemento será completamente opaco.

Confira um exemplo do mesmo elemento `<div>` que tem um estilo diferente com a propriedade “`opacity`”. A primeira `<div>` está definida para ser completamente opaca. A última `<div>` está definida para ser completamente transparente, e é por isso que você não pode vê-la. As demais divisões entre elas são definidas para serem vários graus de transparência.

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
1 <div class="container">
2   <div class="primeiro">Opacidade 1</div>
3   <div class="segundo">Opacidade 0.9</div>
4   <div class="terceiro">Opacidade 0.6</div>
5   <div class="quarto">Opacidade 0.3</div>
6   <div class="quinto">Opacidade 0</div>
7 </div>
```

The 'CSS' tab contains the following code:

```
7 background: #de7a78;
8 opacity: 1;
9 }
10 .segundo {
11   width: 100px;
12   height: 100px;
13   background: #de7a78;
14   opacity: 0.9;
15 }
16 .terceiro {
17   width: 100px;
18   height: 100px;
19   background: #de7a78;
20   opacity: 0.6;
21 }
22 .quarto {
23   width: 100px;
24   height: 100px;
25   background: #de7a78;
26   opacity: 0.3;
27 }
28 .quinto {
29   width: 100px;
30   height: 100px;
31   background: #de7a78;
32   opacity: 0;
33 }
```

Below the code editor, there is a preview window showing five red squares of decreasing opacity from left to right, labeled 'Opacidade 1' through 'Opacidade 0.3'.

Figura 7 - Trabalhando com transparência entre caixas de texto

Fonte: do Autor (2022)

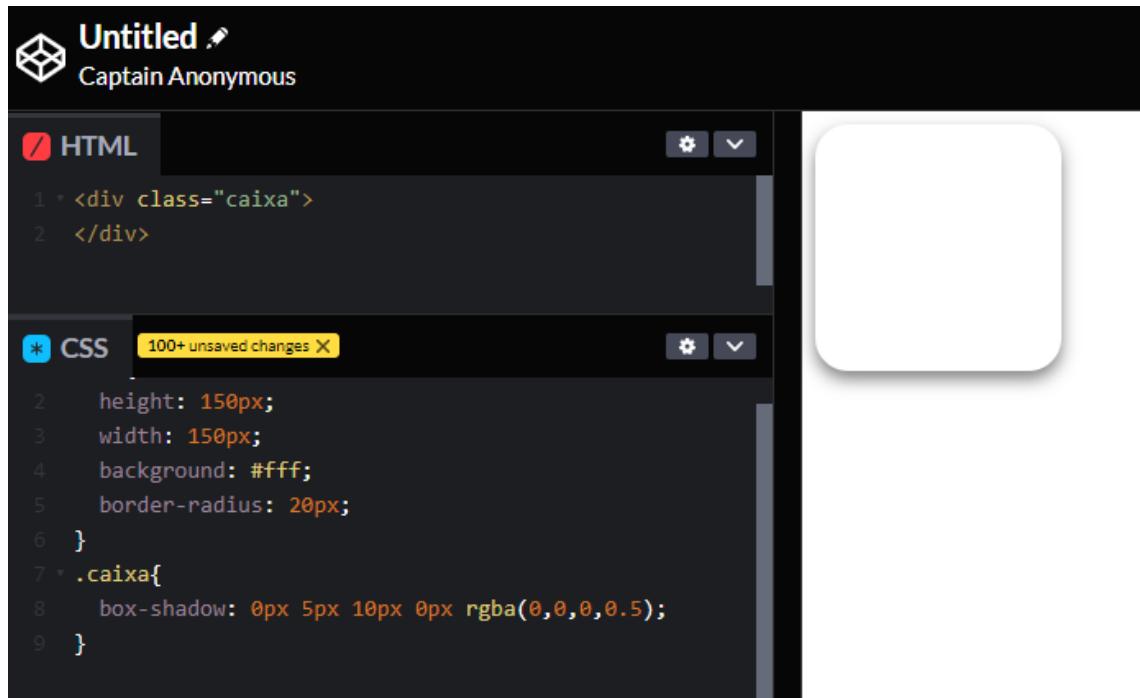
Sombras, gradientes e transformações

Hoje em dia, não é suficiente que um site faça seu trabalho. Ele tem que levar o usuário em uma jornada, uma jornada esteticamente agradável de matrizes, fontes, sombras e tudo mais. Os sites precisam parecer realistas, e as sombras, gradientes e transformações desempenham um papel importante. Veremos como utilizar tais recursos do CSS nas seções seguintes.

Sombras

Nesta seção, você aprende mais sobre a propriedade “box-shadow” e como pode utilizá-la. Essa propriedade permite adicionar uma sombra ao redor de um elemento em uma página da web. Ela pode nos dizer se um elemento como um botão, item de navegação ou um cartão de texto é interativo.

Nossos olhos estão acostumados a ver sombras. Elas dão uma ideia do tamanho e da profundidade de um objeto e trazem esse realismo para nossa experiência on-line. Quando estilizadas corretamente, podem melhorar a estética da página da web. Para começar, primeiro vamos criar um contêiner de caixa simples com HTML. Confira um exemplo:



The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

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

The 'CSS' tab contains the following code:

