🎯 **1. LAYOUT E BOX MODEL**

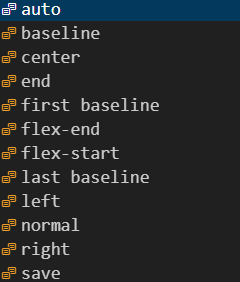
Responsável por como os elementos são organizados e espaçados.

* 📌 LAYOUT/BOX MODEL – DEFINE TAMANHO E ESPAÇAMENTO
* width: Define a largura do elemento.
* height: Define a altura do elemento.
* margin: Define a margem externa do elemento/ Remove margens padrão
* padding: Define o preenchimento interno do elemento.
* border: Define a borda do elemento (ex: 1px solid black).
* box-sizing: Define **como o tamanho da caixa é calculado** (incluindo ou não padding e border).
* **text-overflow:** Define como o texto é exibido quando ultrapassa o container (ex: reticências).
* **text-wrap:** Controla se o texto deve quebrar para a próxima linha. Layout.
* **:root** define **variáveis** que controlam **tamanho, margem, padding, largura, altura, etc.** que fazem parte da estrutura da página.
* 📌 POSICIONAMENTO – CONTROLA ONDE O ELEMENTO APARECE
* position: Define o tipo de posicionamento (ex: absolute, relative, fixed).
* top, right, bottom, left: Define a posição do elemento com base no tipo de posicionamento.
* z-index: Define a sobreposição entre elementos posicionados.
* float: Faz o elemento flutuar à esquerda ou direita.
* **@media usada para aplicar estilos condicionalmente, dependendo das características do dispositivo (largura, altura, tipo de tela, etc.). Ou seja: criar layouts responsivos, ou seja, que se adaptam a diferentes tamanhos de tela**
* 📌 FLEXBOX/GRID – ORGANIZA LAYOUTS DE FORMA DINÂMICA

display: Define como o elemento será exibido na página (comportamento no layout).

* none → Some do layout (invisível e sem espaço)
* inline → Fica na mesma linha (ex: <span>)
* block → Ocupa toda a linha (ex: <div>, <p>)
* inline-block → Igual inline, mas aceita width e height
* flex → Ativa Flexbox no elemento pai
* inline-flex → Flexbox, mas em linha
* grid → Ativa Grid Layout
* inline-grid → Grid, mas em linha
* flow-root → Cria um novo contexto de formatação
* contents → Remove a “caixa” do elemento, mas mantém os filhos
* table, table-row, table-cell → Comportamento de tabela
* justify: Usado pra **alinhar itens, conteúdo ou texto** dentro de containers ou áreas específicas.



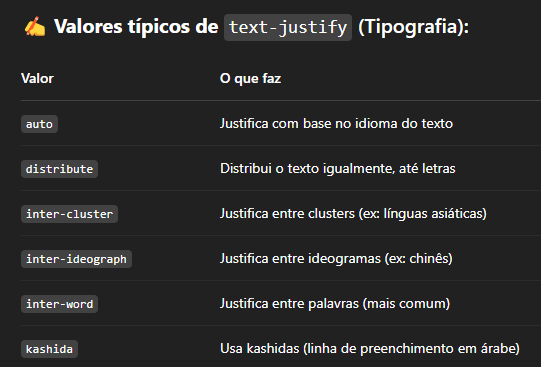
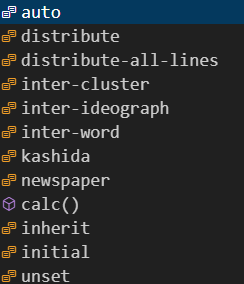
**Observação:** As opções da tabela abaixo aparecem para todos os tipos de transição listados acima

* auto - Valor padrão automático (depende do contexto/layout)
* baseline - Alinha os itens pela linha de base do texto
* center - Centraliza o item ou grupo
* end - Alinha no final do eixo principal (igual a flex-end)
* first - baseline Alinha todos os itens pela primeira linha de base
* flex-end - Alinha no final (mesma ideia do end)
* flex-start - Alinha no início (mesma ideia do start)
* last baseline - Alinha os itens pela última linha de base
* left - Alinha à esquerda (em layouts bidirecionais)
* normal - Valor padrão, sem alinhamento especial
* right - Alinha à direita
* save - Pouco comum, usado em contextos de estilos salvos (raro)
* 📌 ROLAGEM / SCROLL – Comportamento de rolagem
* scroll-margin-left → Espaço à esquerda quando o elemento é focado por rolagem.
* scroll-padding-left → Espaço interno à esquerda durante rolagem com foco.
* scroll-margin-block-end → Margem no final do bloco ao rolar.
* scroll-margin-inline-end → Margem no fim da linha ao rolar.
* scroll-padding-block-end → Padding (interno) no fim do bloco durante rolagem.
* scroll-padding-inline-end → Padding (interno) no fim da linha durante rolagem.
* scrollbar-base-color → Cor base da barra de rolagem (obsoleta).
* scrollbar-face-color → Cor da área que você arrasta (obsoleta).
* scrollbar-gutter → Reserva espaço pra barra de rolagem.
* scroll-snap-type → Define se o scroll "trava" em certos pontos.
* 📌 RESPONSIVIDADE
* **@media** – Aplica estilos conforme largura, altura, tipo de tela, etc.
* **@container** – Aplica estilos conforme o tamanho do contêiner pai.
* **@supports** – Aplica estilos apenas se o navegador suportar certos recursos (indiretamente responsivo).
* **@import com media queries** – Importa CSS condicionalmente baseado em largura ou tipo de mídia.
* **Unidades relativas (%, em, rem, vw, vh)** – Escalam com base no viewport ou elemento pai.
* **Flexbox** – Layout flexível que se adapta à tela.
* **Grid Layout** – Sistema de layout responsivo baseado em linhas e colunas.
* **Clamp(), min(), max()** – Definem valores fluidos e adaptáveis (ex: tamanhos de fonte).

