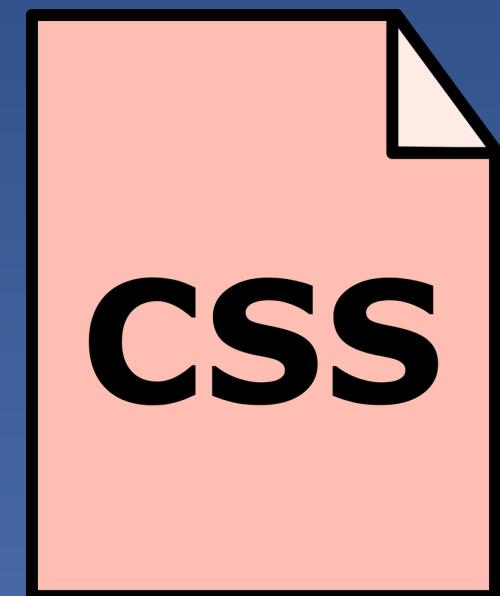


Introdução ao



Desenvolvimento Front-End

Prof^a. Ma. Adrielly Sayonara

Editor e Introdução



Editor de Código

- **Editores**

Visual Studio Code, Sublime Text, Vim e outros.

- **Vantagens**

Destaca a sintaxe, autocompleta o código, organiza o código, informa erros e mais.

- **Visual Studio Code**

<https://code.visualstudio.com>

Configurações

Ctrl + Shift + P

Open Settings (JSON)

```
{  
  "workbench.colorTheme": "1mm - Dracula",  
  "editor.fontSize": 16,  
  "editor.lineHeight": 1.75,  
  "editor.tabSize": 2  
}
```



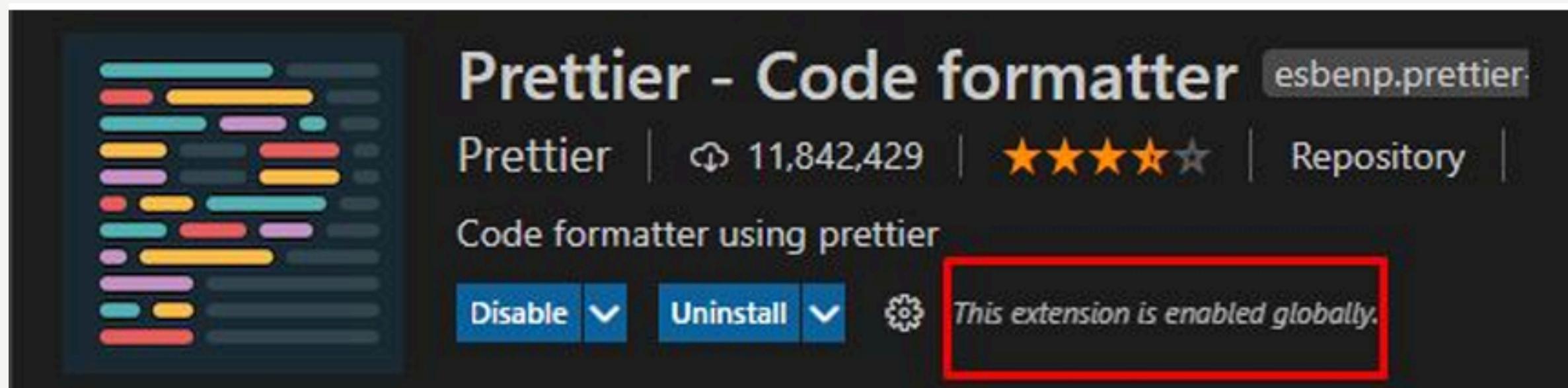
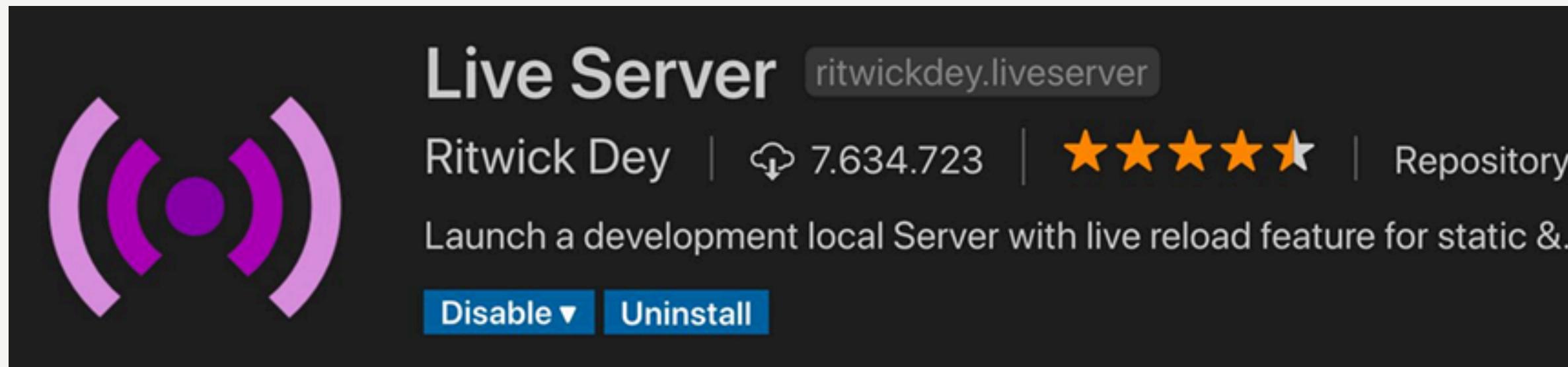
Editor Plugins

- Live Server

<https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

- Prettier

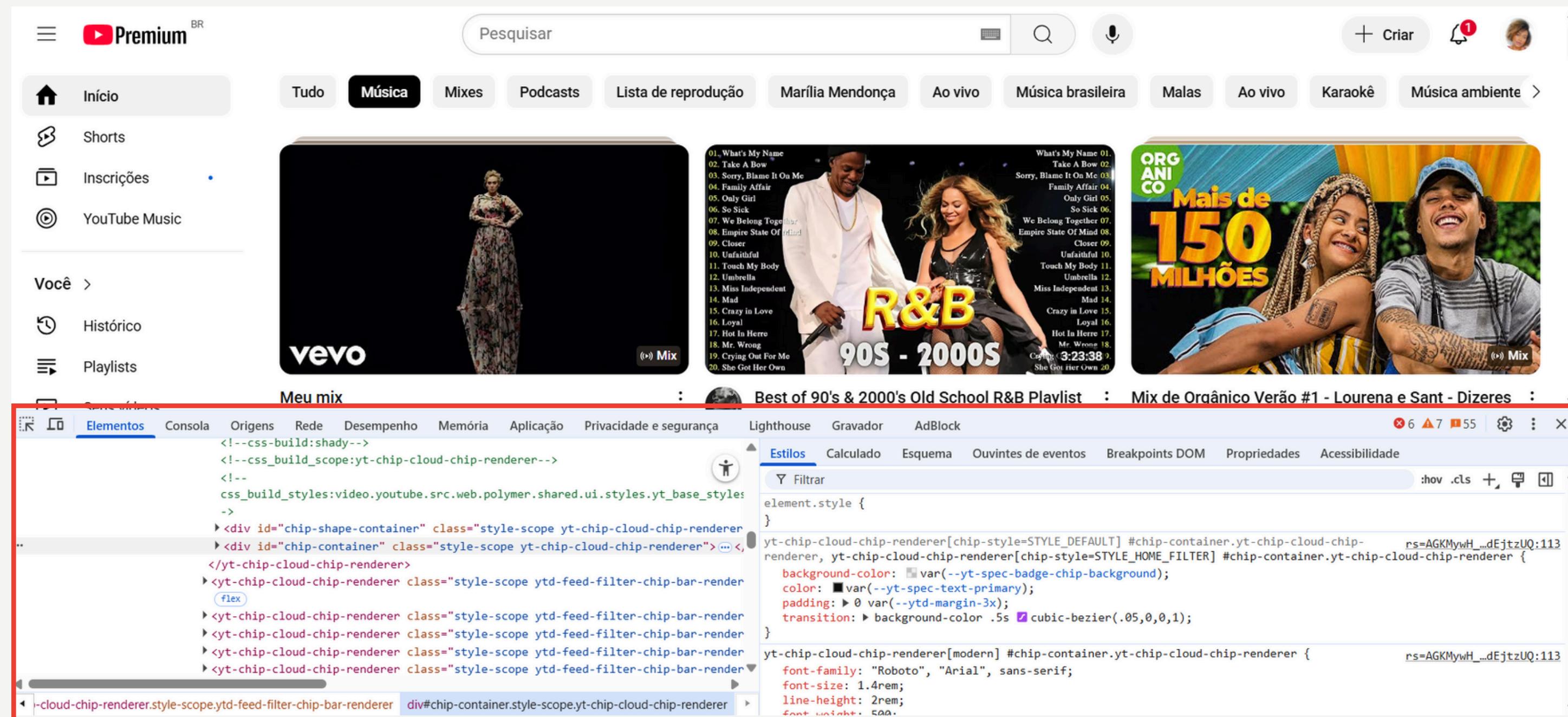
<https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode>



Ferramentas

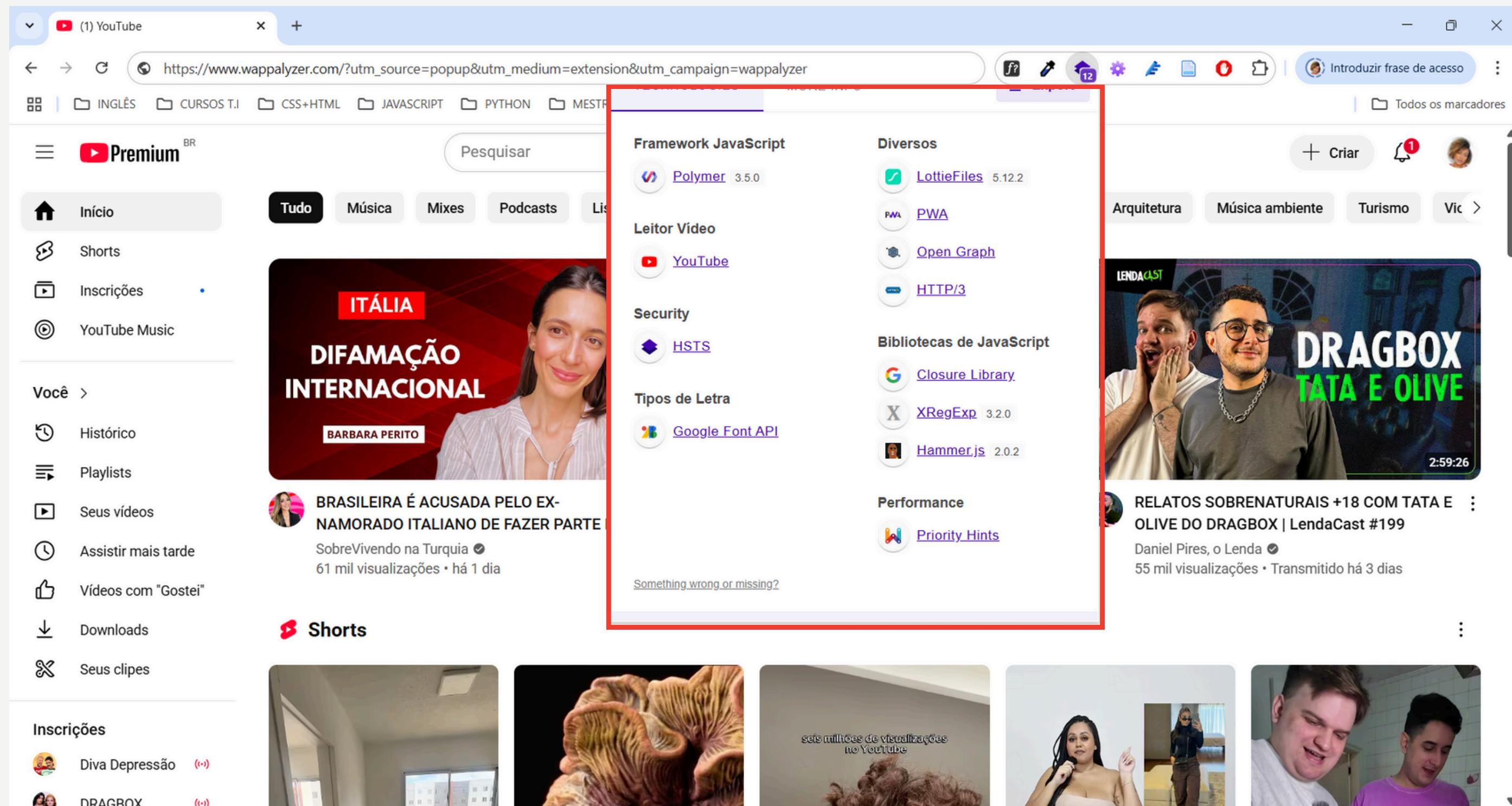
• Dev Tools

As Ferramentas do Desenvolvedor (Developer Tools) são utilizadas para depurar/debug (identificar/eliminar erros) os nossos projetos



Ferramentas

- Wappalyzer



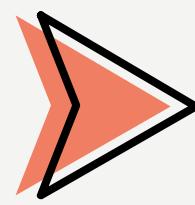
Ferramentas

- Web developer

The screenshot shows a web browser window with a YouTube page loaded. The address bar displays the URL: https://www.wappalyzer.com/?utm_source=popup&utm_medium=extension&utm_campaign=wappalyzer. A red box highlights the Wappalyzer toolbar extension menu, which includes options like Disable, Cookies, CSS, Forms, Images, Information, Miscellaneous, Outline, Resize, Tools, and Options. The main content area shows a video thumbnail for 'ITÁLIA DIFAMAÇÃO INTERNACIONAL' by BARBARA PERITO, a Vevo channel thumbnail, and another video thumbnail for 'DRAGBOX TATA E OLIVE'.

YouTube page content:

- ITÁLIA DIFAMAÇÃO INTERNACIONAL - BARBARA PERITO (1:02:37)
- vevo
- Meu mix (Adele, Bruno Mars, Rihannae outros) - Atualizado hoje
- BRASILEIRA É ACUSADA PELO EX-NAMORADO ITALIANO DE FAZER PARTE DE... (SobreVivendo na Turquia) - 61 mil visualizações • há 1 dia
- RELATOS SOBRENATURAIS +18 COM TATA E OLIVE DO DRAGBOX | LendaCast #199 - Daniel Pires, o Lenda - 55 mil visualizações • Transmitido há 3 dias



O que é CSS ?



CSS é uma linguagem de estilo

CSS (Cascading Style Sheets) é a linguagem responsável por definir a aparência visual dos documentos HTML, permitindo controlar a forma como o conteúdo é apresentado na página.



Separar conteúdo da apresentação

Com o CSS, é possível manter o conteúdo (HTML) separado da apresentação (estilo), o que torna a manutenção do site mais fácil, limpa e eficiente.



Controle total da aparência visual

CSS oferece total controle sobre a aparência da página: Você pode definir cores, fontes, tamanhos, espaçamentos, layouts e muito mais, personalizando a experiência do usuário de forma precisa.



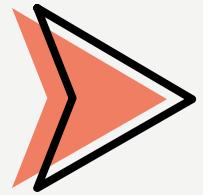
Importância na web moderna

CSS é essencial para criar interfaces atrativas, consistentes e responsivas – adaptáveis a diferentes tamanhos de tela. É uma das habilidades mais importantes para quem desenvolve para a web.



Sintaxe CSS

```
h1 {  
    color: blue;  
    background-color: yellow;  
}  
  
p {  
    color: red;  
}
```



Formas de aplicar CSS

Inline

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS: Formas de aplicar</title>
</head>
<body>
    <p style="color: red;">Lorem ipsum dolor sit amet, consectetur adipiscing elit.<br>
        Nulla maximus metus sed neque tempus luctus. Nullam vitae turpis velit.<br>
        Duis vel dolor eu lectus gravida malesuada.
    </p>
</body>
</html>
```



Formas de aplicar CSS

Internal Style Sheet

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS: Formas de aplicar</title>

    <style>
        p{
            color: red;
        }
    </style>

</head>
<body>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.<br>
       Nulla maximus metus sed neque tempus luctus. Nullam vitae turpis velit.<br>
       Duis vel dolor eu lectus gravida malesuada.
    </p>
</body>
</html>
```

► Formas de aplicar CSS

External Style Sheet

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS: Formas de aplicar</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <p class="paragrafo1">Lorem ipsum dolor sit amet, consectetur adipiscing elit.<br>
        Nulla maximus metus sed neque tempus luctus. Nullam vitae turpis velit.<br>
        Duis vel dolor eu lectus gravida malesuada.
    </p>
</body>
</html>
```

HTML

```
.paragrafo1{
    color: red;
}
```

CSS

Seletores CSS

Seletores: O que são?

São padrões utilizados para identificar quais elementos HTML devem receber determinados estilos. Eles permitem aplicar estilos de forma precisa e eficiente, controlando exatamente onde e como o CSS será aplicado na página.

Tipos de seletores

Existem diferentes tipos de seletores no CSS, cada um com sua forma específica de selecionar elementos e aplicar estilos.

Os mais comuns são:



- **Seletor de tipo:** seleciona elementos pelo nome da tag (ex: p, h1)
- **Seletor de classe:** usa um ponto (.) para selecionar elementos com determinada classe (ex: .destaque)
 - **class="destaque"**
- **Seletor de ID:** usa uma cerquilha (#) para selecionar um elemento com um ID específico (ex: #titulo)
 - **id="titulo"**
- **Seletor descendente:** seleciona elementos que estão dentro de outros elementos (ex: div p seleciona parágrafos dentro de divs)

➤ Seletores CSS

```
body>

<h1 id="cabecalho">Título com ID</h1>
<h2 class="titulo">Subtítulo com Classe</h2>
<p>Parágrafo normal</p>
<p class="alerta">Parágrafo com classe 'alerta'</p>

<div>
  <p>Parágrafo dentro de div</p>
  <section>
    <p>Outro parágrafo dentro de div > section</p>
  </section>
</div>

<ul>
  <li>Item 1</li>
  <li class="alerta">Item 2</li>
  <li>Item 3</li>
  <ul>
    <li>Subitem</li>
  </ul>
</ul>

<h1>Outro Título</h1>
<p>Parágrafo após título</p>
<p>Mais um parágrafo após título</p>

<a href="https://google.com">Link Google</a><br>
<a href="manual.pdf">Manual PDF</a><br>
<a href="/user/login">Login</a><br>

<form>
  <input type="text" required><br>
  <input type="checkbox" checked id="chk1"><label for="chk1">Check</label><br>
  <input type="text" disabled><br>
</form>