```
height: 150px;
width: 150px;
background: #fff;
border-radius: 20px;
}
.caixa{
  box-shadow: 0px 5px 10px 0px rgba(0,0,0,0.5);
}
```

To the right of the editor, there is a preview window showing a white rounded rectangle with a soft shadow effect.

Figura 8 - Contêiner contendo uma sombra

Fonte: do Autor (2022)

Na declaração da propriedade “box-shadow” há quatro parâmetros. Vamos ver o que significa cada um deles. O primeiro parâmetro representa a posição horizontal da sombra (valor x-offset). O segundo parâmetro representa a posição da sombra verticalmente (valor y-offset). O terceiro parâmetro determina o raio de desfoque, que afeta a nitidez da sombra. Valores mais altos significam sombras mais claras e vice-versa. O quarto valor define o espalhamento (spread) ou a extensão da dimensão da sombra. O quinto valor define a cor.

O valor de propagação definido em 0px fará com que a sombra tenha o mesmo tamanho da caixa. Um valor positivo aumentará seu tamanho e um valor negativo, diminuirá.

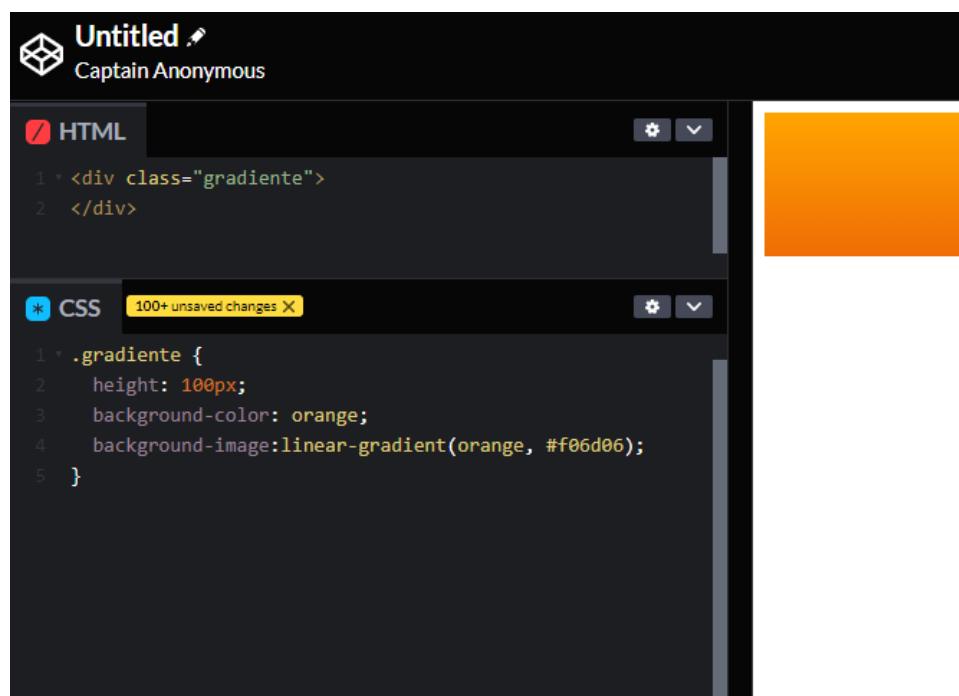
Como descrito, quando definimos a cor, podemos especificar a opacidade, que é um aspecto importante a ser considerado ao estilizar sombras realistas. As sombras em espaços bem iluminados não são pretas ou completamente opacas, você pode ver a cor da área em que a sombra é projetada. Ao estilizar a propriedade, lembre-se de que as sombras transparentes são as melhores, porque ficam ótimas em fundos multicoloridos.

Gradientes

Assim como você pode declarar o plano de fundo de um elemento como uma cor sólida em CSS, também pode declarar esse plano de fundo como um gradiente. Usar gradientes declarados em CSS, em vez de usar um arquivo de imagem real, é melhor para controle e desempenho.

Os gradientes geralmente são uma cor que se transforma em outra, mas no CSS você pode controlar todos os aspectos de como isso acontece, desde a direção das cores (quantas você quiser) até onde essas mudanças de cores acontecem.

Ao declarar a propriedade “background-color” de uso de uma cor sólida em CSS, os gradientes usam “background-image”. Talvez o tipo de gradiente mais comum e útil seja o “linear-gradient()”. O “eixo” de gradientes pode ir da esquerda para a direita, de cima para baixo ou em qualquer ângulo que você escolher. Confira um exemplo.



The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML Tab:

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

CSS Tab:

```
.gradiente {
  height: 100px;
  background-color: orange;
  background-image: linear-gradient(orange, #f06d06);
}
```

To the right of the editor, there is a preview window showing a vertical gradient from orange at the top to a lighter shade (#f06d06) at the bottom, matching the colors defined in the CSS.

Figura 9 - Contêiner contendo um gradiente

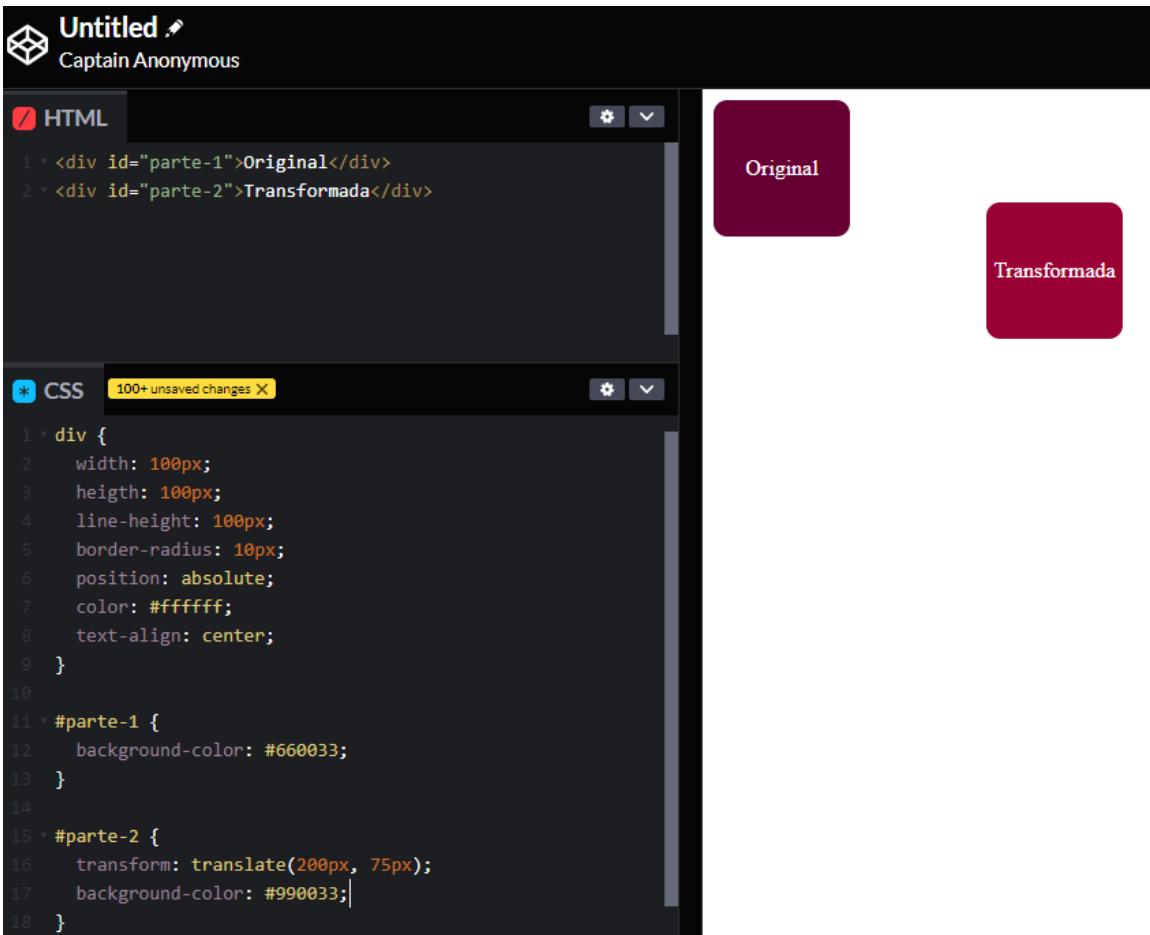
Fonte: do Autor (2022)

Transformações

As transformações permitem traduzir, girar, dimensionar e inclinar elementos no espaço 2D ou 3D. Elas são um recurso CSS muito legal, especialmente quando combinadas com animações. A propriedade “transform” aceita essas funções (CSS..., 2021; REFERÊNCIA..., [s.d.]):

- › translate(): para mover elementos ao redor;
- › rotate(): para girar elementos;
- › scale(): para dimensionar elementos em tamanho;
- › skew(): para torcer ou inclinar um elemento;
- › matrix(): para executar qualquer uma das operações acima usando uma matriz de seis elementos.

Confira a seguir, um exemplo de transformação utilizando a função “translate()”. Nesse exemplo, podemos movimentar uma tag <div> por um número de pixels:



The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML Tab:

```
<div id="parte-1">Original</div>
<div id="parte-2">Transformada</div>
```

CSS Tab:

```
* div {
    width: 100px;
    height: 100px;
    line-height: 100px;
    border-radius: 10px;
    position: absolute;
    color: #ffffff;
    text-align: center;
}

#parte-1 {
    background-color: #660033;
}

#parte-2 {
    transform: translate(200px, 75px);
    background-color: #990033;
}
```

On the right side of the editor, there are two rounded square boxes. The top one is dark purple and contains the text "Original". The bottom one is dark red and contains the text "Transformada".

Figura 10 - Transformação utilizando a função “translate()”

Fonte: do Autor (2022)

O método “translate()”, como vimos, traduz, ou move, um elemento de página para cima, para baixo, para a esquerda ou para a direita na página a partir de um valor especificado. Entre parênteses, o primeiro número especifica a distância horizontal e o segundo número especifica a distância vertical.

Box model

Cada elemento CSS é essencialmente uma caixa. O modelo de caixa explica o dimensionamento dos elementos com base em algumas propriedades CSS. De dentro para fora, temos: área de conteúdo, espaçamento, borda e margem. A melhor forma de visualizar o box model é a partir da representação da imagem seguinte:

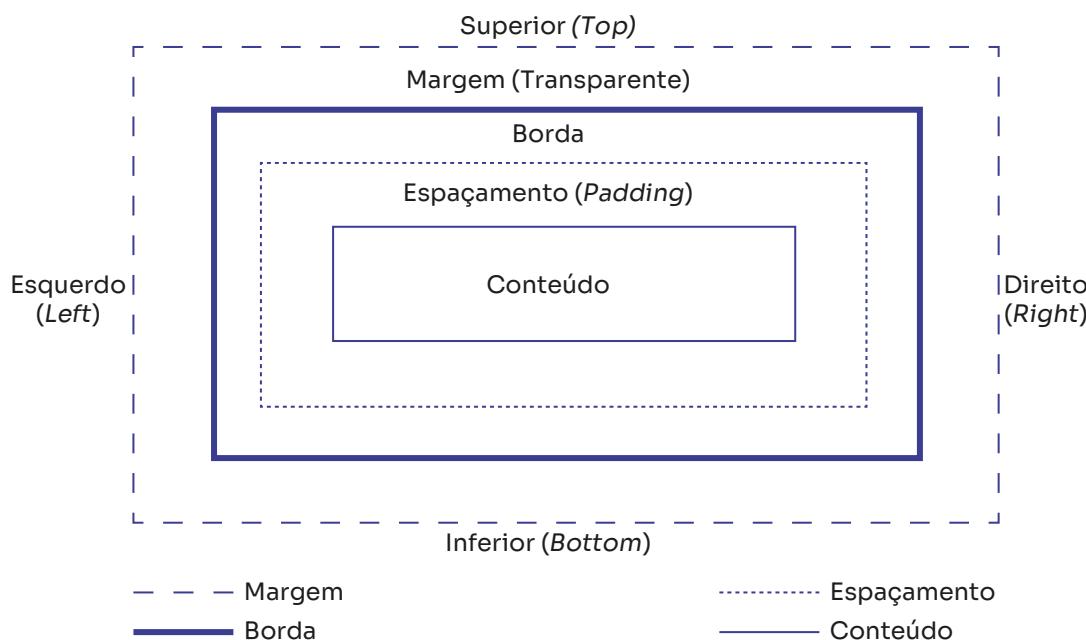


Figura 11 - A estrutura do box model

Fonte: W3 ([s.d.])

Espaçamento (padding)

A propriedade “padding” é comumente usada em CSS para adicionar espaço no lado interno de um elemento. Lembre-se: “margin” adiciona espaço fora de uma borda de elemento e “padding” adiciona espaço dentro de uma borda de elemento.

A propriedade “padding” tem quatro propriedades relacionadas, que alteram o preenchimento de uma única aresta de uma só vez: “padding-top”, “padding-right”, “padding-bottom” e “padding-left”.

Já “padding” é um atalho para especificar vários valores de preenchimento ao mesmo tempo e, dependendo do número de valores inseridos, ele se comporta de maneira diferente. confira:

Quantidade de valores	Descrição	Exemplo
1 valor	Aplica-se a todos os preenchimentos: superior, direito, inferior e esquerdo.	padding: 20px;
2 valores	Aplica o primeiro às partes inferior e superior e o segundo à esquerda e à direita.	padding: 20px 10px;
3 valores	Aplica o primeiro ao topo, o segundo à esquerda e à direita e o terceiro ao fundo.	padding: 20px 10px 30px;
4 valores	Aplica o primeiro para cima, o segundo para a direita, o terceiro para baixo e o quarto para a esquerda.	padding: 20px 10px 5px 0px;

Quadro 2 - Espaçamentos

Fonte: do Autor (2022)

Borda (border)

A borda é uma camada fina entre o espaçamento e a margem. Ao editar a borda, você pode fazer com que os elementos desenhem seu perímetro na tela. Você pode trabalhar em bordas usando essas propriedades: “border-style”, “border-color” e “border-width”.

Estilo da borda (border-style)

A propriedade “border-style” permite que você escolha o estilo da borda. As opções que você pode usar são: “dotted”, “dashed”, “solid”, “double”, “groove”, “ridge”, “inset”, “outset”, “none” e “hidden”.

Perceba a seguir um exemplo.

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
1 <div>
2   <p class="dotted">dotted</p>
3   <p class="solid">solid</p>
4   <p class="double">double</p>
5 </div>
```