<button>Botão</button>

<article>
  <p>Artigo com imagem abaixo</p>
  
</article>

</body>
```

HTML

```
/* Seletor de Tipo */
p {
  font-size: 16px;
  margin-bottom: 10px;
}

/* Classe */
.alerta {
  color: red;
}

/* ID */
#cabecalho {
  background-color: lightgray;
  padding: 10px;
}

/* Universal */
* {
  box-sizing: border-box;
}

/* Descendente */
div p {
  color: green;
}

/* Filho Direto */
ul > li {
  font-weight: bold;
}

/* Irmão Adjacente */
h1 + p {
  color: orange;
}

/* Irmão Geral */
h1 ~ p {
  font-style: italic;
}

/* Múltiplos */
h1, h2, .titulo {
  font-family: Arial, sans-serif;
}
```

CSS

```
/* Atributos */
input[required] {
  border: 2px solid red;
}
a[href="https://google.com"] {
  color: blue;
}
img[src^="icon"] {
  width: 50px;
}
a[href$=".pdf"] {
  color: brown;
}
a[href*="login"] {
  background-color: yellow;
}

/* Pseudo-classes */
button:hover {
  background-color: lightgreen;
}
li:first-child {
  text-decoration: underline;
}
li:last-child {
  color: purple;
}
li:nth-child(2) {
  background-color: #eee;
}
p:nth-of-type(odd) {
  background-color: #f9f9f9;
}
p:not(.alerta) {
  color: gray;
}
input:checked + label {
  font-weight: bold;
}
input:disabled {
  background-color: #ccc;
}
```

CSS

```
/* Pseudo-elementos */
h1::before {
  content: "👉 ";
}
p::after {
  content: " ✅ ";
}
p::first-letter {
  font-size: 200%;
}
p::first-line {
  color: blue;
}

/* Seletor Combinado */
ul > li.alerta:first-child:hover {
  color: red;
}

/* :is() */
:is(h1, h2, h3) {
  border-bottom: 1px solid #000;
}

/* :where() */
:where(header, footer) {
  background: #eee;
}
```

CSS



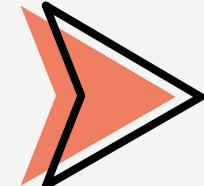
Especificidade no CSS

A especificação é a maneira de como os navegadores definem quais valores de propriedades são os mais relevantes para o elemento a ser utilizado. A especificação é baseada apenas nas regras impostas na composição de diferentes tipos de seletores.

Prioridade	Tipo de seletor	Exemplo
	Estilo Inline	<p style="... ">
	ID	#menu
	Classe, Atributo, Pseudo-classe	.alerta, [type], :hover
	Tipo (Elemento HTML)	p, div, h1
	Universal	*



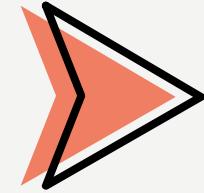
A exceção !important



Herança no CSS

A herança em CSS define o que acontece quando nenhum valor é explicitamente definido para uma propriedade em um elemento. Ela não depende da especificidade ou do tipo de seletor, mas sim de como a propriedade CSS foi projetada: algumas herdam automaticamente, outras não.

Tipo de Propriedade	Exemplos de Propriedades	Herda por padrão?	Valor padrão (se não herdado)	Observações
Texto e Fonte	color, font-family, font-size, line-height, letter-spacing	<input checked="" type="checkbox"/> Sim	—	Muito usadas em herança, especialmente para textos.
Layout	margin, padding, width, height, display, position	<input type="checkbox"/> Não	0, auto, block, etc.	Necessário definir em cada elemento, se desejado.
Borda e Fundo	border, border-radius, background, box-shadow	<input type="checkbox"/> Não	none, transparent, etc.	Não herdam — deve ser reaplicado em cada elemento.
Texto decorativo	text-align, text-indent, visibility, white-space	<input checked="" type="checkbox"/> Sim	—	Essas afetam a renderização do texto e são herdadas.
Flex/Grid/Box	justify-content, align-items, flex, grid-template-columns	<input type="checkbox"/> Não	—	Não herdam — específicas do layout do container.
Espaçamento e Box Model	overflow, box-sizing, z-index, float, clear	<input type="checkbox"/> Não	visible, content-box, etc.	Essenciais no controle do fluxo e renderização de caixas.
Tabela	border-collapse, empty-cells, caption-side	<input type="checkbox"/> Não	separate, show, etc.	Propriedades específicas de tabelas, não herdadas.
Outras visuais	opacity, visibility, cursor, z-index	<input type="checkbox"/> Não	1, visible, auto, etc.	Não herdam. Precisa definir em cada elemento.
Comportamento de herança	inherit, initial, unset	<input type="checkbox"/> Depende	—	Palavras-chave que controlam manualmente o comportamento da herança.



Propriedades no CSS

Propriedades: O que são?

As propriedades determinam quais aspectos visuais dos elementos HTML serão modificados. Elas permitem controlar características como:



- **Cor** (color, background-color)
- **Tamanho** (width, height, font-size)
- **Espaçamento** (margin, padding)
- **Bordas, fontes, alinhamento** e muito mais



Div e Span

<div>: **container de bloco** ocupa 100% da largura disponível é ideal para áreas grandes como cabeçalho, conteúdo e rodapé.

****: **container inline** ocupa apenas o espaço do conteúdo é ideal para pequenos trechos de texto dentro de parágrafos.

Para que servem?

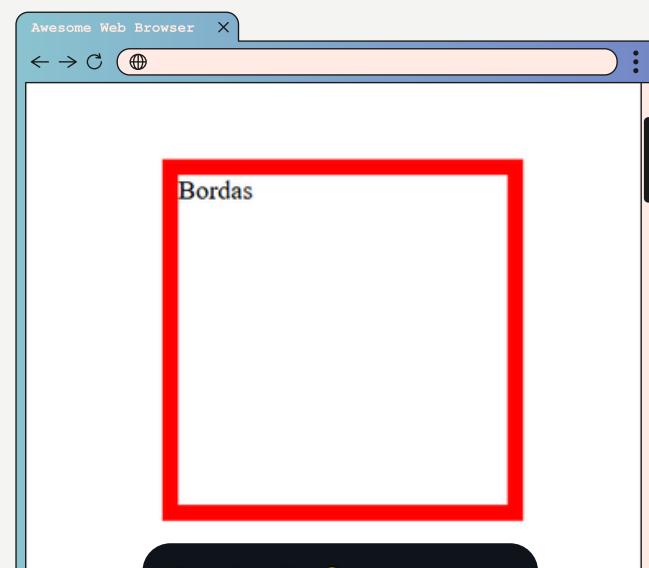
Organizar visualmente o conteúdo do site
Aplicar estilos CSS com mais precisão
Agrupar elementos HTML relacionados



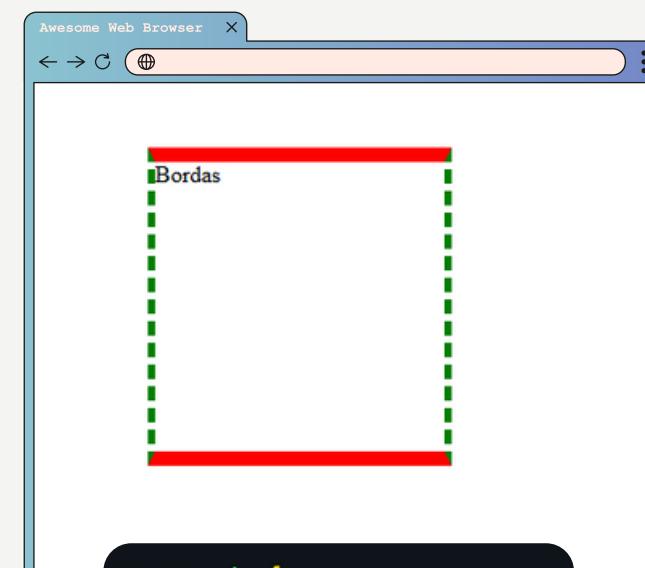
Bordas

➤ As bordas no CSS são linhas que contornam um elemento HTML.
Elas servem para:

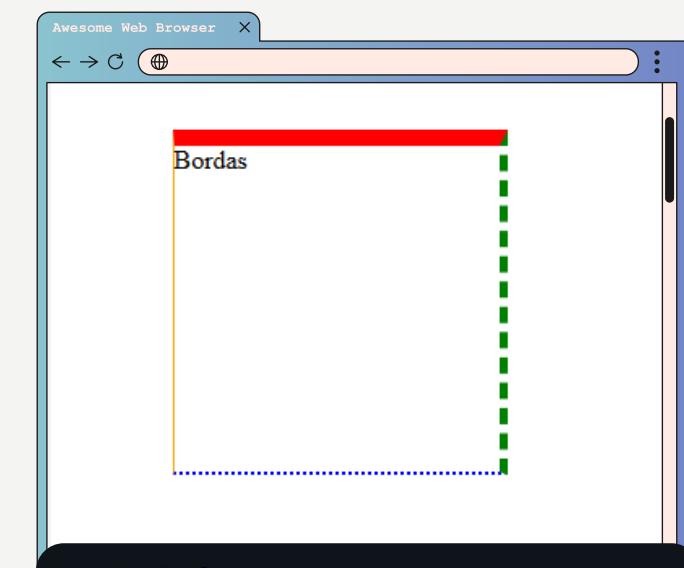
- • Destacar o conteúdo
- Delimitar áreas da página
- Criar efeitos visuais (ex: botões, caixas, cartões, etc.)



```
#conteudo {  
    border-width: 10px;  
    border-style: solid;  
    border-color: red;  
}
```



```
#conteudo {  
    border-width: 10px 5px;  
    border-style: solid dashed;  
    border-color: red green;  
}
```



```
#conteudo {  
    border-width: 10px 5px 2px 1px;  
    border-style: solid dashed dotted double;  
    border-color: red green blue orange;  
}
```

Fontes e Cores

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Fontes e Cores</title>
</head>
<body>

  <h1 class="titulo">Trabalhando com Fontes e Cores</h1>

  <p class="texto">
    Este parágrafo usa uma fonte personalizada, tamanho definido, cor do texto e cor de fundo.
  </p>

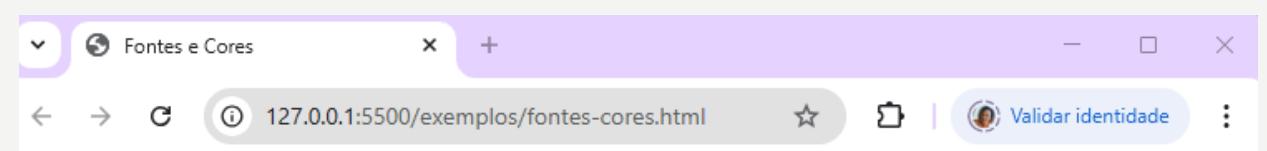
</body>
</html>
```

HTML

```
/* Define estilos para o título principal */
.titulo {
  color: #2c3e50;           /* Cor do texto (azul escuro) */
  background-color: #ecf0f1;  /* Cor de fundo (cinza claro) */
  font-size: 36px;          /* Tamanho da fonte */
  font-family: Arial, sans-serif; /* Tipo da fonte */
  padding: 10px;
  text-align: center;
}

/* Define estilos para o parágrafo */
.texto {
  color: #ffffff;           /* Cor do texto (branco) */
  background-color: #e67e22;  /* Cor de fundo (laranja) */
  font-size: 20px;          /* Tamanho da fonte */
  font-family: "Georgia", serif; /* Tipo da fonte */
  padding: 15px;
  margin: 20px;
  border-radius: 8px;
}
```

CSS



Trabalhando com Fontes e Cores

Este parágrafo usa uma fonte personalizada, tamanho definido, cor do texto e cor de fundo.

Estilos de Texto

► **font-family: escolhendo a fonte do texto**

- Define a família da fonte.
- Lista de preferência: se a primeira não carregar, usa a próxima.

► **font-size: tamanho da fonte**

Outras unidades que podem ser usadas:

- em, rem, %

► **font-weight: peso da fonte (negrito)**

Valores possíveis:

- Palavras-chave: normal, bold, lighter, bolder
- Números: 100 (fino) a 900 (mais grosso)

► **font-style: estilo da fonte (itálico)**

Outros valores:

- normal: texto padrão
- oblique: inclinado, semelhante ao itálico

► **text-decoration: decoração do texto**

Outras opções:

- none: remove decorações
- overline: linha acima do texto
- line-through: riscado

► **text-align: alinhar horizontalmente o conteúdo de texto**

Valores possíveis:

- left, right, center, justify, start e end

► **Propriedade abreviada**

```
.exemplo {  
    font: bold 24px Arial, sans-serif;  
}
```



Exemplo de Estilos de Texto

127.0.0.1:5500/exemplos/estilos-text...

Validar identidade

Título Estilizado

Este é um parágrafo com a propriedade font abreviada.

Fonte-family: Arial, Helvetica, sans-serif.

Tamanho da fonte em em: 1.5em.

Texto com peso bold e outro com peso 300. e 300

Texto normal, itálico e obliqua.

Texto sublinhado, linha acima, riscado.

Alinhado à esquerda

Alinhado à direita

Centralizado

Texto justificado. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor.

.titulo {
text-align: center;
font: bold 32px 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.exemplo {
font: bold 24px Arial, sans-serif;
}

.familia {
font-family: Arial, Helvetica, sans-serif;
}

.tamanho {
font-size: 1.5em;
}

.peso {
font-weight: bold;
}
.peso::after {
content: " e 300";
font-weight: 300;
}

.estilo {
font-style: normal;
}
.italico {
font-style: italic;
}
.obliqua {
font-style: oblique;
}

.decoracao .sublinhado {
text-decoration: underline;
}
.decoracao .linha-cima {
text-decoration: overline;
}

.decoracao .riscado {
text-decoration: line-through;
}

.alinhamento p {
background: #e1e1e1;
padding: 5px 10px;
margin-bottom: 5px;
}

.esquerda {
text-align: left;
}
.direita {
text-align: right;
}
.centro {
text-align: center;
}
.justificado {
text-align: justify;
}

HTML

```
<h1 class="titulo">Título Estilizado</h1>  
  
<p class="exemplo">Este é um parágrafo com a propriedade <code>font</code> abreviada.</p>  
  
<p class="familia">Fonte-family: Arial, Helvetica, sans-serif.</p>  
  
<p class="tamanho">Tamanho da fonte em <strong>em</strong>: 1.5em.</p>  
  
<p class="peso">Texto com peso <strong>bold</strong> e outro com peso <strong>300</strong>. </p>  
  
<p class="estilo">Texto normal, <span class="italico">itálico</span> e <span class="obliqua">obliqua</span>. </p>  
  
<p class="decoracao">Texto <span class="sublinhado">sublinhado</span>, <span class="linha-cima">linha acima</span>, <span class="riscado">riscado</span>. </p>  
  
<div class="alinhamento">  
  <p class="esquerda">Alinhado à esquerda</p>  
  <p class="direita">Alinhado à direita</p>  
  <p class="centro">Centralizado</p>  
  <p class="justificado">Texto justificado. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor.</p>  
</div>
```

Unidades de Medida no CSS

Por que existem diferentes unidades?

No CSS, usamos unidades de medida para definir:

- Tamanhos de texto
- Largura e altura de elementos
- Margens, paddings, bordas
- Posicionamentos e espaçamentos

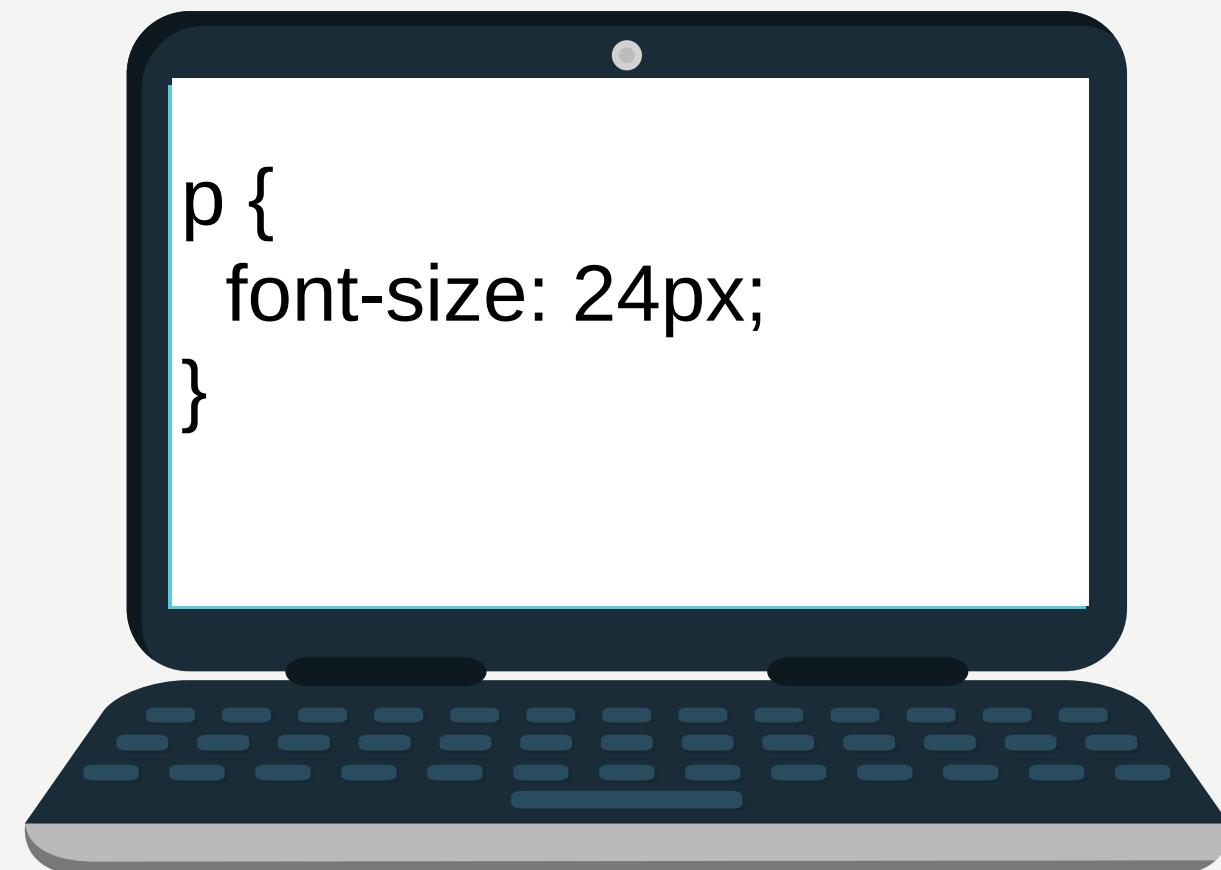
⚠️ Algumas são fixas (não mudam com o contexto).
Outras são relativas ao tamanho do elemento pai, da fonte raiz ou da janela do navegador.

Unidades de Medida no CSS

► Unidades Fixas

px (pixel)

- Medida absoluta e fixa.
- 1px equivale a um ponto na tela, independente do zoom ou do tamanho do container.
- Muito usado para precisão visual.



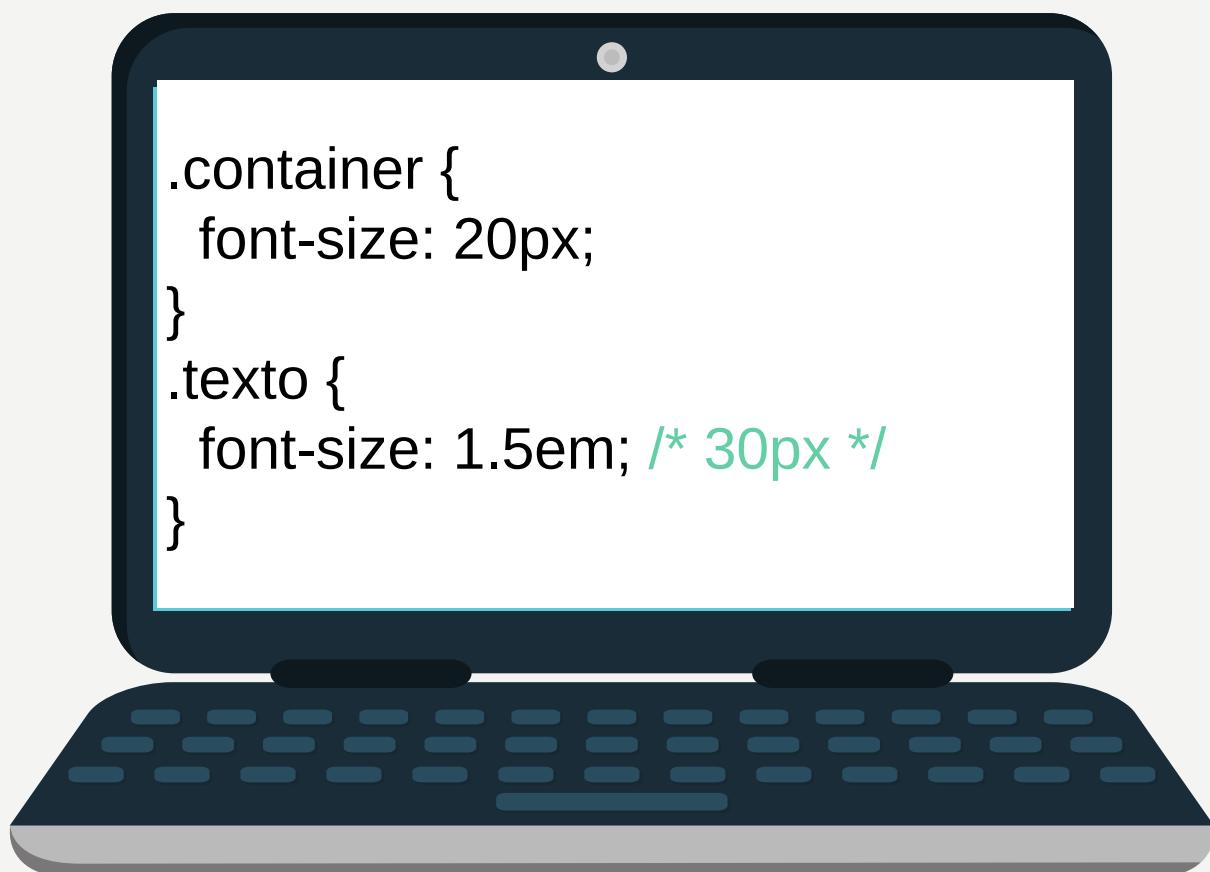
Usar com moderação em layouts modernos – não se adapta automaticamente a diferentes tamanhos de tela.

Unidades de Medida no CSS

➤ Unidades Relativas

em

- relativo ao tamanho da fonte do elemento pai;
- Se o pai tem font-size: 20px, então 1em = 20px



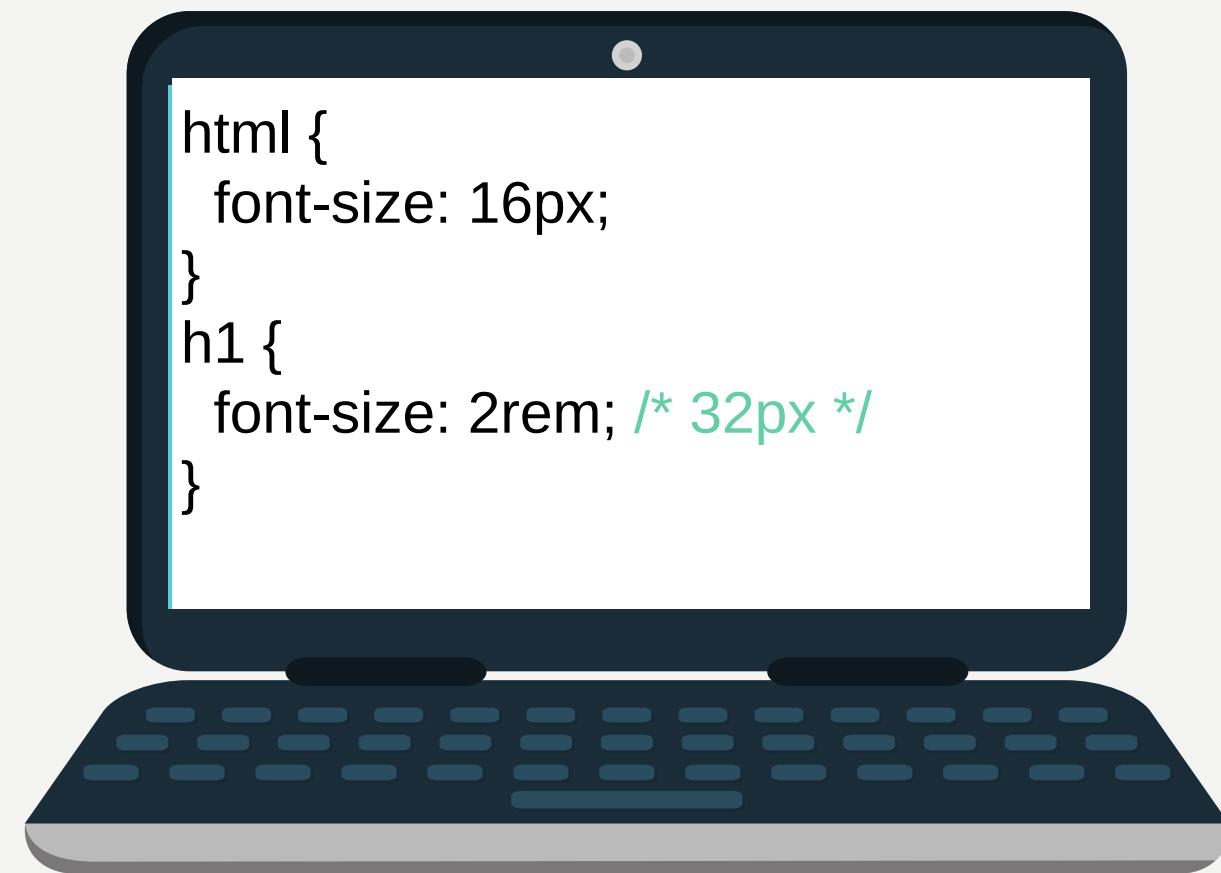
Muito usado em componentes reutilizáveis para manter proporções.

Unidades de Medida no CSS

➤ Unidades Relativas

rem

- Relativo ao tamanho da fonte raiz (html);
- Geralmente `html {font-size: 16px;}`, então $1\text{rem} = 16\text{px}$.



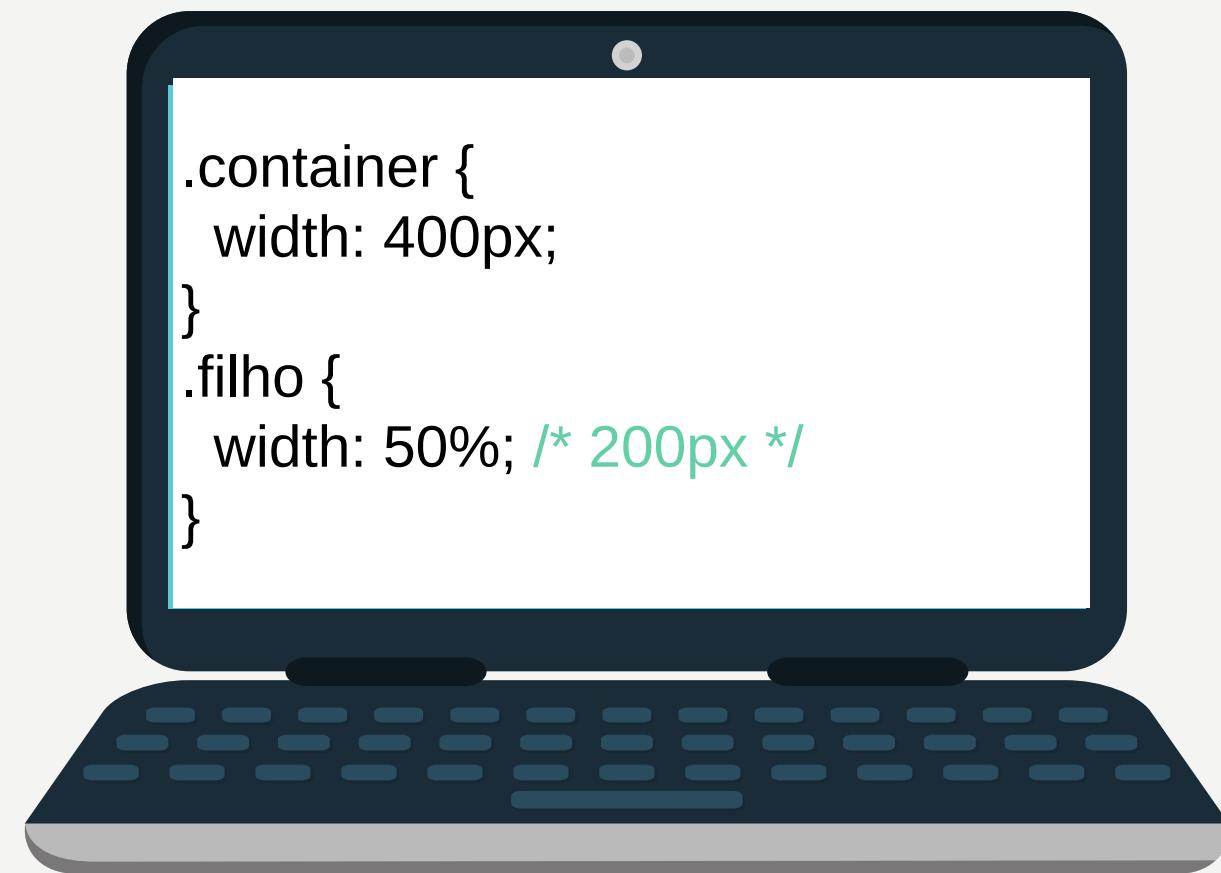
Muito usado para escalar todo o site com base em uma única definição de tamanho.

Unidades de Medida no CSS

➤ Unidades Relativas

%

- Relativo ao elemento pai;
- Pode se aplicar a tamanho de fonte, largura, altura, etc.



Super útil para criar layouts fluidos e responsivos.

Propriedades de background

➤ **background-color**

- red
- #00ff00
- rgb(255, 255, 0)
- transparent

➤ **background-image**

```
background-image: url("imagem.jpg");
```

➤ **background-position**

Posição da imagem. Exemplos:

- left top
- center center
- right bottom
- 50% 50%

➤ **background-size**

Tamanho da imagem de fundo. Exemplos:

- cover → cobre tudo
- contain → cabe sem cortar
- 100px 200px
- auto

➤ **background-repeat**

Controla se a imagem se repete:

- repeat (padrão)
- no-repeat
- repeat-x
- repeat-y

➤ **background-attachment**

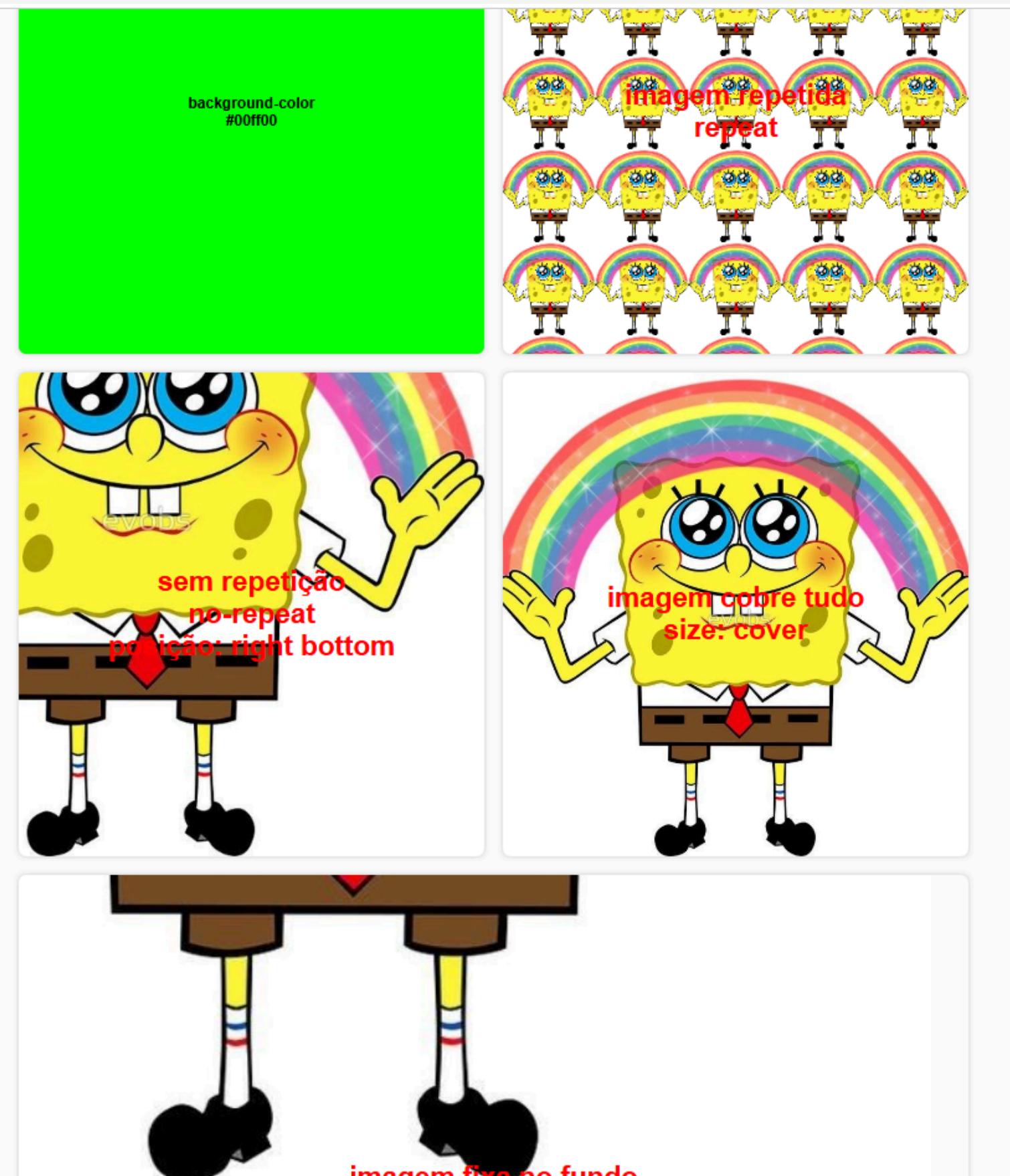
Controla o rolar da imagem:

- scroll (rola com a página)
- fixed (fica parada no fundo)
- local



CSS

```
.bg-color {  
background-color: #00ff00;  
color: black;  
}  
  
.bg-repeat {  
background-image: url('/imagens/bobesponja.jpg');  
background-repeat: repeat;  
background-color: red;  
color: black;  
}  
  
.bg-no-repeat {  
background-image: url('/imagens/bobesponja.jpg');  
background-repeat: no-repeat;  
background-position: right bottom;  
background-color: blue;  
color: black;  
}  
  
.bg-cover {  
background-image: url('/imagens/bobesponja.jpg');  
background-repeat: no-repeat;  
background-position: center;  
background-size: cover;  
background-color: yellow;  
color: black;  
}  
  
.bg-fixed {  
background-image: url('/imagens/bobesponja.jpg');  
background-repeat: no-repeat;  
background-position: center;  
background-size: contain;  
background-attachment: fixed;  
background-color: transparent;  
color: black;  
height: 200px;  
grid-column: span 2;  
}
```



HTML

```
<div class="grid">  
  <div class="box bg-color">  
    | background-color <br> #00ff00  
  </div>  
  
  <div class="box bg-repeat">  
    | imagem repetida <br> repeat  
  </div>  
  
  <div class="box bg-no-repeat">  
    | sem repetição <br> no-repeat <br> posição: right bottom  
  </div>  
  
  <div class="box bg-cover">  
    | imagem cobre tudo <br> size: cover  
  </div>  
  
  <div class="box bg-fixed">  
    | imagem fixa no fundo <br> attachment: fixed <br> size: contain  
  </div>  
</div>
```

Propriedades do CSS que ajudam a criar animações e transições

Propriedade	Tipo	O que faz
transition	Transição	Aplica uma transição suave entre valores de propriedades.
transition-property	Transição	Define quais propriedades sofrerão transição.
transition-duration	Transição	Define o tempo da transição.
transition-timing-function	Transição	Define a curva de aceleração da transição (ex: ease, linear, ease-in).
transition-delay	Transição	Define o tempo de espera antes da transição começar.

Propriedades do CSS que ajudam a criar animações e transições

Propriedade	Tipo	O que faz
animation	Animação	Propriedade abreviada para todas as propriedades de animação.
@keyframes	Animação	Define os passos de uma animação.
animation-name	Animação	Nome da animação criada com @keyframes.
animation-duration	Animação	Duração da animação.
animation-delay	Animação	Atraso antes de começar.
animation-iteration-count	Animação	Quantas vezes a animação vai se repetir.
animation-direction	Animação	Define a direção (normal, reverse, alternate).
animation-fill-mode	Animação	Define o estado final (forwards, backwards, both, none).
animation-timing-function	Animação	Aceleração da animação (ease, linear...).
animation-play-state	Animação	Controla se a animação está em execução ou pausada.

Propriedades do CSS que ajudam a criar animações e transições

animista

On-Demand CSS Animations Library

Animista is a CSS animation library and a place where you can play with a collection of ready-made CSS animations and download only those you will use.

 Animista



Animate.css | A cross-browser library of CSS animations.

Animate.css is a library of ready-to-use, cross-browser animations for you to use in your projects. Great for emphasis, home pages, sliders, and attention-guiding hints.

 animate.style

CSS-TRICKS

A website about building websites.

animation

The animation property in CSS can be used to animate many other CSS properties such as color, background-color, height, or width. Each animation needs to be

* CSS-Tricks / Sep 19, 2024

ver.