The 'CSS' tab contains the following code:

```
1 * p {
2   padding: 20px;
3   width: 140px;
4   text-align: center;
5 }
6
7 * .dotted {
8   border-style: dotted;
9 }
10
11 * .solid {
12   border-style: solid;
13 }
14
15 * .double {
16   border-style: double;
17 }
```

To the right of the editor, there are three boxes demonstrating the border styles:

- A box with a dotted border labeled "dotted".
- A box with a solid border labeled "solid".
- A box with a double border labeled "double".

Figura 12 - Exemplo de bordas

Fonte: do Autor (2022)

Cor da borda (border-color)

A propriedade “border-color” é usada para definir a cor da borda. Se você não definir uma cor, a borda por padrão será colorida usando a cor do texto no elemento. Você pode passar qualquer valor de cor válido para “border-color”.

Confira um exemplo:

The screenshot shows a browser developer tools interface with a preview window. The preview shows a single paragraph element with a yellow border. The surrounding page has a light gray background with orange circular icons at the top.

```
p {  
  border-color: yellow;  
}
```

Largura da borda (border-width)

A propriedade “border-width” é usada para definir a largura da borda. Você pode usar um dos valores predefinidos: “thin”, “medium” (o valor padrão) e “thick” ou expressar um valor em pixels, em ou rem ou qualquer outro valor de comprimento válido. Confira um exemplo:

```
p {  
    border-width: 2px;  
}
```

Essas três propriedades mencionadas, “border-width”, “border-style” e “border-color”, podem ser definidas usando a propriedade abreviada “border”. Exemplo:

```
p {  
    border: 2px orange solid;  
}
```

A propriedade “border-radius” é usada para definir cantos arredondados para a borda. Você precisa definir um valor que será usado como o raio do círculo que será usado para arredondar a borda. Confira um exemplo de uso:

```
p {  
    border-radius: 3px;  
}
```

Margem (margin)

A propriedade “margin” é comumente usada em CSS para adicionar espaço ao redor de um elemento. A margem tem quatro propriedades relacionadas que alteram a margem de uma única aresta de uma só vez: “margin-top”, “margin-right”, “margin-bottom” e “margin-left”.

A propriedade “margin” é um atalho para especificar várias margens ao mesmo tempo e, dependendo do número de valores inseridos, se comporta de maneira diferente. Confira:

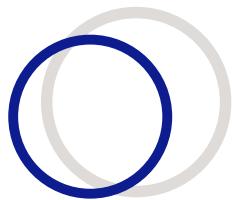
Quantidade de valores	Descrição	Exemplo
1 valor	Aplica-se a todas as margens: superior, direita, inferior e esquerda.	margin: 20px;
2 valores	Aplica o primeiro às partes inferior e superior e o segundo à esquerda e à direita.	margin: 20px 10px;
3 valores	Aplica o primeiro ao topo, o segundo à esquerda e à direita e o terceiro ao fundo.	margin: 20px 10px 30px;
4 valores	Aplica o primeiro para cima, o segundo para a direita, o terceiro para baixo e o quarto para a esquerda.	margin: 20px 10px 5px 0px;

Quadro 3 - Margens

Fonte: do Autor (2022)

O valor “auto” pode ser usado para dizer ao navegador para selecionar automaticamente uma margem e é mais comumente usado para centralizar um elemento desta maneira:





Propriedade “box-sizing”

O comportamento padrão dos navegadores ao calcular a largura de um elemento é aplicar a largura e a altura calculadas à área de conteúdo, sem levar em consideração o preenchimento (espaçamento), a borda e a margem. Essa abordagem provou ser bastante complicada de se trabalhar. Você pode alterar esse comportamento definindo a propriedade “box-sizing”. Ela tem dois valores: “border-box” e “content-box”. Confira o exemplo de definição de um elemento:

```
.caixa {  
    box-sizing: border-box;  
}
```

Caso essa propriedade for utilizada, o cálculo de largura e altura inclui o preenchimento e a borda. Apenas a margem é deixada fora, o que é razoável, pois em nossa mente também vemos isso como uma coisa separada: a margem está fora da caixa.

Propriedade “display”

A propriedade “display” de um objeto determina como ele é renderizado pelo navegador. É uma propriedade muito importante e provavelmente aquela com o maior número de valores que você pode usar. Esses valores incluem, por exemplo: “block”, “inline”, “none”, “flow”, “flex”, “grid” e “list-item”.

Por exemplo, o recurso “display: flex” dá acesso ao sistema de layout Flex, que simplifica a forma como projetamos e organizamos nossas páginas da web. Perceba pelo exemplo a seguir:

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
4      </div>
5  <div class="filho">
6    Texto
7  </div>
8  <div class="filho">
9    Texto
10 </div>
11 </div>
```

The 'CSS' tab contains the following code:

```
1 * .conteudo {
2   display: flex;
3   align-items: center;
4   justify-content: center;
5   height: 100px;
6   font-size: 40px;
7 }
8
9 * .filho {
10   border: 2px solid red;
11   margin: 4px;
12
13 }
```

To the right of the code editor, there is a preview area showing three red-bordered boxes containing the word 'Texto' each.

Figura 13 - Recurso “display: flex”

Fonte: do Autor (2022)

Viewport e media query

O recurso viewport é definido como a área visível em uma tela de janela que se refere às exibições dos dispositivos móveis. Já media query corresponde ao recurso do CSS, que permite ajustar as telas no formato do dispositivo de um usuário, envolvendo, por exemplo, sua largura e proporção.

Viewport

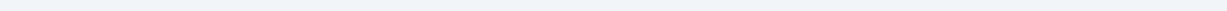
Adicionar viewport é uma maneira eficiente de melhorar as páginas da web para que apareçam em telas menores. Em termos simples, viewport ajuda os navegadores da web a quebrar as páginas e adicioná-las em uma tela pequena em um formato legível (evita a rolagem lateral). Usando a regra CSS “@viewport”, podemos defini-la de seguinte forma:

```
@viewport {  
    width: device-width;  
    zoom: 1.1;  
    min-zoom: 0.4;  
    max-zoom: 2;  
    user-zoom: fixed;  
}
```

Essa regra tem os seguintes parâmetros:

- › Largura (width): ajusta-se ao dispositivo em visão normal ou paisagem. Aceita valores como "auto", "device-width", "device-height", "length" e "percentagem". E a largura define as propriedades abreviadas, como largura máxima e largura mínima ("max-width" e "min-width").
- › Zoom: define o zoom dando escala inicial na metatag.

Para direcionar uma boa página web responsiva, sugere-se ter um dispositivo com CSS e adicionar alguns estilos a ele. Uma das principais funções em qualquer design responsivo é o tamanho da janela de visualização. Veja que no exemplo seguinte foi definida a largura em 640px:

```
@viewport {  
    width: 640px;  
}
```

Media queries

As consultas de mídia ou media queries podem modificar a aparência (e até mesmo o comportamento) de um site ou aplicativo com base em um conjunto correspondente de condições sobre as configurações do dispositivo, navegador ou sistema do usuário. Confira, na imagem seguinte, um comportamento comum de ajustes de tela utilizando media queries.

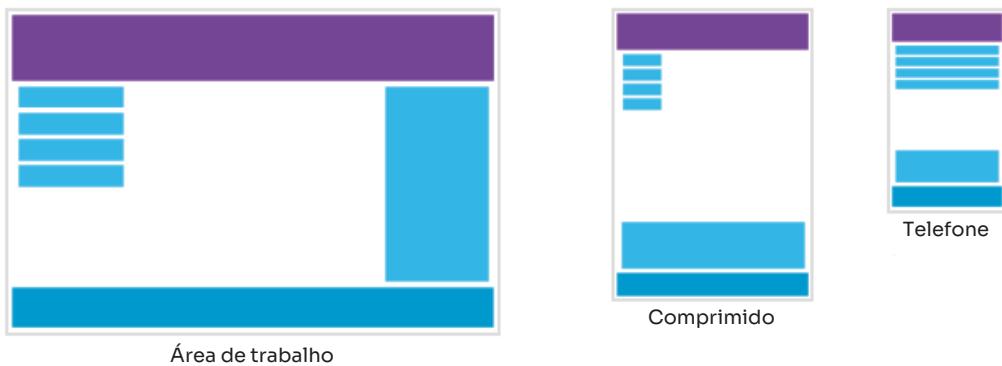


Figura 14 - Ajustes de tela utilizando media queries

Fonte: w3bai.com

As consultas de mídia CSS são uma maneira de direcionar o navegador para determinadas características, recursos e preferências do usuário e, em seguida, aplicar estilos com base nessas predefinições. Perceba com um exemplo:

```
/* Quando o navegador tiver pelo menos 600px ou mais. */
@media screen and (min-width: 600px) {
    .elemento {
        /* Aplica alguma estilização. */
    }
}
```

Existem muitas outras coisas que podemos segmentar além da largura da janela de visualização. Isso pode ser resolução de tela, orientação do dispositivo, preferência do sistema operacional etc. As folhas de estilo recebem uma regra “@media” que envolve elementos com condições para quando e onde aplicar um conjunto de estilos quando um navegador corresponder a essas condições. Entenda na imagem seguinte a sintaxe presente nas media queries.

@media screen (min-width: 320px) and (max-width: 768px)

Regra	Tipo de mídia	Recurso de mídia	Operador	Recurso de mídia
-------	---------------	------------------	----------	------------------

Figura 15 - Sintaxe de media queries
Fonte: Fonte: do Autor (2022)

O primeiro item em uma consulta de mídia é a própria regra “@media”. Essa regra é voltada para o tipo de mídia com o qual um site é visualizado, quais recursos esse tipo de mídia suporta e operadores que podem ser combinados para associar condições simples e complexas.

Em muitos casos, você verá um valor “screen” definido, pois muitos dos tipos de mídia que estamos tentando corresponder são dispositivos com telas anexadas a eles. Mas as telas não são o único tipo de mídia que podemos segmentar. Temos alguns, incluindo: “all”, “print” e “speech”.

Depois de determinar o tipo de mídia que estamos tentando corresponder, podemos começar a definir com quais recursos estamos tentando corresponder. Podemos combinar as telas com a largura, onde “screen” é o tipo de mídia e os recursos de mídia podem receber valores específicos como “min-width” e “max-width”.



Dica

As consultas de mídia suportam operadores lógicos, como muitas linguagens de programação, para que possamos corresponder os tipos de mídia com base em determinadas condições. Os operadores lógicos aceitáveis são: “and” para “e”, “or” para “ou” e “not” para “não”.



Flexbox

Flexbox é um dos dois sistemas de layouts modernos, um modelo de layout unidimensional. Ele controlará o layout com base em uma linha ou em uma coluna, mas não juntas ao mesmo tempo. O principal objetivo do flexbox é permitir que os itens preencham todo o espaço oferecido pelo seu contêiner, dependendo de algumas regras que você definir.

Um layout flexbox é aplicado a um contêiner, definindo por: “display: flex;” ou “display: inline-flex;”. O conteúdo dentro do contêiner será alinhado usando flexbox. Confira um exemplo:

The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```

1 <p>display: block;</p>
2 <section class="container">
3   <div class="item">1</div>
4   <div class="item">2</div>
5   <div class="item">3</div>
6 </section>
7
8 <p>display: flex;</p>
9 <section class="container flex">
10  <div class="item">1</div>
11  <div class="item">2</div>
12  <div class="item">3</div>
13 </section>
14
15
16
17
18