A collection of CSS3 powered hover effects

A collection of CSS3 powered hover effects to be applied to links, buttons, logos, SVG, featured...

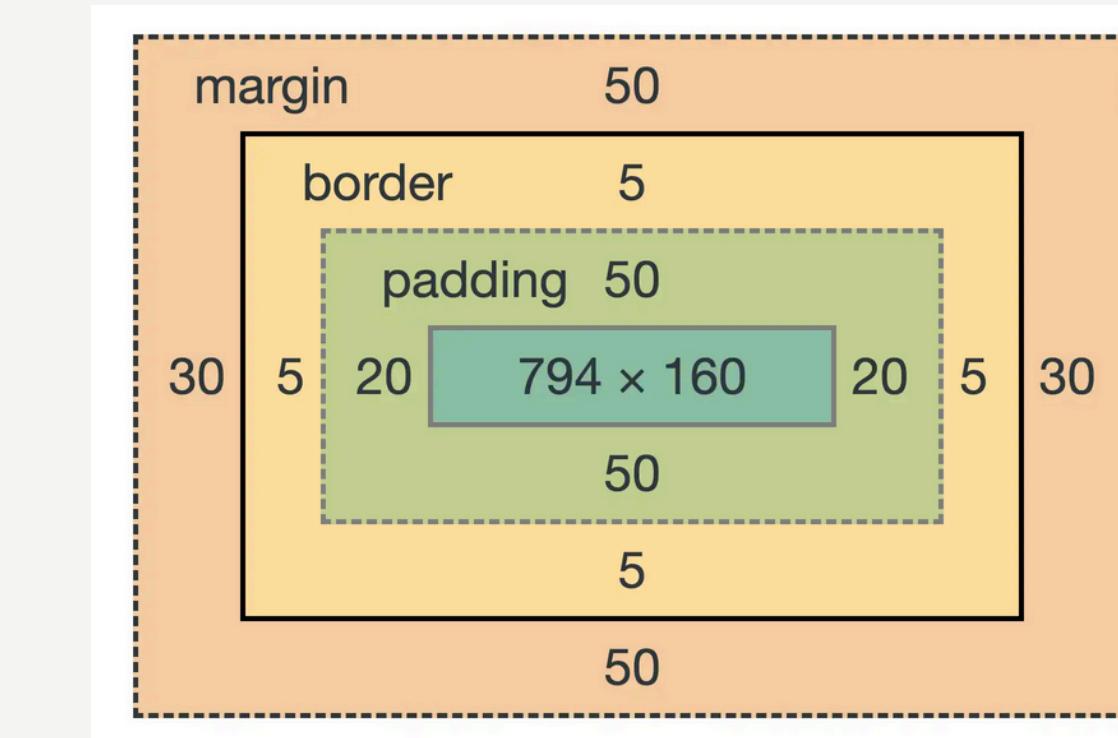
Hover.css

Pseudoclasses e Pseudoelementos

Tipo	Nome	O que faz
Pseudoclasse	:hover	Aplica estilo quando o mouse está sobre o elemento
Pseudoclasse	:active	Aplica estilo no momento em que o elemento está sendo clicado
Pseudoclasse	:focus	Aplica estilo quando o elemento recebe foco (ex: input)
Pseudoelemento	::before	Insere conteúdo antes do conteúdo do elemento
Pseudoelemento	::after	Insere conteúdo depois do conteúdo do elemento

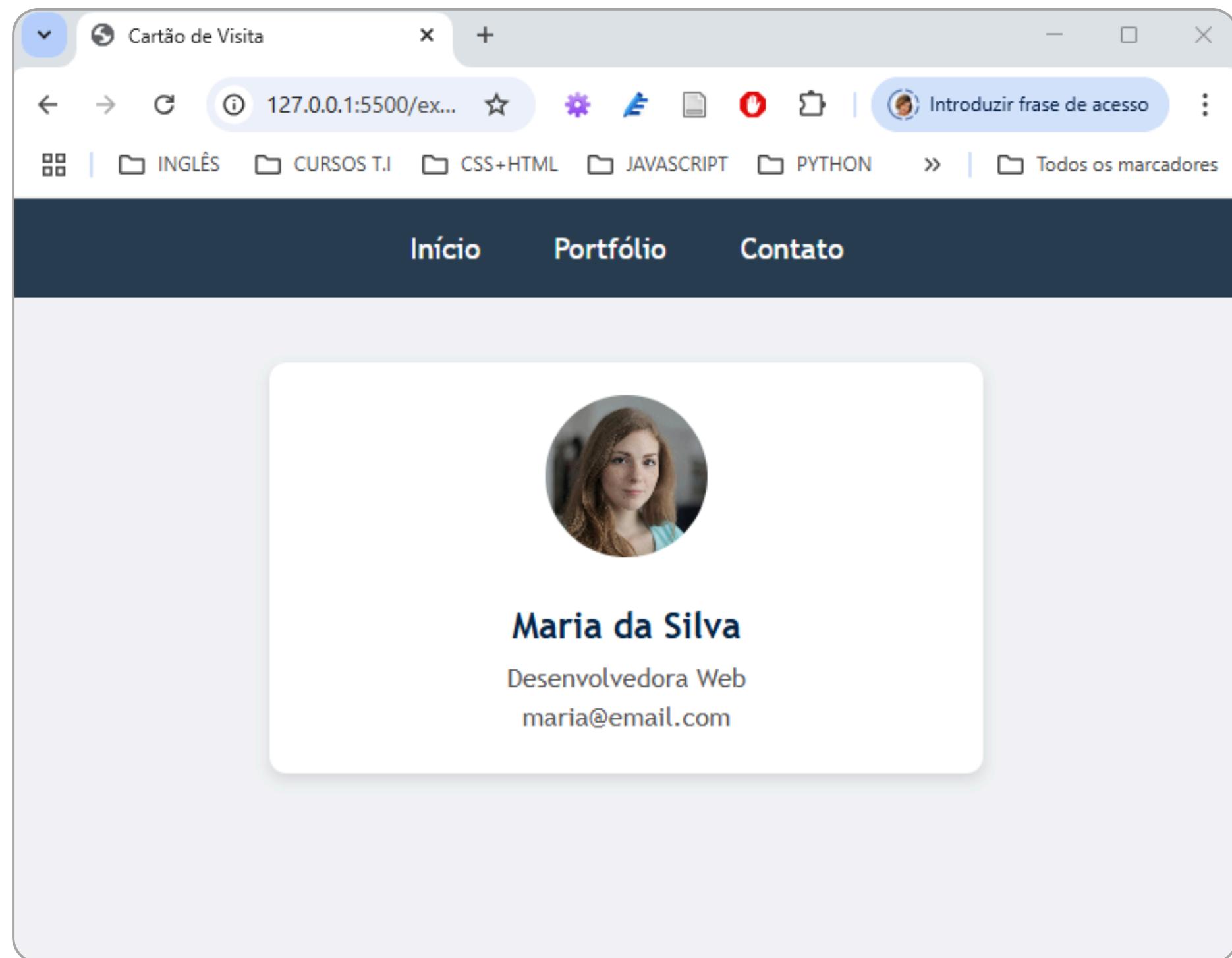
Box Model (Modelo de Caixa)

content	O conteúdo real da caixa (ex: texto ou imagem).
padding	Espaço interno, entre o conteúdo e a borda.
border	A borda da caixa (pode ter cor, espessura e estilo).
margin	Espaço externo, entre a borda da caixa e outros elementos.



1. Comportamento padrão do padding no tamanho da caixa
2. padding e margin com 1, 2, 3 e 4 valores
3. Corrigindo com box-sizing: border-box
4. Margens colapsadas
5. Comparação visual entre content, padding, border e margin

Exercício 4



Elementos Block, Inline e Inline-block

► No HTML, os elementos se comportam de maneiras diferentes na página.

Eles podem ser classificados em:

- Elementos do tipo Block
- Elementos do tipo Inline
- Elementos do tipo Inline-block (híbrido)

ELEMENTO TIPO BLOCK

ELEMENTO TIPO INLINE

ELEMENTO TIPO INLINE

ELEMENTO TIPO INLINE

ELEMENTO TIPO INLINE

ELEMENTO TIPO
INLINE-BLOCK



Elementos Inline, Block e Inline-Block

Tipos de elementos em HTML e seu comportamento padrão:

1. Elementos Block

- Ocupam toda a largura disponível da linha (100%)
- Sempre começam em uma nova linha
- Exemplo: <div>, <p>, <h1>,

2. Elementos Inline

- Ocupam apenas o espaço necessário para seu conteúdo
- Podem ficar lado a lado, na mesma linha
- Exemplo: , <a>, ,

3. Elementos Inline-Block

- Comportamento híbrido: ficam lado a lado (como inline)
- Mas aceitam largura, altura, margem e padding (como block)
- Exemplo: (por padrão), <button>

Elementos Block, Inline e Inline-block

Exemplo: Block

Bloco 1

Bloco 2

Bloco 3

```
<h1>Exemplo: Block</h1>
<div class="bloco">Bloco 1</div>
<div class="bloco">Bloco 2</div>
<div class="bloco">Bloco 3</div>
```

HTML

```
/* Estilo para elementos do tipo block */
.bloco {
background-color: #f8b400;
border: 2px solid #c97a00;
color: white;
padding: 20px;
margin-bottom: 10px;
```

CSS

Elementos Block, Inline e Inline-block

Exemplo: Inline

Inline 1

Inline 2

Inline 3

```
<h1>Exemplo: Inline</h1>
<div class="linha">Inline 1</div>
<div class="linha">Inline 2</div>
<div class="linha">Inline 3</div>
```

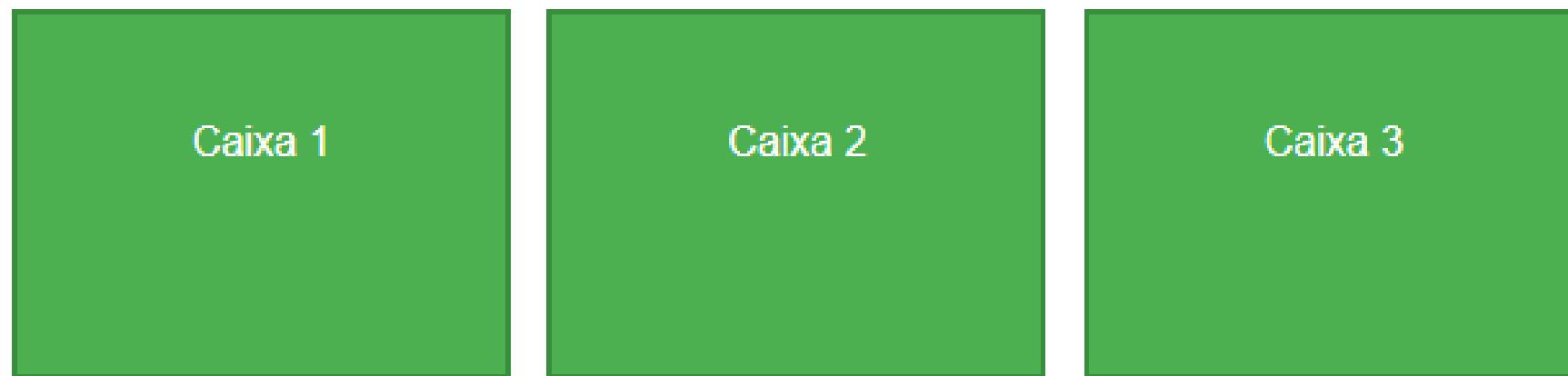
HTML

```
/* Estilo para elementos do tipo inline */
.linha {
background-color: #2196f3;
color: white;
border: 2px solid #0b79d0;
padding: 10px;
margin-right: 10px;
display: inline;
}
```

CSS

Elementos Block, Inline e Inline-block

Exemplo: Inline-block



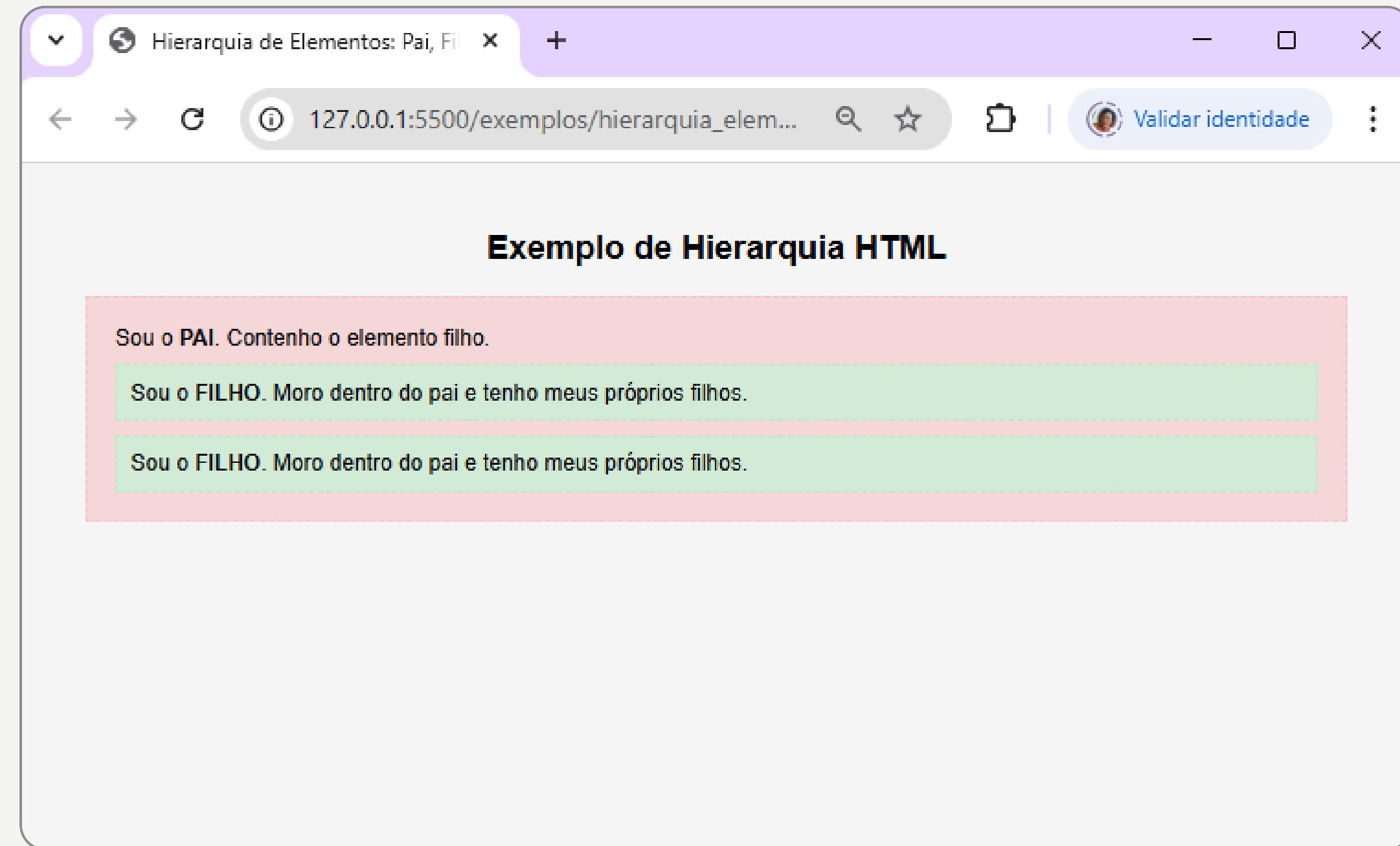
```
<h1>Exemplo: Inline-block</h1>
<div class="meio">Caixa 1</div>
<div class="meio">Caixa 2</div>
<div class="meio">Caixa 3</div>
```

HTML

```
/* Estilo para elementos inline-block */
.meio {
background-color: #4CAF50;
color: white;
border: 2px solid #388E3C;
padding: 20px;
margin: 5px;
width: 150px;
height: 100px;
text-align: center;
line-height: 60px;
display: inline-block;
}
```

CSS

Hierarquia de Elementos: Pai e Filho



Elementos Flutuantes

► O que é Float?

A propriedade float permite que elementos "flutuem" à esquerda ou à direita, saindo do fluxo normal da página.

O que é Float?

Caixa normal

Texto ao redor da caixa flutuante.

Caixa flutuante

```
<!-- Float simples -->
<div class="container">
  <h2>O que é Float?</h2>
  <div class="box bg-lightblue">Caixa normal</div>
  <div class="box float-left bg-lightgreen">Caixa flutuante</div>
  <p>Texto ao redor da caixa flutuante.</p>
  <div class="clear"></div>
</div>
```

HTML

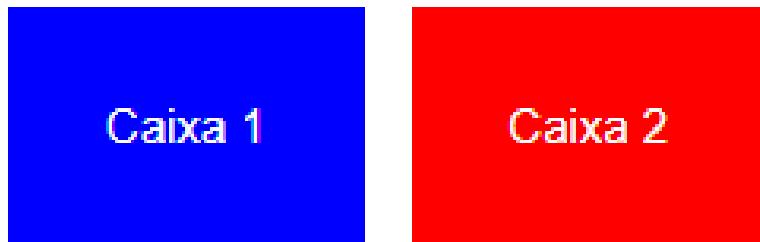
```
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}

/* Flutuação */
.float-left {
  float: left;
}
```

CSS

Elementos Flutuantes

Exemplo Básico de Float



```
<!-- Duas caixas lado a lado -->
<div class="container">
  <h2>Exemplo Básico de Float</h2>
  <div class="box float-left bg-blue">Caixa 1</div>
  <div class="box float-left bg-red">Caixa 2</div>
  <div class="clear"></div>
</div>
```

HTML

```
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}
```

```
/* Flutuação */
.float-left {
  float: left;
}
```

CSS

Elementos Flutuantes

Como o Float Afeta o Fluxo

Float Left

O texto se ajusta ao lado da caixa flutuante. Isso pode causar comportamentos inesperados se não for usado corretamente.

```
<!-- Float com texto fluindo ao lado -->
<div class="container">
  <h2>Como o Float Afeta o Fluxo</h2>
  <div class="box float-left bg-orange">Float Left</div>
  <p>O texto se ajusta ao lado da caixa flutuante. Isso pode causar comportamentos inesperados se não for usado corretamente.</p>
  <div class="clear"></div>
</div>
```

HTML

```
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}
```

/* Flutuação */

```
.float-left {
  float: left;
}
```

css

```
/* Limpeza */
.clear {
  clear: both;
}
```

Elementos Flutuantes

Float Esquerda e Direita



```
<!-- Float em direções opostas -->
<div class="container">
  <h2>Float Esquerda e Direita</h2>
  <div class="box float-left bg-teal">Esquerda</div>
  <div class="box float-right bg-purple">Direita</div>
  <p>Texto entre as caixas.</p>
  <div class="clear"></div>
</div>
```

HTML

```
/* Containers */
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}

/* Flutuação */
.float-left {
  float: left;
}

.float-right {
  float: right;
}

/* Limpeza */
.clear {
  clear: both;
}
```

CSS

Elementos Flutuantes

Problemas Comuns

Flutuante

O container acima não envolve a caixa flutuante porque falta a limpeza do float.