```

The 'CSS' tab contains the following code:

```

1 .container { /* Definição do flex container. */
2   max-width: 400px;
3   margin: 0 auto;
4   border: 1px solid #ccc;
5 }
6
7 .item { /* Definição do flex item. */
8   margin: 5px;
9   background: green;
10  color: #fff;
11  text-align: center;
12  font-size: 1.5em;
13 }
14
15 .flex {
16   display: flex;
17 }
18
19 p {
20   text-align: center;
21   margin: 20px 0 0 0;
22   font-size: 1.25em;
23 }

```

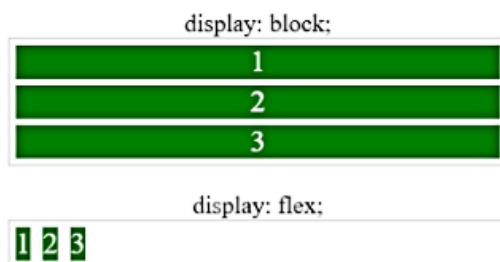


Figura 16 - Exemplo de um layout flexbox

Fonte: do Autor (2022)

Algumas propriedades do flexbox se aplicam ao contêiner, que define as regras gerais para seus itens. Veremos na sequência algumas dessas regras.

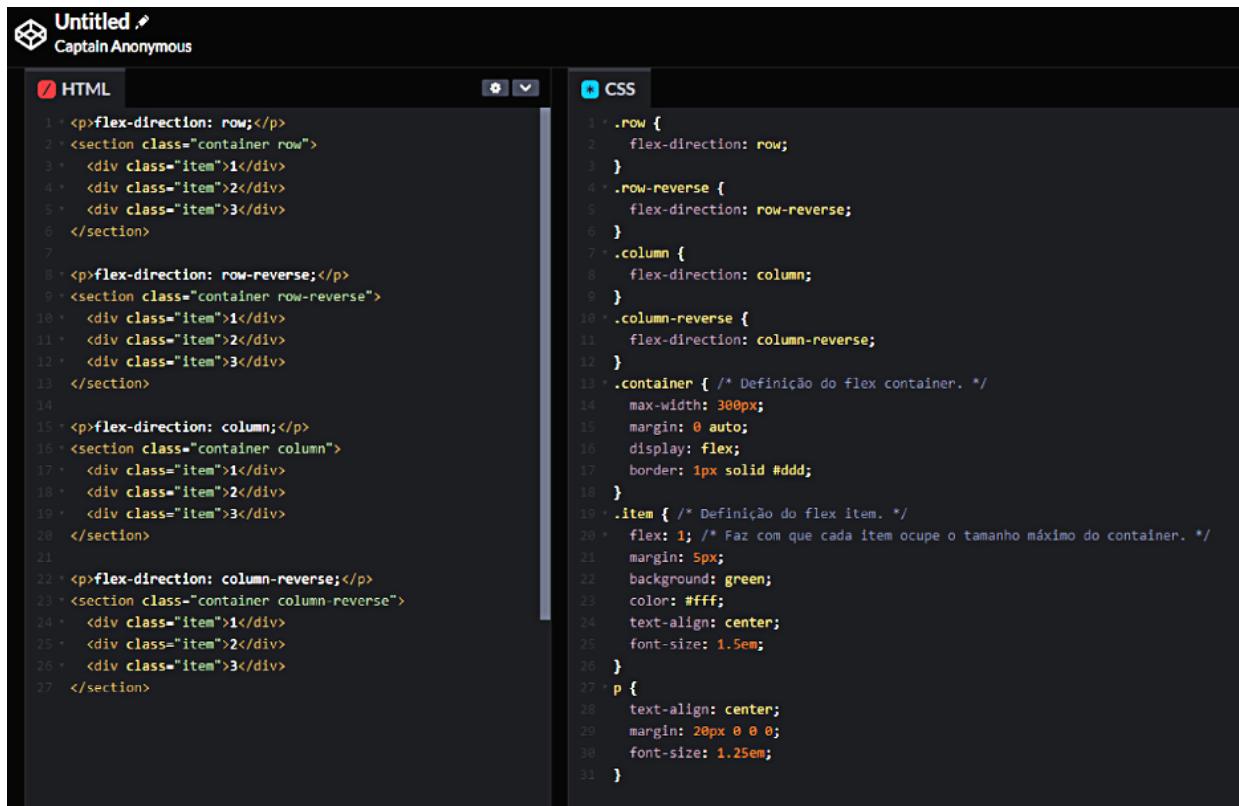
Alinhamento de linhas ou colunas

A primeira propriedade que vemos, “flex-direction”, determina se o contêiner deve alinhar seus itens como linhas ou colunas:

- › “flex-direction: row”: coloca os itens como uma linha, na direção do texto.
- › “flex-direction: row-reverse”: coloca itens exatamente como uma linha, mas na direção oposta.
- › “flex-direction: column”: coloca os itens em uma coluna, ordenando de cima para baixo.

- “flex-direction: column-reverse”: coloca os itens também em uma coluna, mas na direção oposta.

Confira um exemplo a seguir de cada uma dessas possibilidades apresentadas:



The screenshot shows a code editor window titled "Untitled" with two tabs: "HTML" and "CSS".

HTML Tab:

```

1 <p>flex-direction: row;</p>
2 <section class="container row">
3   <div class="item">1</div>
4   <div class="item">2</div>
5   <div class="item">3</div>
6 </section>
7
8 <p>flex-direction: row-reverse;</p>
9 <section class="container row-reverse">
10  <div class="item">1</div>
11  <div class="item">2</div>
12  <div class="item">3</div>
13 </section>
14
15 <p>flex-direction: column;</p>
16 <section class="container column">
17  <div class="item">1</div>
18  <div class="item">2</div>
19  <div class="item">3</div>
20 </section>
21
22 <p>flex-direction: column-reverse;</p>
23 <section class="container column-reverse">
24  <div class="item">1</div>
25  <div class="item">2</div>
26  <div class="item">3</div>
27 </section>

```

CSS Tab:

```

1 .row {
2   flex-direction: row;
3 }
4 .row-reverse {
5   flex-direction: row-reverse;
6 }
7 .column {
8   flex-direction: column;
9 }
10 .column-reverse {
11   flex-direction: column-reverse;
12 }
13 .container { /* Definição do flex container. */
14   max-width: 300px;
15   margin: 0 auto;
16   display: flex;
17   border: 1px solid #ddd;
18 }
19 .item { /* Definição do flex item. */
20   flex: 1; /* Faz com que cada item ocupe o tamanho máximo do container. */
21   margin: 5px;
22   background: green;
23   color: #fff;
24   text-align: center;
25   font-size: 1.5em;
26 }
27 p {
28   text-align: center;
29   margin: 20px 0 0 0;
30   font-size: 1.25em;
31 }