Corrigindo com Overflow

Flutuante

Agora o container envolve corretamente a caixa usando `overflow: auto;`

```
<!-- Problema clássico do float: container colapsado -->
<div class="container">
  <h2>Problemas Comuns</h2>
  <div class="sem-corrigir">
    <div class="box float-left bg-coral">Flutuante</div>
  </div>
  <p>O container acima não envolve a caixa flutuante porque falta a limpeza do float.</p>
<h2>Corrigindo com Overflow</h2>
<div class="com-overflow">
  <div class="box float-left bg-coral">Flutuante</div>
</div>
<p>Agora o container envolve corretamente a caixa usando <code>overflow: auto</code>.</p>
</div>
```

HTML

```
/* Problemas comuns e soluções */
.sem-corrigir {
  background-color: #ddd;
  border: 2px dashed black;
}

.com-overflow {
  background-color: #ddd;
  overflow: auto;
}
```

CSS

```
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}

/* Flutuação */
.float-left {
  float: left;
}
```

Elementos Flutuantes

Exemplo Completo: Galeria Flutuante



Texto depois da galeria flutuante.

```
<!-- Galeria flutuante -->
<div class="container">
  <h2>Exemplo Completo: Galeria Flutuante</h2>
  <div class="galeria">
    <div class="box float-left bg-item1">1</div>
    <div class="box float-left bg-item2">2</div>
    <div class="box float-left bg-item3">3</div>
  </div>
  <p>Texto depois da galeria flutuante.</p>
</div>
```

HTML

```
.container {
  border: 2px solid red;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 10px;
}

/* Flutuação */
.float-left {
  float: left;
}
```

CSS

Exercício 5

A screenshot of a web browser window titled "Galeria Simples". The address bar shows the URL "127.0.0.1:5500/exercicios/exercicio_5.html". The main content area displays a title "Galeria de Produtos" followed by three product cards. Each card contains a small image of a brown backpack and a caption. The first card is labeled "Produto 1" and has the text "Texto de exemplo do Produto 1.". The second card is labeled "Produto 2" and has the text "Texto de exemplo do Produto 2.". The third card is labeled "Produto 3" and has the text "Texto de exemplo do Produto 3.".

Galeria de Produtos

 **Produto 1**
Texto de exemplo do Produto 1.

 **Produto 2**
Texto de exemplo do Produto 2.

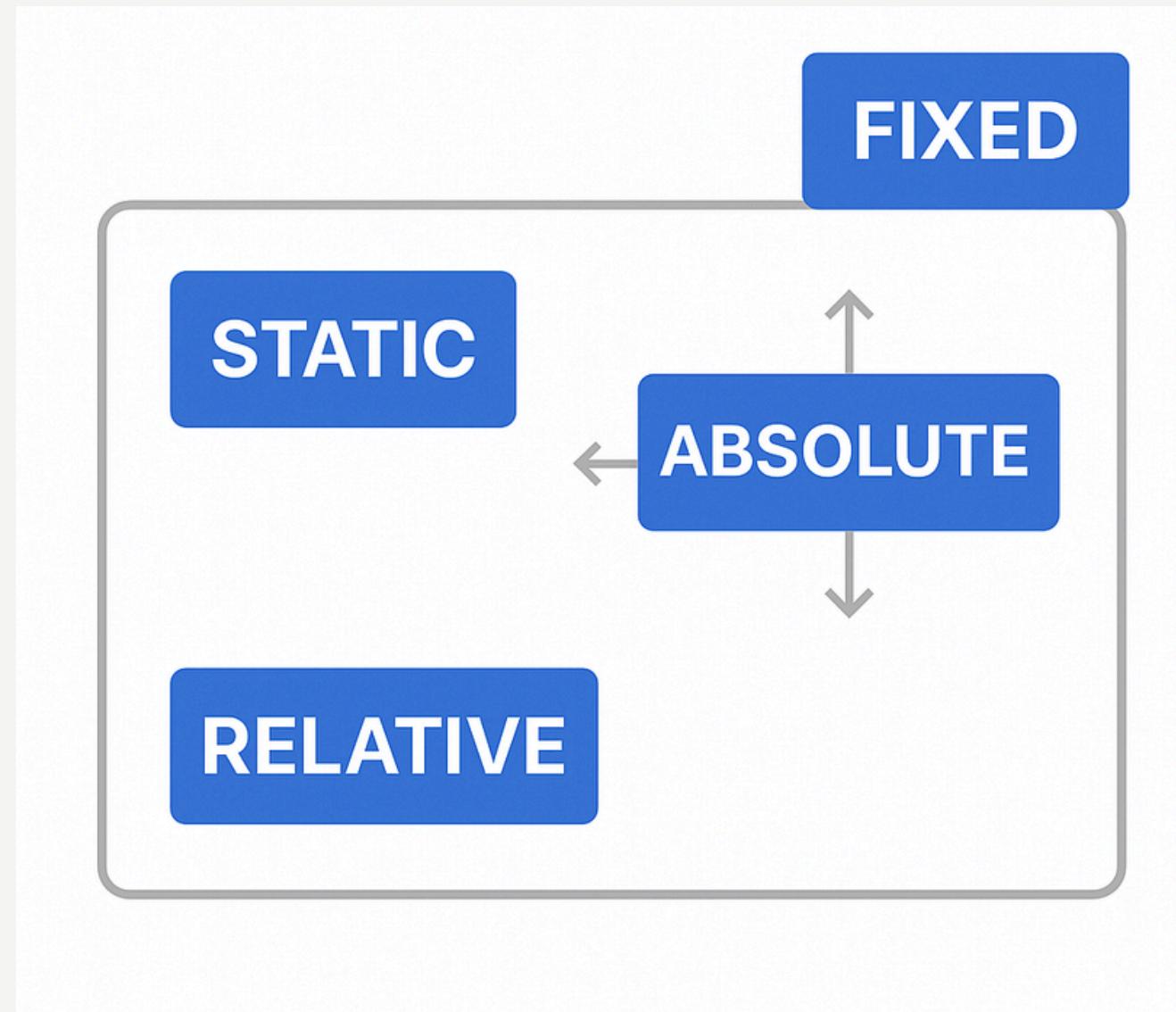
 **Produto 3**
Texto de exemplo do Produto 3.

Posicionamiento de Elementos

Tipos de posicionamiento no CSS:

- static (estático)
- relative (relativo)
- absolute (absoluto)
- fixed (fixo)

Propriedade - Position



Posicionamento de Elementos

position: static (padrão)

- É o posicionamento normal dos elementos na página.
- O navegador posiciona os elementos seguindo o fluxo natural do documento, ou seja, um após o outro, de cima para baixo.
- Não aceita propriedades como top, left, right ou bottom.
- Use quando não precisa alterar a posição do elemento.

Posicionamento de Elementos

position: relative

- O elemento é posicionado em relação à sua posição original no fluxo do documento.
- Você pode usar top, left, right e bottom para mover o elemento a partir dessa posição, sem afetar o espaço reservado para ele.
- Ou seja, o elemento “sai” visualmente do lugar original, mas o espaço que ele ocupa no layout continua o mesmo.
- Útil para fazer pequenos ajustes sem quebrar o fluxo da página.

Posicionamento de Elementos

position: absolute

- O elemento é removido do fluxo normal da página e posicionado em relação ao seu ancestral posicionado mais próximo (que tenha position: relative, absolute ou fixed).
- Usa as propriedades top, left, right, bottom para definir sua posição exatamente dentro desse ancestral.
- Como está fora do fluxo, não ocupa espaço e pode sobrepor outros elementos.
- Muito usado para criar layouts flexíveis, modais, tooltips, etc.

Posicionamento de Elementos

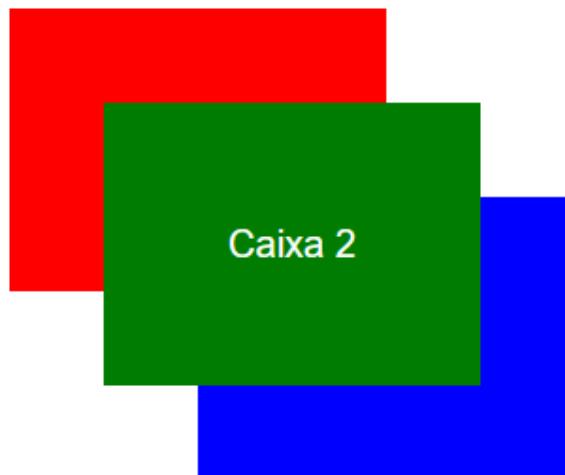
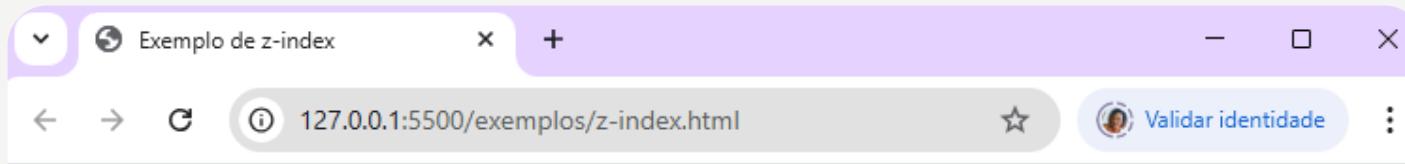
position: fixed

- Semelhante ao absolute, mas o elemento é posicionado em relação à janela do navegador (viewport).
- Fica fixo na tela, independente do quanto você role a página.
- Usado para menus fixos, botões flutuantes, cabeçalhos que acompanham a rolagem.
- Ideal para elementos que devem sempre estar visíveis.

Z-index

► O z-index define a ordem de empilhamento de elementos HTML que se sobrepõem. Elementos com z-index maior ficam mais acima, ou seja, visíveis na frente de outros.

- Mas atenção: só funciona em elementos posicionados, ou seja, com position: relative, absolute, fixed ou sticky.



```
body {  
    font-family: Arial, sans-serif;  
    padding: 20px;  
}  
  
.caixa {  
    width: 200px;  
    height: 150px;  
    position: absolute;  
    color: white;  
    font-size: 20px;  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.caixa1 {  
    background-color: red;  
    top: 50px;  
    left: 50px;  
    z-index: 1;  
}  
  
.caixa2 {  
    background-color: green;  
    top: 100px;  
    left: 100px;  
    z-index: 2; /* Fica por cima */  
}  
  
.caixa3 {  
    background-color: blue;  
    top: 150px;  
    left: 150px;  
    z-index: 0; /* Fica por baixo das outras */  
}
```

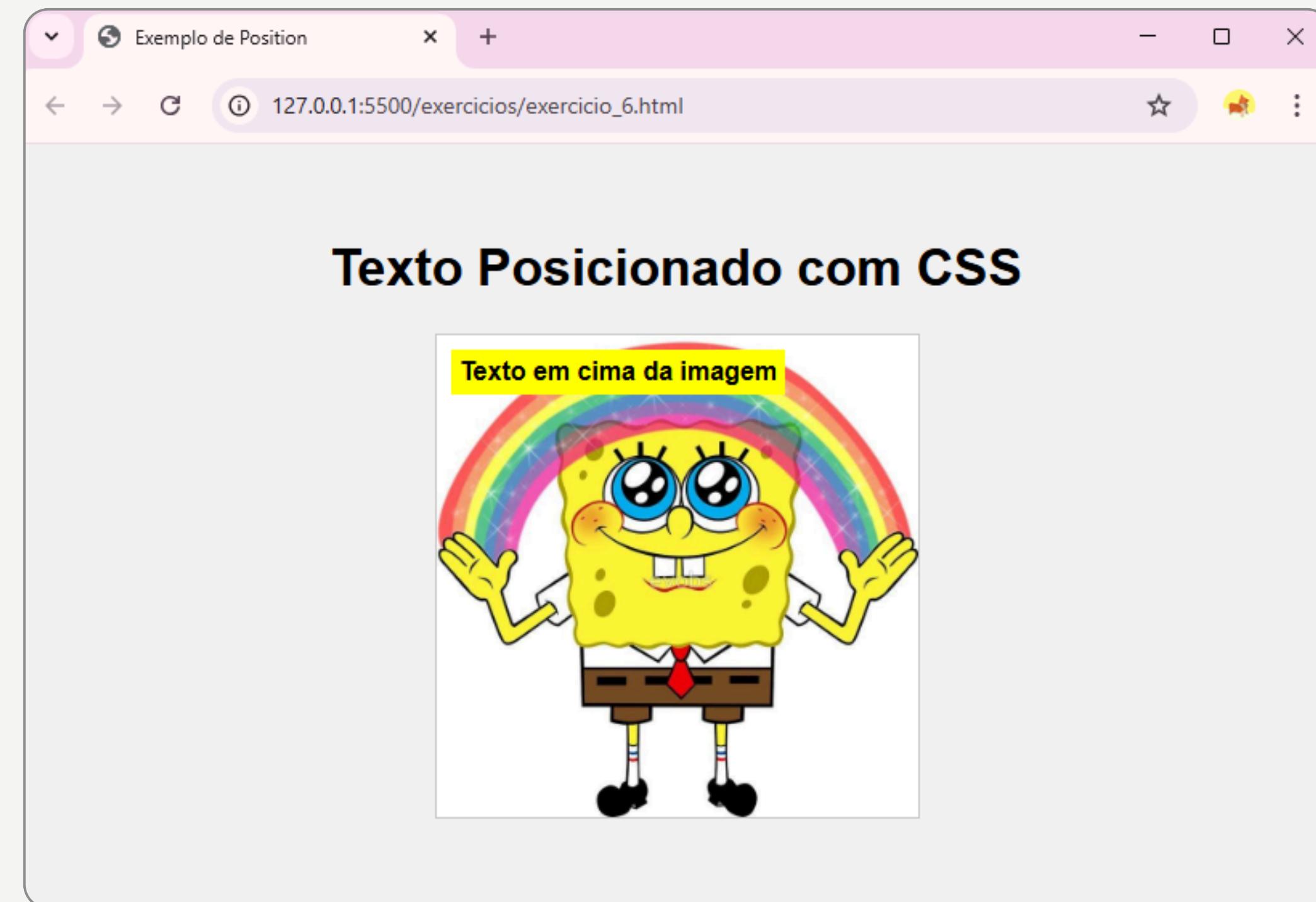
HTML

```
<div class="caixa caixa1">Caixa 1</div>  
<div class="caixa caixa2">Caixa 2</div>  
<div class="caixa caixa3">Caixa 3</div>
```

CSS

```
.caixa3 {  
    background-color: blue;  
    top: 150px;  
    left: 150px;  
    z-index: 0; /* Fica por baixo das outras */  
}
```

Exercício 6



CSS Grid Layout

► **Grid** é um sistema de layout bidimensional que nos permite organizar elementos em linhas e colunas, formando uma grade (grid) superflexível.

- O CSS Grid trabalha nas duas direções ao mesmo tempo.

► **O que é fr?**

- fr significa fração do espaço disponível.
- Quando usamos 1fr, dizemos: "ocupe uma parte do espaço disponível".
- Se usarmos 2fr 1fr, por exemplo, a primeira coluna ocupará o dobro de espaço da segunda.

► **O que é gap?**

- gap define o espaçamento entre as colunas e as linhas do grid.

Display: grid

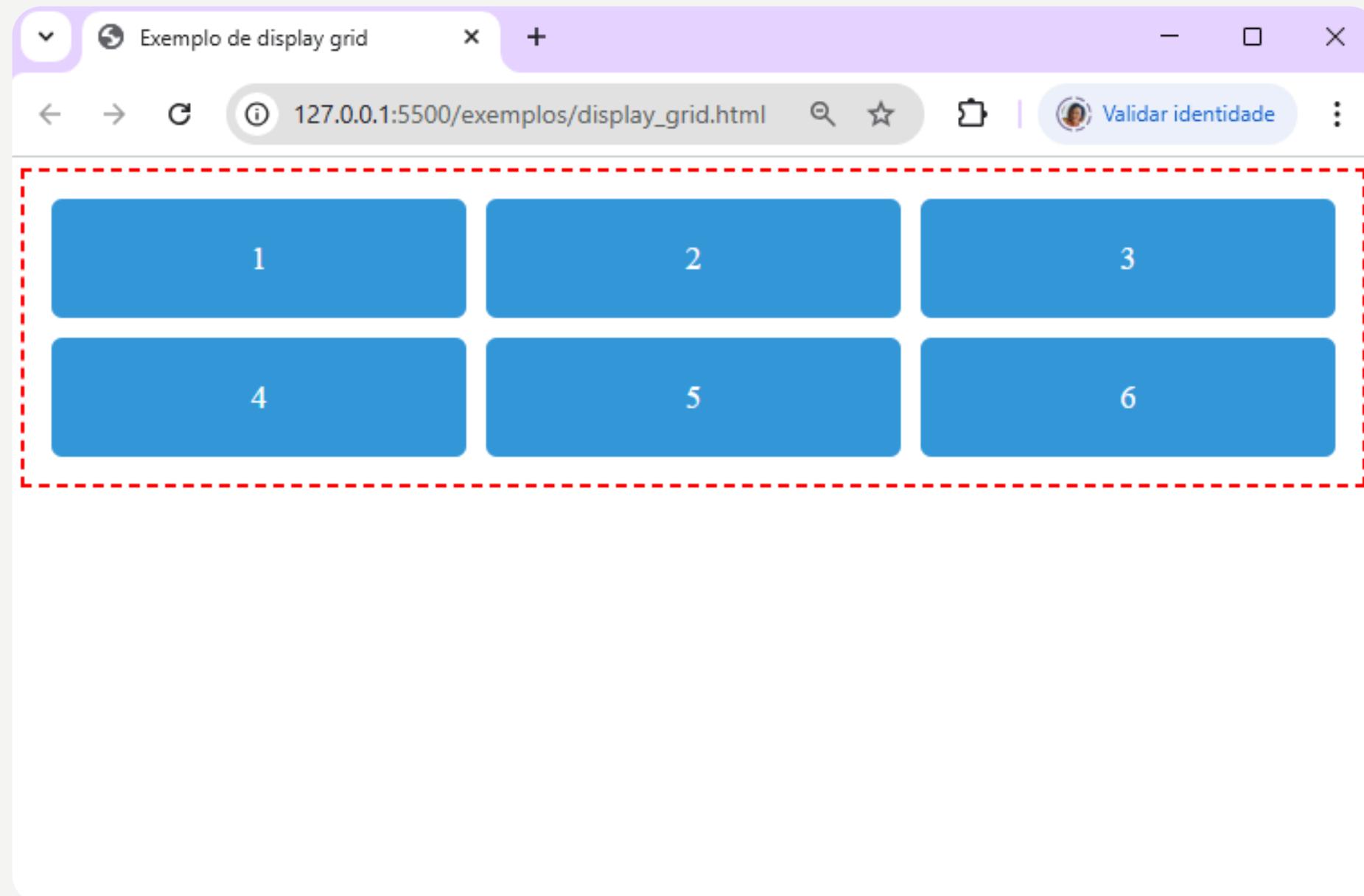
➤ A propriedade **display: grid** cria um container de grid em bloco, ou seja, o elemento que recebe display: grid vira um elemento de bloco (block-level) e seus filhos se tornam itens de grade.