```

Figura 17 - Exemplo de código do “flex-direction”

Fonte: do Autor (2022)

O resultado do exemplo apresentado acima será o seguinte:

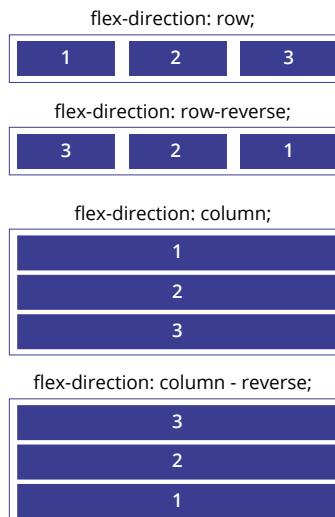


Figura 18 - Exemplo do resultado visual do “flex-direction”

Fonte: do Autor (2022)

Alinhamento vertical e horizontal

Por padrão, os itens começam da esquerda se “flex-direction” for uma linha e do topo se “flex-direction” for uma coluna. Você pode alterar esse comportamento usando “justify-content” para alterar o alinhamento horizontal e “align-items” para alterar o alinhamento vertical.

Para realizar o alinhamento horizontal, a propriedade “justify-content” pode ter cinco valores possíveis. Confira:

- › “flex-start”: alinha ao lado esquerdo do contêiner.
- › “flex-end”: alinha ao lado direito do contêiner.
- › “center”: alinha no centro do contêiner.
- › “space-between”: exibe os elementos com espaçamento igual entre eles.
- › “space-around”: exibe os elementos com espaçamento igual ao redor deles.

The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML Tab:

```
1 <p>justify-content: flex-start;</p>
2 <section class="container flex-start">
3   <div class="item">1</div>
4   <div class="item">2</div>
5   <div class="item">3</div>
6 </section>
7
8 <p>justify-content: flex-end;</p>
9 <section class="container flex-end">
10  <div class="item">1</div>
11  <div class="item">2</div>
12  <div class="item">3</div>
13 </section>
14
15 <p>justify-content: center;</p>
16 <section class="container center">
17  <div class="item">1</div>
18  <div class="item">2</div>
19  <div class="item">3</div>
20 </section>
21
22 <p>justify-content: space-between;</p>
23 <section class="container space-between">
24  <div class="item">1</div>
25  <div class="item">2</div>
26  <div class="item">3</div>
27 </section>
28
29 <p>justify-content: space-around;</p>
30 <section class="container space-around">
31  <div class="item">1</div>
32  <div class="item">2</div>
33  <div class="item">3</div>
34 </section>
35
```

CSS Tab:

```
1 .flex-start {
2   justify-content: flex-start;
3 }
4 .flex-end {
5   justify-content: flex-end;
6 }
7 .center {
8   justify-content: center;
9 }
10 .space-between {
11   justify-content: space-between;
12 }
13 .space-around {
14   justify-content: space-around;
15 }
16 .container {
17   /* Definição do flex container. */
18   max-width: 300px;
19   margin: 0 auto;
20   display: flex;
21   border: 1px solid #ddd;
22 }
23 .item {
24   /* Definição do flex item. */
25   margin: 5px;
26   padding: 0 10px;
27   background: green;
28   color: #fff;
29   text-align: center;
30   font-size: 1.5em;
31 }
32 p {
33   text-align: center;
34   margin: 20px 0 0 0;
35   font-size: 1.25em;
36 }
37
```

Figura 19 - Exemplo de código do “justify-content”

Fonte: do Autor (2022)

A imagem seguinte apresenta o resultado do código-fonte descrito anteriormente.

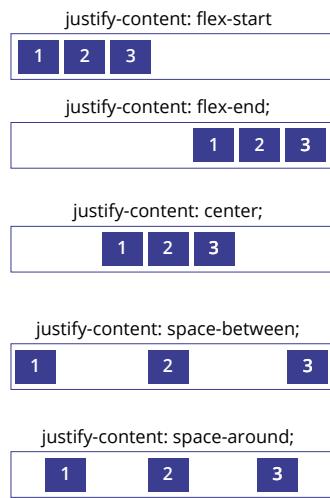


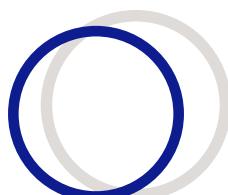
Figura 20 - Exemplo do resultado visual do “justify-content”

Fonte: do Autor (2022)

Para alterar o alinhamento vertical, a propriedade “align-items” tem cinco valores possíveis:

- › “flex-start”: permite alinhar ao topo do contêiner.
- › “flex-end”: permite alinhar ao fundo do contêiner.
- › “center”: permite alinhar no centro vertical do contêiner.
- › “baseline”: permite exibir o elemento na linha de base do contêiner.
- › “stretch”: faz com que os itens sejam esticados para caber no contêiner.

Compreenda com um exemplo para cada uma das possibilidades apresentadas anteriormente.



The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML Tab:

```
1 <p>align-items: flex-start;</p>
2 <section class="container flex-start">
3   <div class="item">Teste 1</div>
4   <div class="item">Teste 2 Teste 2</div>
5   <div class="item">Teste 3 Teste 3 Teste 3</div>
6 </section>
7
8 <p>align-items: flex-end;</p>
9 <section class="container flex-end">
10  <div class="item">Teste 1</div>
11  <div class="item">Teste 2 Teste 2</div>
12  <div class="item">Teste 3 Teste 3 Teste 3</div>
13 </section>
14
15 <p>align-items: center;</p>
16 <section class="container center">
17  <div class="item">Teste 1</div>
18  <div class="item">Teste 2 Teste 2</div>
19  <div class="item">Teste 3 Teste 3 Teste 3</div>
20 </section>
21
22 <p>align-items: baseline;</p>
23 <section class="container baseline">
24  <div class="item">Teste 1</div>
25  <div class="item">Teste 2 Teste 2</div>
26  <div class="item">Teste 3 Teste 3 Teste 3</div>
27 </section>
28
29 <p>align-items: stretch;</p>
30 <section class="container stretch">
31  <div class="item">Teste 1</div>
32  <div class="item">Teste 2 Teste 2</div>
33  <div class="item">Teste 3 Teste 3 Teste 3</div>
34 </section>
```

CSS Tab:

```
1 .flex-start {
2   align-items: flex-start;
3 }
4 .flex-end {
5   align-items: flex-end;
6 }
7 .center {
8   align-items: center;
9 }
10 .baseline {
11   align-items: baseline;
12 }
13 .stretch {
14   align-items: stretch;
15 }
16 .container { /* Definição do flex container. */
17   max-width: 420px;
18   margin: 0 auto;
19   display: flex;
20   border: 1px solid #ddd;
21 }
22 .column {
23   flex-direction: column;
24 }
25 .item { /* Definição do flex item. */
26   flex: 1;
27   margin: 5px;
28   padding: 0 10px;
29   background: green;
30   color: #fff;
31   text-align: center;
32   font-size: 1.5em;
33 }
34 p {
35   text-align: center;
36   margin: 20px 0 0 0;
37   font-size: 1.25em;
38 }
```

Figura 21 - Exemplo de código do “align-items”

Fonte: do Autor (2022)

O resultado do código apresentado anteriormente pode ser visualizado na sequência:

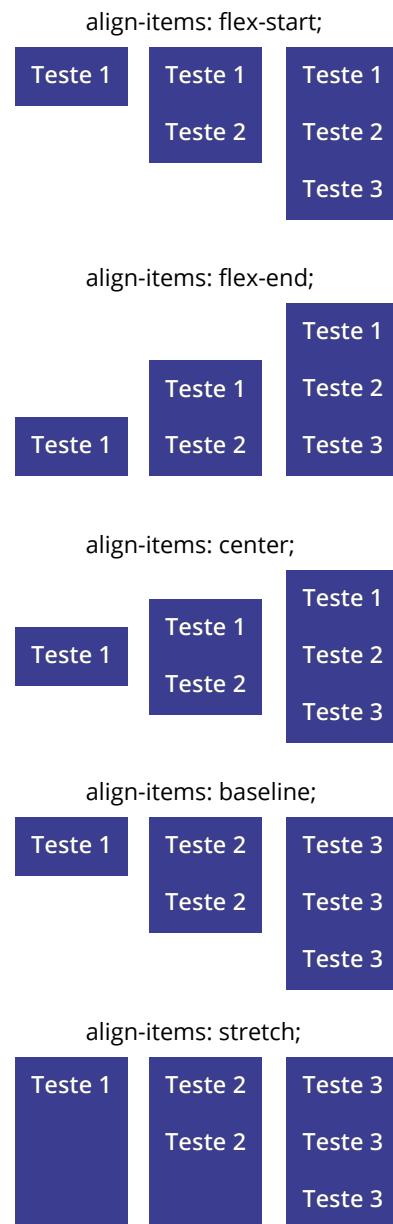
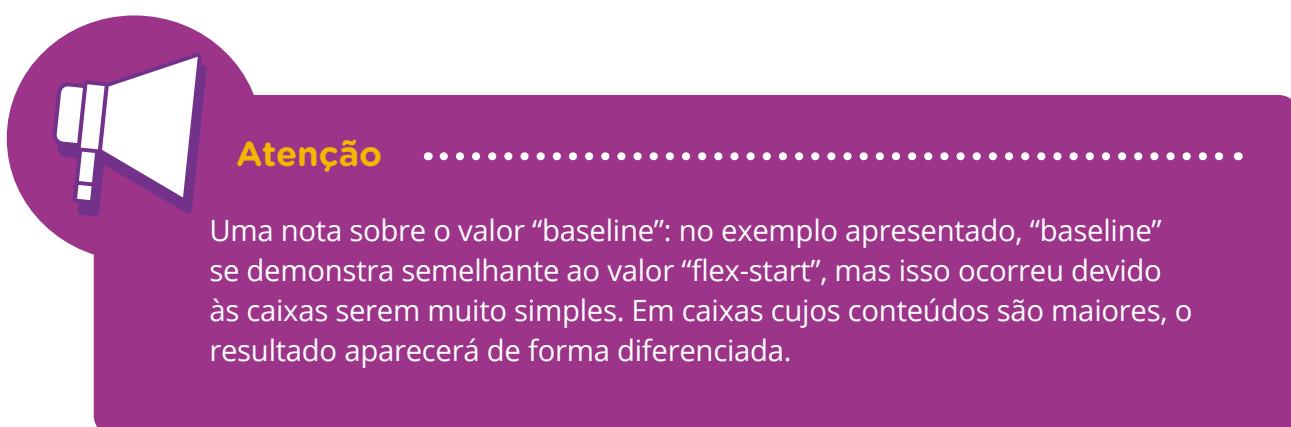


Figura 22 - Exemplo do resultado visual do “align-items”

Fonte: do Autor (2022)



Como você pode perceber, neste estudo houve apresentação dos recursos de posicionamento para auxiliar na manipulação de elementos; visibilidade, para ocultação de um elemento sem alterar o layout da página; e transparência, responsável por definir a opacidade de um elemento. Apresentamos, também, como aplicar sombras, gradientes e transformações em elementos, pois esses itens desempenham um papel importante na aparência do site.



Você conheceu a importância do box model de modo a identificar o dimensionamento dos elementos com base em algumas propriedades CSS. Posteriormente, você utilizou recursos de viewport e media query. Assim, percebeu as formas de implementação desses itens para garantir a exibição adequada em dispositivos móveis, permitindo construir um layout flexível. E, por fim, compreendeu como realizar a organização de elementos utilizando um recurso tão poderoso: o flexbox. Esperamos que este material tenha auxiliado em seus estudos rumo à construção de páginas web cada vez mais estilizadas e responsivas. Continue seus estudos!

REFERÊNCIAS

CSS Snapshot 2021. **W3**, 2021. Disponível em: <https://www.w3.org/TR/css-2021/>. Acesso em: 12 jul. 2022.

REFERÊNCIA de CSS. **MDN**, [s.d.]. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS/Reference>. Acesso em: 12 jul. 2022.

WEB DESIGN Responsive - Consultas de mídia. w3bai.com, [s.d.]. Disponível em: https://www.w3bai.com/pt/css/css_rwd_mediaqueries.html#gsc.tab=0. Acesso em: 25 jul. 2022.