Comportamento:

- O container se expande em largura como um bloco (div, section, etc).
- Os filhos diretos são colocados dentro de uma grade definida por grid-template-columns, grid-template-rows, etc.

HTML

```
<div class="grid-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
</div>
```



CSS

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 15px;
  padding: 20px;
  border: 3px dashed red;
}

.item {
  background-color: #3498db;
  color: white;
  font-size: 24px;
  text-align: center;
  padding: 30px 0;
  border-radius: 8px;
}
```

Display: inline-grid

- Cria um grid que se comporta como um elemento inline – ou seja, ele se alinha com outros elementos em linha, mas ainda mantém o comportamento interno de grid.

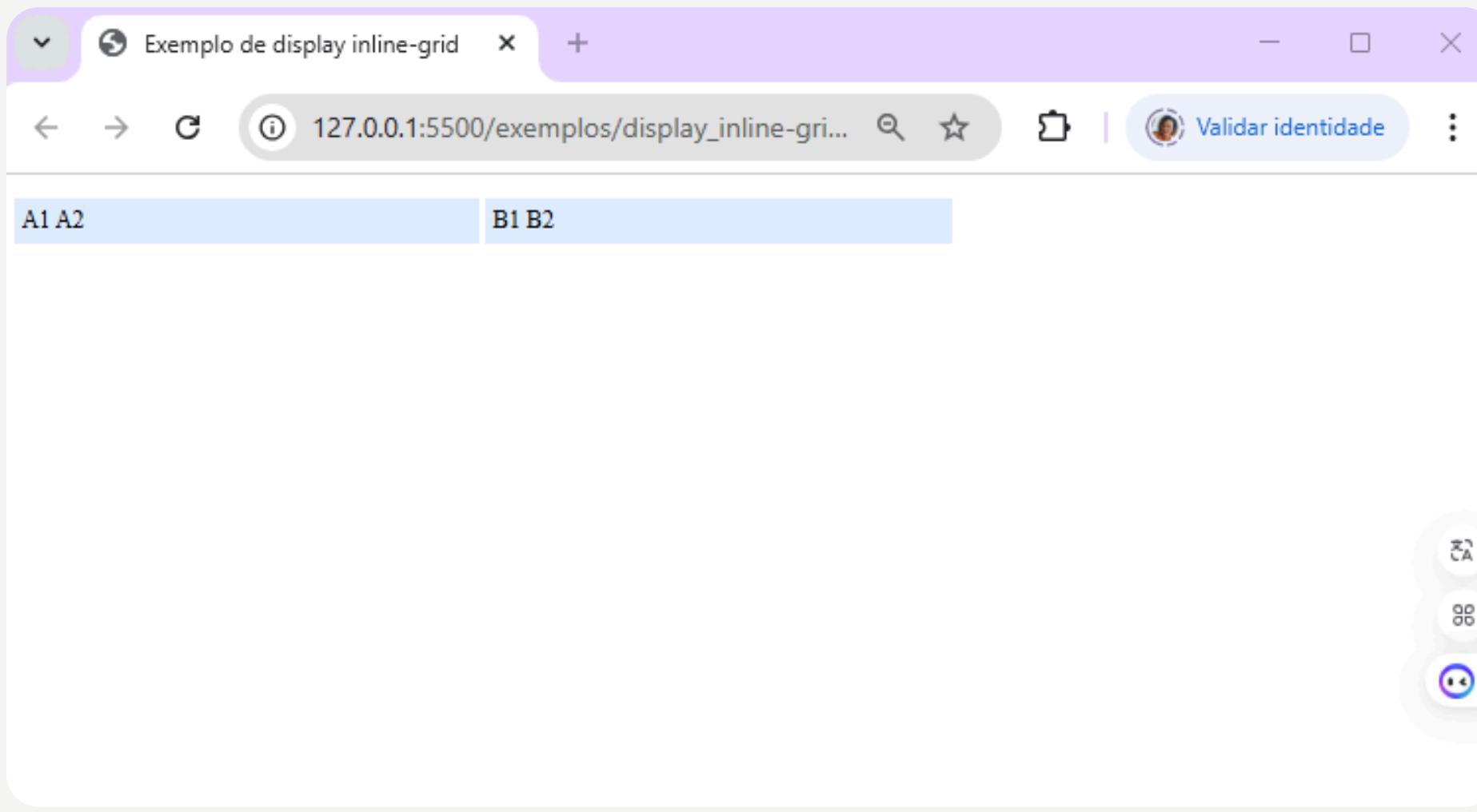
Diferença para grid:

- **grid** - ocupa toda a linha (comportamento de bloco).
- **inline-grid** - só ocupa o espaço necessário para seu conteúdo (comportamento inline).

Uso comum: quando você quer posicionar vários grids na mesma linha.

HTML

```
<p>
<span class="inline-grid">A1 A2</span>
<span class="inline-grid">B1 B2</span>
</p>
```

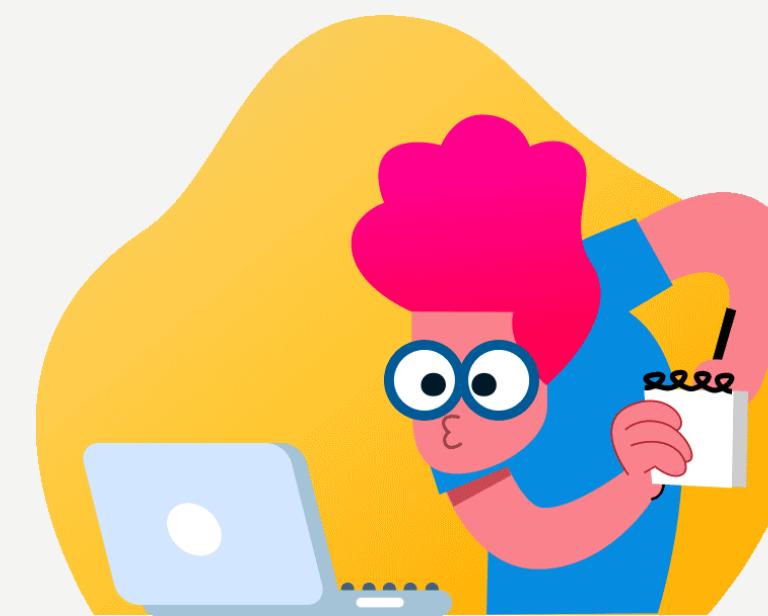


CSS

```
.inline-grid {
  display: inline-grid;
  grid-template-columns: 150px 150px;
  gap: 5px;
  background-color: #ddeeef;
  padding: 5px;
}
```

CSS Grid Layout

Propriedade	Função
grid-template-columns / rows	Define o tamanho e número de colunas/linhas
grid-gap, row-gap, column-gap	Espaçamento entre linhas/colunas
grid-column, grid-row, grid-area	Posicionamento manual dos itens na grade
justify-self, align-self	Alinhamento individual de cada item
grid-auto-flow: row/column	Controla o preenchimento automático do grid

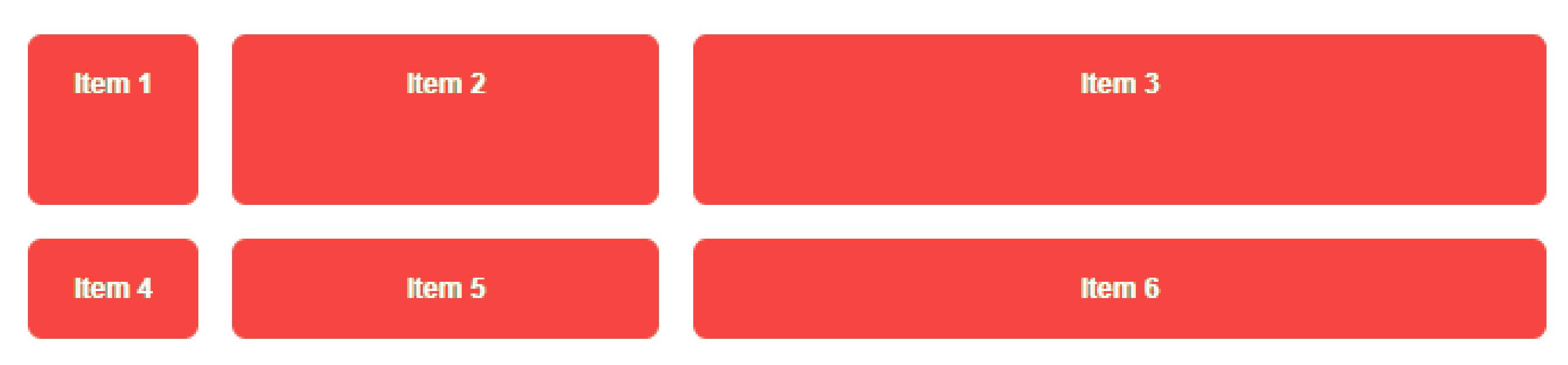


CSS Grid Layout

► Criando Linhas e Colunas

- grid-template-columns
- grid-template-rows

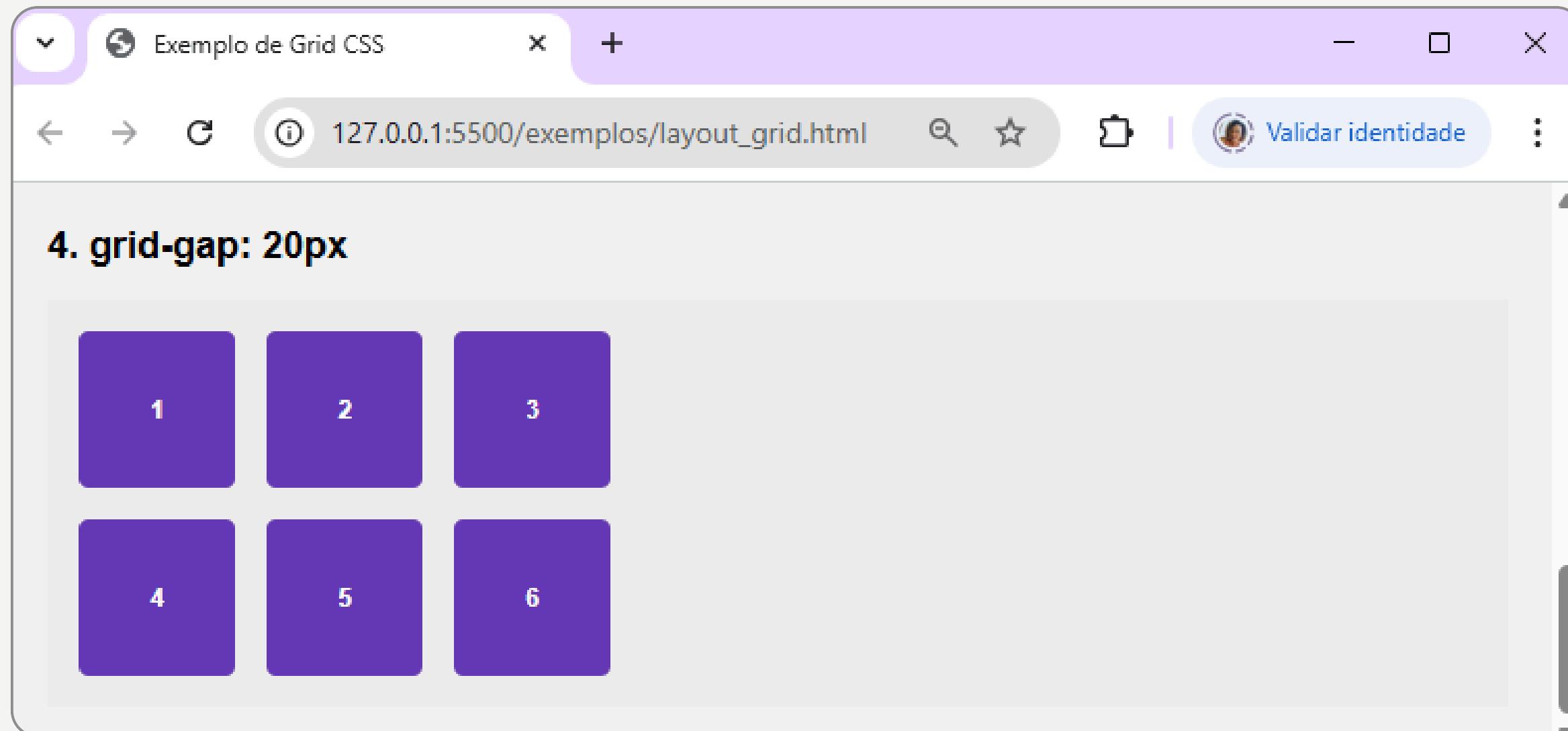
3. grid-template-columns, grid-template-rows, gap



CSS Grid Layout

➤ Espaçamento entre Linhas e Colunas

- grid-gap
- row-gap e column-gap

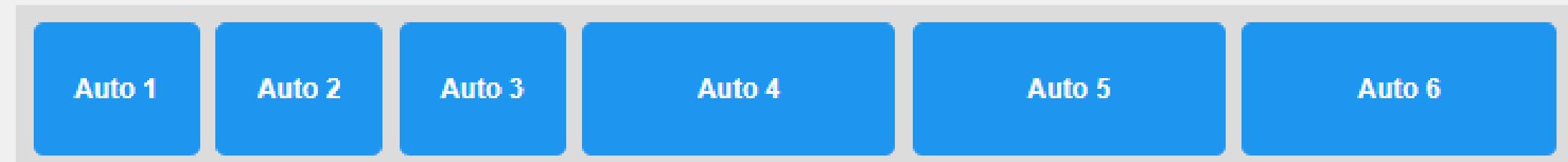


CSS Grid Layout

➤ Posicionamento Manual dos Itens

- grid-column, grid-row
- grid-area
- Justify-self e Align-self
- Grid Auto Flow

2. Exemplo de grid-auto-flow: column



Exemplo de Grid CSS

127.0.0.1:5500/exemplos/layout_grid.html

1. Exemplo de Grid com posicionamento manual

grid-column + grid-row

grid-area

justify-self + align-self

grid-row: 1 / 3

Align e Justify Content

- **Essas propriedades controlam o posicionamento do conjunto de itens dentro do container (pai) quando sobra espaço.**



Diferença entre eles:

Propriedade	Atua na direção	Exemplo comum
justify-content	Horizontal (eixo X)	alinha o conjunto de colunas no eixo horizontal
align-content	Vertical (eixo Y)	alinha o conjunto de linhas no eixo vertical



Possíveis valores: start, center, end, space-between, space around e space evenly

The screenshot shows a browser window with six examples of justify-content values:

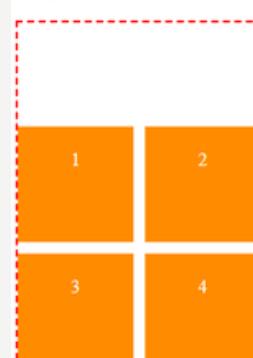
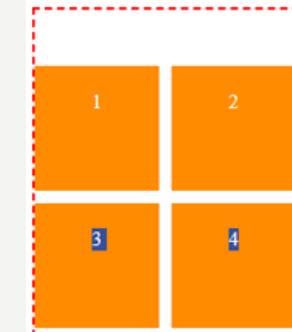
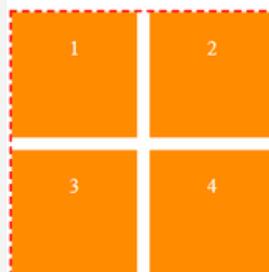
- justify-content: start**: Items are aligned to the top-left corner.
- justify-content: center**: Items are centered horizontally.
- justify-content: end**: Items are aligned to the top-right corner.
- justify-content: space-between**: Items have equal width and gaps between them.
- justify-content: space-around**: Items have gaps around them, with the total width of the gaps being equal to the width of the items.
- justify-content: space-evenly**: Items have equal widths and gaps between them, with the total width of the gaps being equal to the width of the items.

CSS

```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 100px);  
    border: 2px dashed red;  
    height: 200px;  
    gap: 10px;  
    margin-bottom: 40px;  
}  
  
.item {  
    background-color: steelblue;  
    color: white;  
    text-align: center;  
    padding: 20px 0;  
}
```

HTML

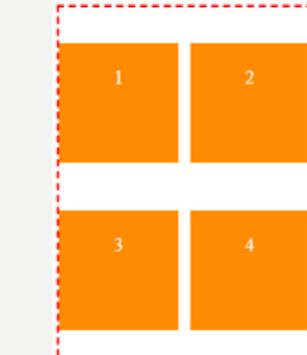
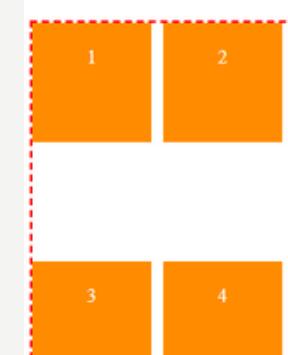
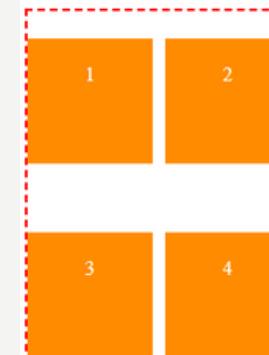
```
<h2>justify-content: start</h2>  
<div class="container" style="justify-content: start;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>  
  
<h2>justify-content: center</h2>  
<div class="container" style="justify-content: center;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>  
  
<h2>justify-content: end</h2>  
<div class="container" style="justify-content: end;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>  
  
<h2>justify-content: space-between</h2>  
<div class="container" style="justify-content: space-between;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>  
  
<h2>justify-content: space-around</h2>  
<div class="container" style="justify-content: space-around;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>  
  
<h2>justify-content: space-evenly</h2>  
<div class="container" style="justify-content: space-evenly;">  
    <div class="item">1</div><div class="item">2</div><div class="item">3</div>  
</div>
```



```
<h2>align-content: start</h2>
<div class="container" style="align-content: start;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```

```
<h2>align-content: center</h2>
<div class="container" style="align-content: center;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```

```
<h2>align-content: end</h2>
<div class="container" style="align-content: end;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```



```
<h2>align-content: space-around</h2>
<div class="container" style="align-content: space-around;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```

```
<h2>align-content: space-between</h2>
<div class="container" style="align-content: space-between;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```

```
<h2>align-content: space-evenly</h2>
<div class="container" style="align-content: space-evenly;">
  <div class="item">1</div><div class="item">2</div>
  <div class="item">3</div><div class="item">4</div>
</div>
```

```
.container {
  display: grid;
  grid-template-columns: 100px 100px;
  grid-template-rows: 100px 100px;
  gap: 10px;
  height: 300px;
  border: 2px dashed red;
  margin-bottom: 40px;
}
```

```
.item {
  background-color: darkorange;
  color: white;
  text-align: center;
  padding: 20px 0;
}
```

Align e Justify Items

As propriedades controlam o alinhamento do conteúdo interno de cada célula/item dentro do grid.



Diferença entre eles:

Propriedade	Atua na direção	Exemplo comum
justify-items	Horizontal (eixo X)	alinha o conteúdo horizontalmente dentro de cada item.
align-items	Vertical (eixo Y)	alinha o conteúdo verticalmente dentro de cada item.



Possíveis valores: start, center, end e stretch

HTML

```
<h1>justify-items</h1>
```

```
<h2>justify-items: start</h2>
<div class="grid" style="justify-items: start;">
  <div class="item">Item</div>
  <div class="item">Item</div>
  <div class="item">Item</div>
</div>
```

```
<h2>justify-items: center</h2>
<div class="grid" style="justify-items: center;">
  <div class="item">Item</div>
  <div class="item">Item</div>
  <div class="item">Item</div>
</div>
```

```
<h2>justify-items: end</h2>
<div class="grid" style="justify-items: end;">
  <div class="item">Item</div>
  <div class="item">Item</div>
  <div class="item">Item</div>
</div>
```

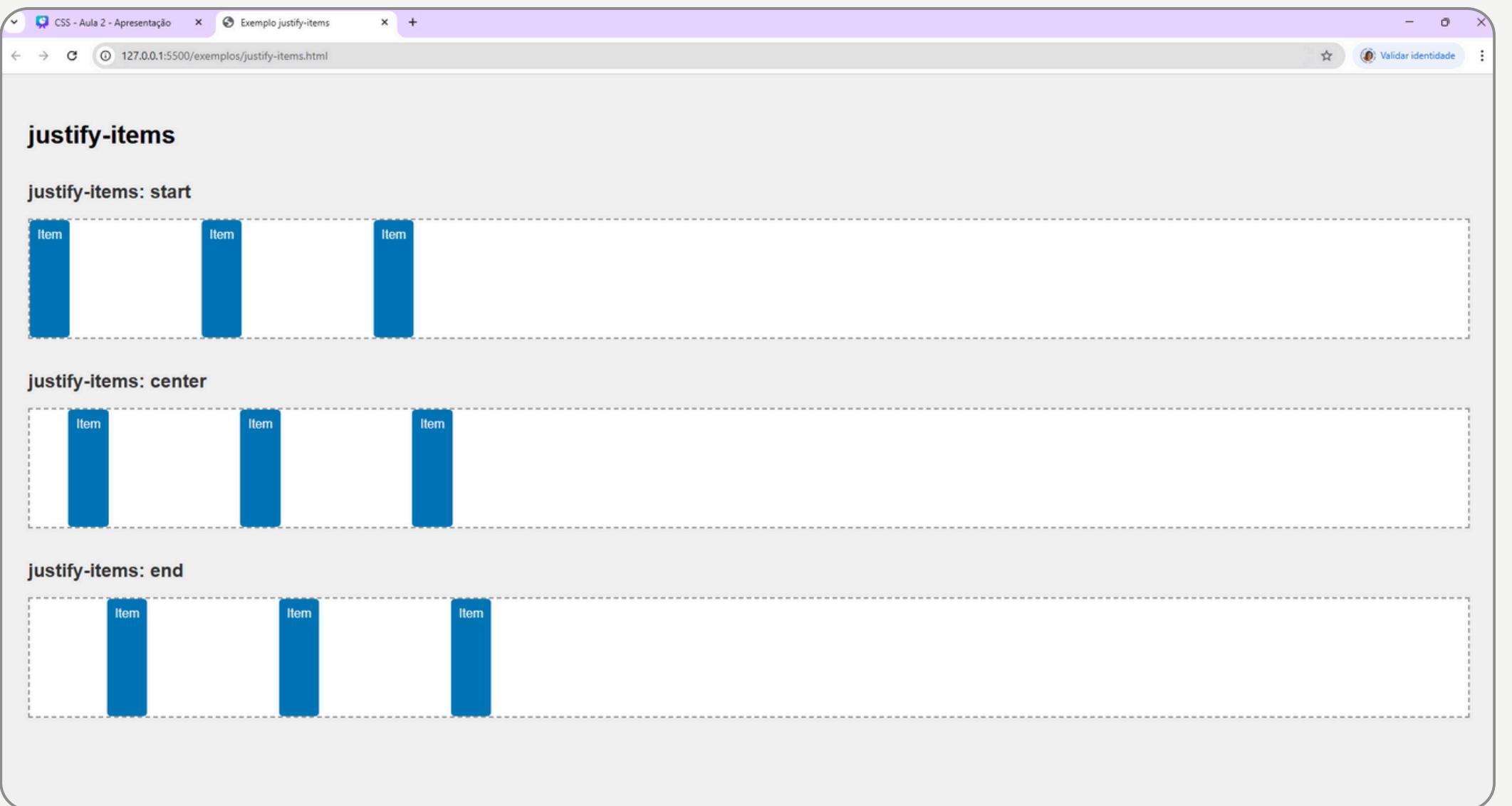
CSS

```
body {
  font-family: Arial, sans-serif;
  padding: 30px;
  background-color: #f0f0f0;
}

h2 {
  margin-top: 40px;
  color: #333;
}

.grid {
  display: grid;
  grid-template-columns: repeat(3, 150px);
  gap: 10px;
  height: 150px;
  background-color: #fff;
  border: 2px dashed #999;
  margin-bottom: 40px;
}

.item {
  background-color: #0077b6;
  color: white;
  padding: 10px;
  border-radius: 5px;
}
```



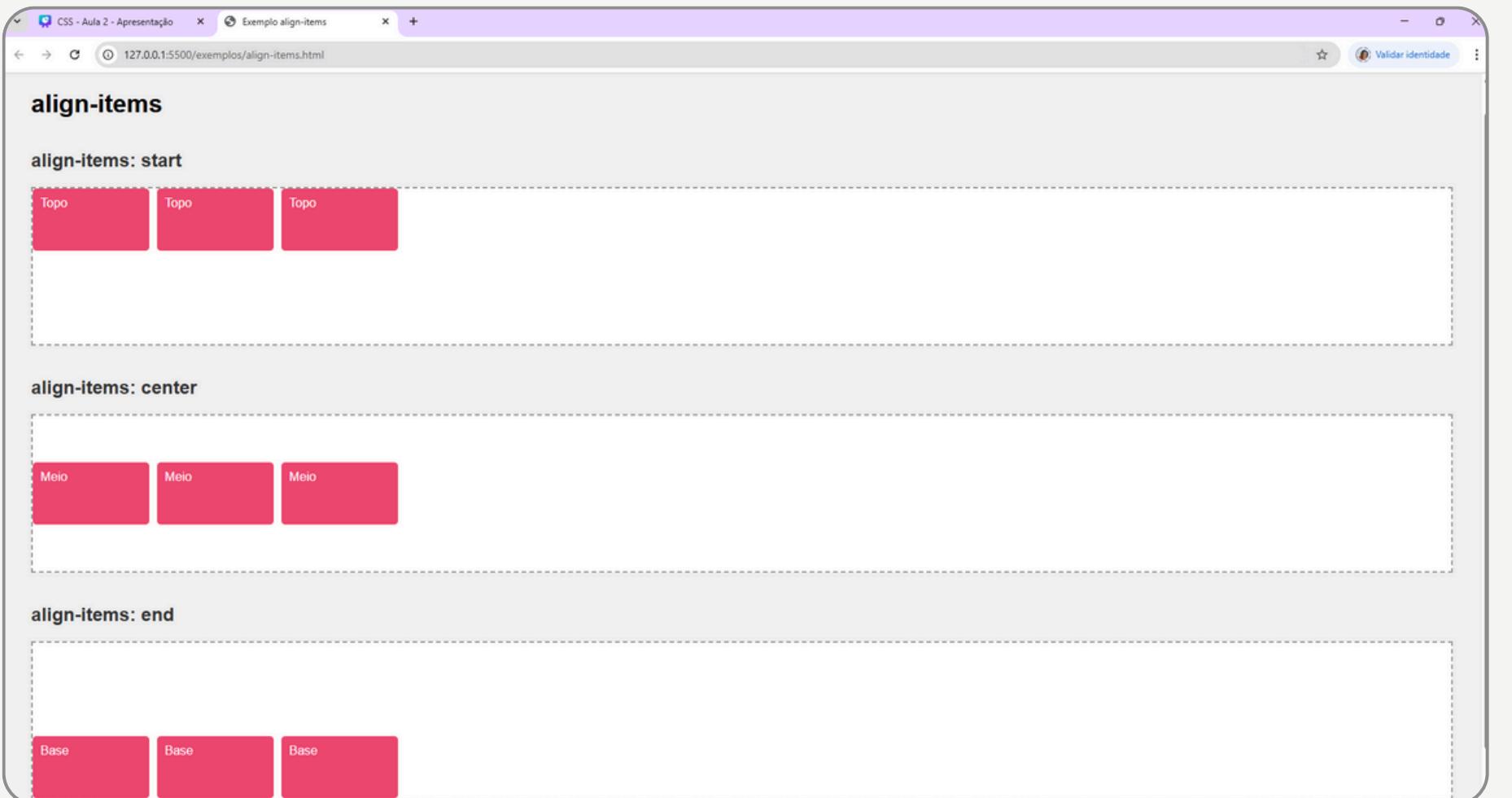
HTML

```
<h1>align-items</h1>

<h2>align-items: start</h2>
<div class="grid" style="align-items: start;">
  <div class="item">Topo</div>
  <div class="item">Topo</div>
  <div class="item">Topo</div>
</div>
```

```
<h2>align-items: center</h2>
<div class="grid" style="align-items: center;">
  <div class="item">Meio</div>
  <div class="item">Meio</div>
  <div class="item">Meio</div>
</div>
```

```
<h2>align-items: end</h2>
<div class="grid" style="align-items: end;">
  <div class="item">Base</div>
  <div class="item">Base</div>
  <div class="item">Base</div>
</div>
```



CSS

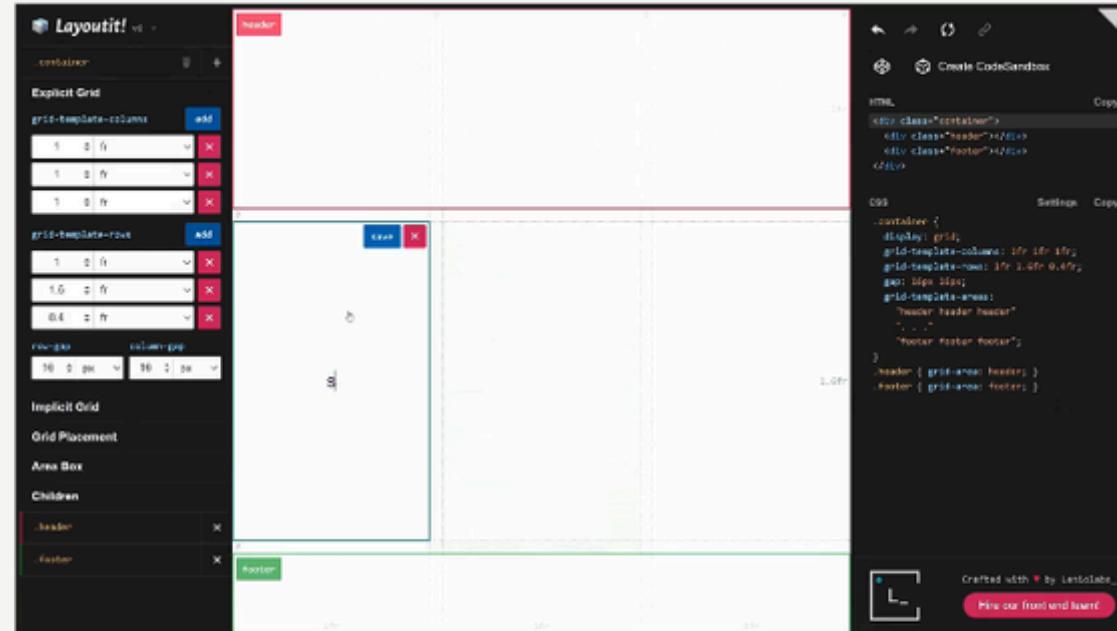
```
body {
  font-family: Arial, sans-serif;
  padding: 30px;
  background-color: #f0f0f0;
}
```

```
h2 {
  margin-top: 40px;
  color: #333;
}
```

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 150px);
  gap: 10px;
  height: 200px;
  background-color: #fff;
  border: 2px dashed #999;
  margin-bottom: 40px;
}
```

```
.item {
  background-color: #ef476f;
  color: white;
  padding: 10px;
  border-radius: 5px;
  height: 60px;
}
```

Ferramentas CSS Grid



Layoutit Grid — CSS Grids layouts made easy!

Quickly design web layouts, and get HTML and CSS code. Learn CSS Grid visually build web layouts with our interactive CSS Grid Generator.

layoutit.com



CSS Grid Generator

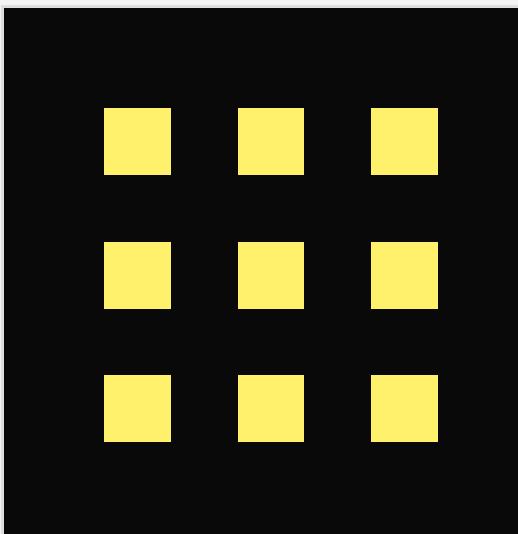
netlify.app



CSS Grid Layout Guide

Our comprehensive guide to CSS grid, focusing on all the settings both for the grid parent container and the grid child elements.

* CSS-Tricks / Sep 26, 2024

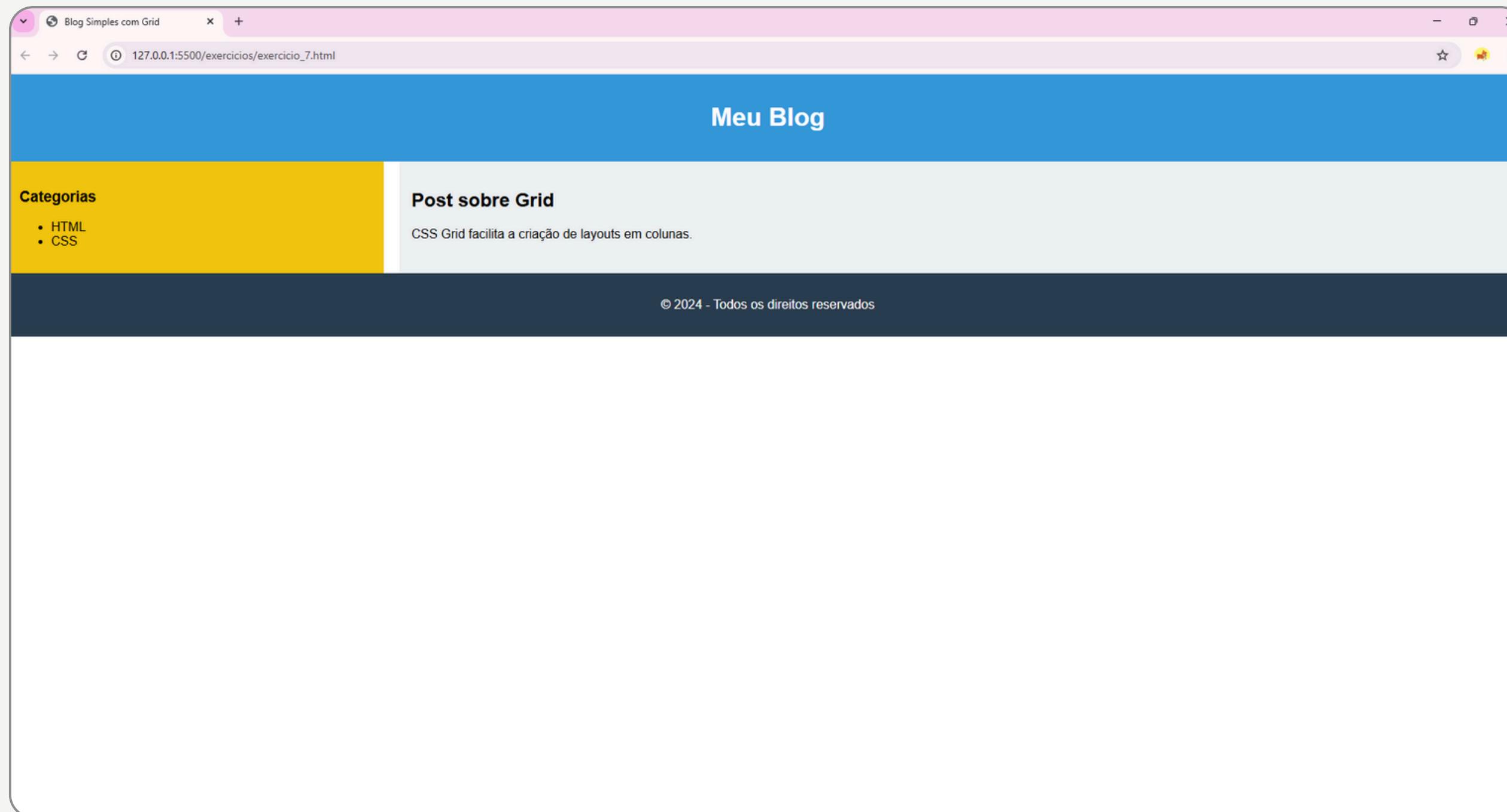


A simple visual cheatsheet for CSS Grid Layout

Learn all about the properties available in CSS Grid Layout through simple visual examples.

 GRID

Exercício 7



Flexbox

► O Flexbox é um **modelo de layout unidimensional (linha ou coluna)**.

- É ativado com a propriedade `display: flex` no elemento pai (container).

► **`display: flex;`**

- Os filhos passam a ter um tamanho flexível e ficam um ao lado do outro.

► **`flex-wrap: wrap;`**

- Caso não caiba todos os elementos em uma mesma linha, quebre para a próxima.

Align e Justify

O Align e Justify funcionam da mesma forma que no CSS Grid Layout. Lembre-se que essas propriedades só funcionam se existir espaço entre os elementos

```
body {  
    font-family: Arial, sans-serif;  
    padding: 20px;  
}  
  
h2 {  
    margin-top: 40px;  
    border-bottom: 2px solid #ccc;  
    padding-bottom: 5px;  
}  
  
.container {  
    border: 2px dashed #999;  
    padding: 10px;  
    margin-bottom: 20px;  
}  
  
.item {  
    background-color: #4CAF50;  
    color: white;  
    padding: 20px;  
    margin: 5px;  
    text-align: center;  
}  
  
/* 1. display: flex */  
.flex-exemplo {  
    display: flex;  
}
```

CSS

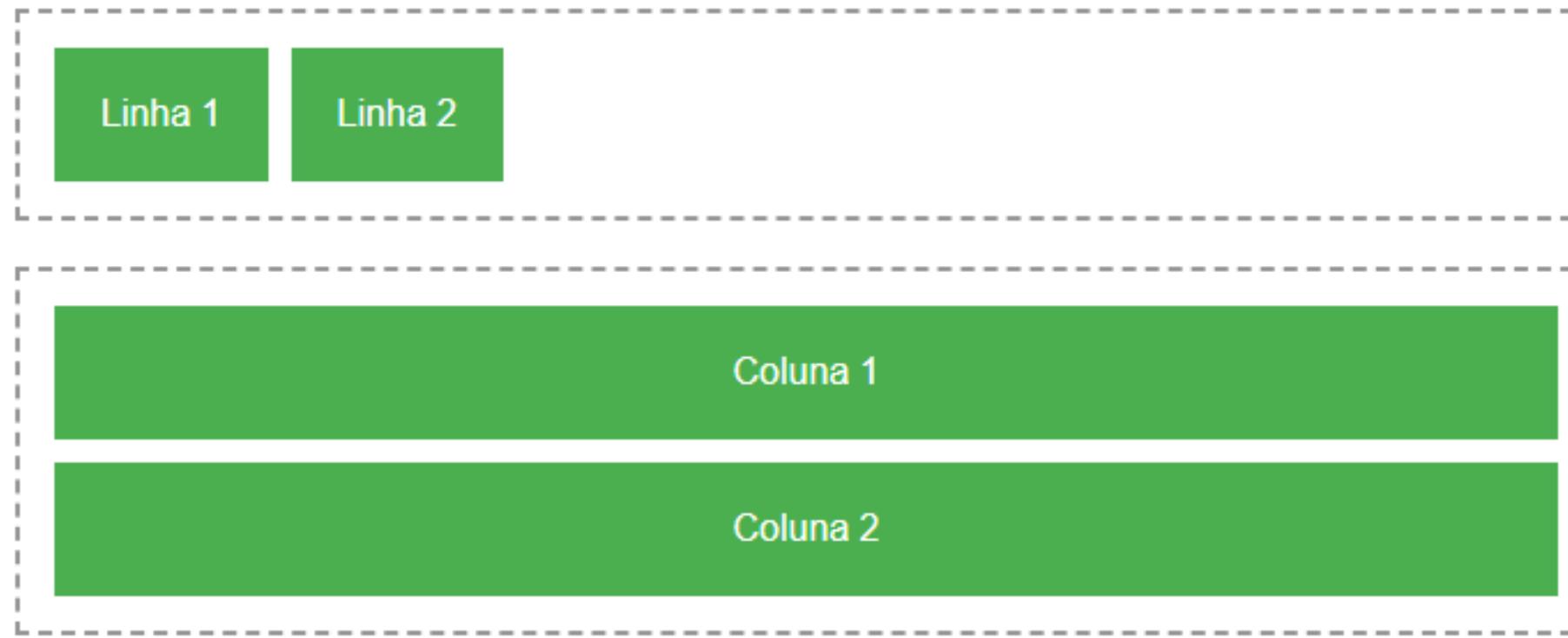
```
<h2>1. display: flex</h2>  
<div class="container flex-exemplo">  
    <div class="item">Item 1</div>  
    <div class="item">Item 2</div>  
    <div class="item">Item 3</div>  
</div>
```

HTML

1. display: flex



2. flex-direction: row / column



```
<h2>2. flex-direction: row / column</h2>
<div class="container flex-exemplo row">
  <div class="item">Linha 1</div>
  <div class="item">Linha 2</div>
</div>
<div class="container flex-exemplo column">
  <div class="item">Coluna 1</div>
  <div class="item">Coluna 2</div>
</div>
```

HTML

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center; /* 2. flex-direction */
}
```

CSS

```
.row { flex-direction: row; }
.column { flex-direction: column; }
```

3. justify-content: space-between



HTML

```
<h2>3. justify-content: space-between</h2>
<div class="container flex-exemplo space-between">
  <div class="item">A</div>
  <div class="item">B</div>
  <div class="item">C</div>
</div>
```

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

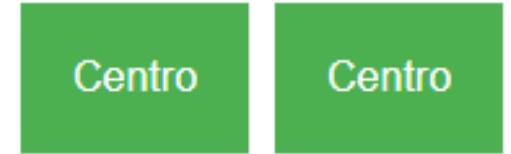
.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}

/* 3. justify-content */
.space-between { justify-content: space-between; }
```

CSS

4. align-items: center



HTML

```
<h2>4. align-items: center</h2>
<div class="container flex-exemplo align-center">
  <div class="item">Centro</div>
  <div class="item">Centro</div>
</div>
```

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

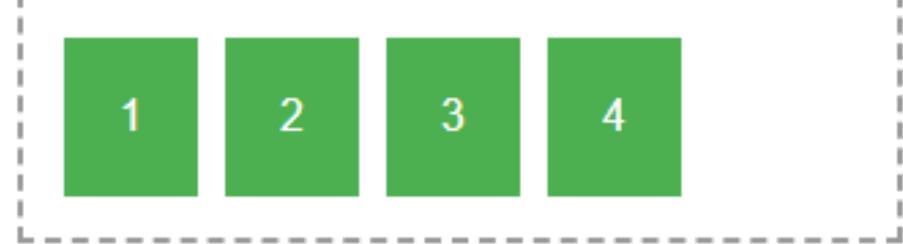
.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}

/* 4. align-items */
.align-center { align-items: center; height: 150px; }
```

CSS

5. flex-wrap: wrap



HTML

```
<h2>5. flex-wrap: wrap</h2>
<div class="container flex-exemplo wrap">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
</div>
```

CSS

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

```
/* 5. flex-wrap */
.wrap {
  flex-wrap: wrap;
  width: 300px;
}
```

6. gap entre os itens



```
<h2>6. gap entre os itens</h2>
```

```
<div class="container flex-exemplo gap">
  <div class="item">X</div>
  <div class="item">Y</div>
  <div class="item">Z</div>
</div>
```

HTML

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

CSS

```
/* 6. gap */
.gap {
  gap: 10px;
}
```

7. flex-grow



HTML

```
<h2>7. flex-grow</h2>
<div class="container flex-exemplo grow">
  <div class="item">1x</div>
  <div class="item">2x</div>
</div>
```

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

CSS

```
/* 7. flex-grow */
.grow .item:nth-child(1) { flex-grow: 1; }
.grow .item:nth-child(2) { flex-grow: 2; }
```

8. align-self (individual)



```
<h2>8. align-self (individual)</h2>
<div class="container flex-exemplo align-center self-align">
  <div class="item">Topo</div>
  <div class="item">Fundo</div>
  <div class="item">Topo</div>
</div>
```

HTML

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

h2 {
  margin-top: 40px;
  border-bottom: 2px solid #ccc;
  padding-bottom: 5px;
}

.container {
  border: 2px dashed #999;
  padding: 10px;
  margin-bottom: 20px;
}

.item {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

CSS

```
/* 8. align-self */
.self-align .item:nth-child(2) {
  align-self: flex-end;
}
```

Flexbox



CSS Flexbox Layout Guide

Our comprehensive guide to CSS flexbox layout. This complete guide explains everything about flexbox, focusing on all the different possible properties for the parent element (the flex container) and the child...

* CSS-Tricks / Aug 12, 2024

EDIT ON CODEPEN

HTML CSS JS Result

FLEX-WRAP (property of the flex container)

nowrap wrap wrap-reverse

1 2 3 4 5

FLEX-BOX (property of flex items)

Run Pen

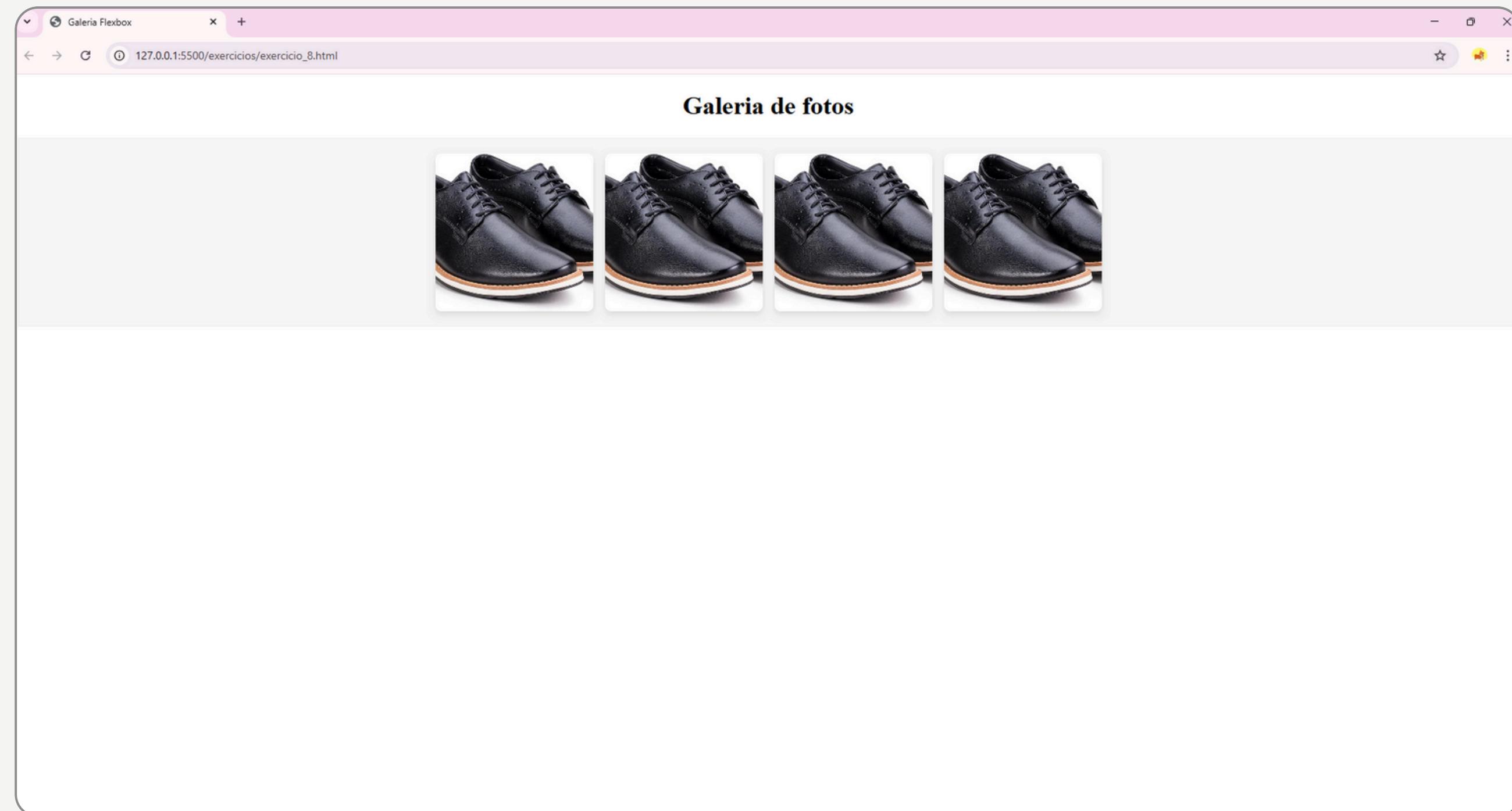
Resources 1x 0.5x 0.25x Rerun

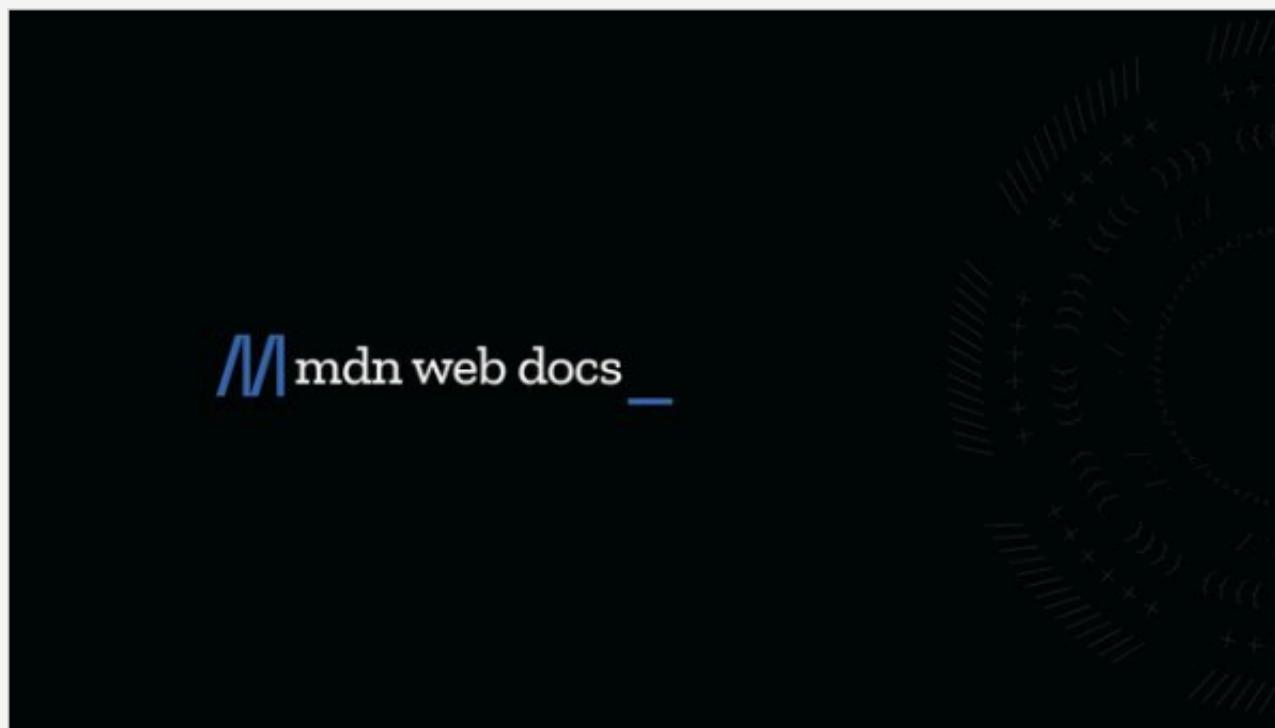
<https://the-echoplex.net/flexyboxes/>

Comparação: Flexbox vs CSS Grid

Característica	Flexbox	CSS Grid
Modelo	Unidimensional	Bidimensional
Direção do Layout	Linha ou Coluna	Linhas e Colunas
Posicionamento	Ordem no HTML	Posicionamento livre na grade
Controle de Espaçamento	justify-content, gap, margin	grid-gap, row-gap, column-gap
Responsividade	Fluído e simples	Preciso para layouts complexos
Áreas nomeadas	Não	Sim, com grid-template-areas

Exercício 8

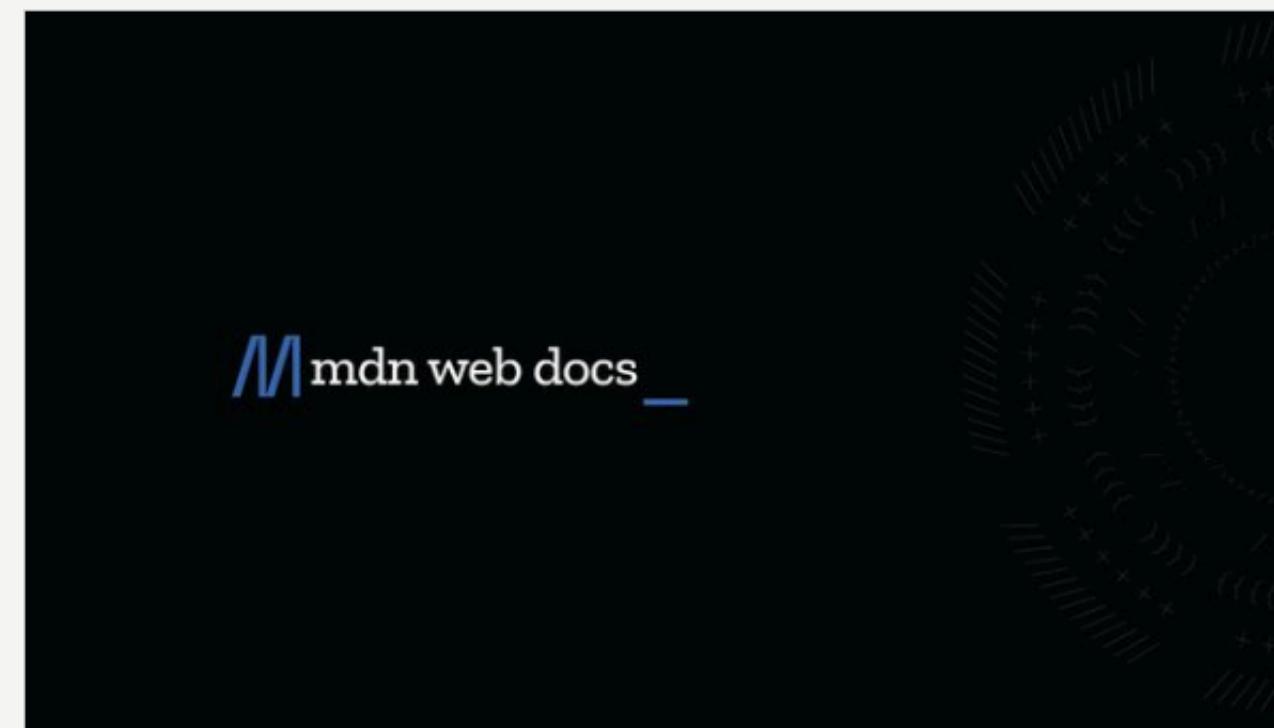




CSS Layout cookbook - CSS | MDN

O livro de receitas (Cookbook) de layout CSS visa reunir receitas para padrões de layout comuns, coisas que você pode precisar implementar em seus próprios sites.

 MDN Web Docs



Referência de CSS - CSS | MDN

Use esta referência de CSS para navegar por um índice alfabético das propriedades padrão do CSS, pseudo-classes, pseudo-elementos, tipos de dados e @-rules.

 MDN Web Docs

<https://jigsaw.w3.org/css-validator/>