🎯 **2. TIPOGRAFIA**

* TIPOGRAFIA
* font-style: Define o estilo da fonte (normal, itálico, etc.)
* font-weight: Define a espessura da fonte (ex: fino, normal, negrito, etc).
* font-size: Define o tamanho da fonte.
* font-family: Define o tipo de fonte.
* font-display: (Avançado) Define como a fonte é exibida enquanto carrega
* font-variant: Controla variações como versaletes (pequenas maiúsculas)
* font-feature-settings: (Avançado) Ativa recursos específicos da fonte (ex: ligaduras, frações, etc.)
* font-stretch: Alarga ou estreita a fonte (quando suportado)
* font-variation-settings: Ajusta eixos variáveis em fontes variáveis
* font-variant-ligatures: Controla ligaduras (junções entre letras como “fi”)
* font-variant-numeric: Controla o estilo de números (ex: tabelados, proporcionais)
* line-height: Define a altura da linha de texto.
* color: Define a cor do texto.
* text-align: Controla o alinhamento do texto (esquerda, centro, direita, justificado, funciona apenas para centralizar conteúdo *dentro* de um elemento pai em bloco).
* text-transform: controla a forma do texto (maiúsculo, minúsculo, capitalizado).
* text-justify: controla **como o texto justificado é espaçado** em uma linha.
* **text-size-adjust:** Ajusta o tamanho do texto em dispositivos móveis.
* Text-decoration: Define **o tipo de linha** que será usada no texto. (underline (linha embaixo), overline (linha em cima), line-through (linha no meio), none (sem linha).
* text-decoration-color: Define **a cor da linha** de decoração.
* text-decoration-style: Define **o estilo visual da linha**. solid (linha contínua), dotted (pontilhada), dashed (tracejada), wavy (ondulada), double (duas linhas)
* **text-decoration-thickness:** Define a espessura da linha de decoração.
* **text-overflow:** Define como o texto é exibido quando ultrapassa o container (ex: reticências).
* **text-shadow:** Adiciona sombra ao texto.
* **text-indent:** Faz a **primeira linha do parágrafo começar mais pra dentro**, como nos livros. É só um **espaço extra no começo da linha**.
* **text-rendering:** Controla a otimização de renderização do texto.
* **text-wrap:** Controla se o texto deve quebrar para a próxima linha. Layout.
* erve para **controlar o espaçamento entre as letras** de um texto.
* **letter-spacing:** **Controla o espaçamento entre as letras** de um texto.
* **initial-letter:** Define o tamanho e a elevação da letra inicial em relação ao restante do texto. Ex: (initial-letter: 2 1). Primeiro valor = altura da letra/ Segundo valor (1) = define alinhamento vertical
* **initial-letter-align:** Controla como a letra capitular é alinhada horizontalmente em relação à linha de base do texto.

obs:   Propriedade shorthand font: Condensar as configurações: Exemplo: font: + os valores de font-style, font-weight, font-size, font-family



🎯 **3. CORES E FUNDOS**

* CORES

background-color: Define a cor de fundo do elemento.

* background-color: red; → Fundo vermelho.
* background-color: #000000; → Fundo preto (hexadecimal).
* background-color: rgb(255, 255, 255); → Fundo branco (RGB).
* FUNDOS

background-image → Define a imagem de fundo de um elemento.

* background-image: url: Usa uma imagem externa.
* background-image: linear-gradient: serve pra criar um **degradê** como fundo.
* background-image: linear-gradient – Degradê linear
* background-image: radial-gradient – Degradê em círculo/elipse
* background-image: conic-gradient – Degradê em formato de cone
* background-image: repeating-linear-gradient – Degradê linear repetido
* background-image: repeating-radial-gradient – Degradê radial repetido
* background-image: repeating-conic-gradient – Degradê cônico repetido

background-size → Controla o tamanho da imagem de fundo.

* cover → Faz a imagem cobrir todo o elemento, mantendo a proporção.
* contain → Ajusta a imagem para caber inteira dentro do elemento.
* Valores específicos como 100px 200px definem largura e altura manualmente.

background-position → Define a posição da imagem dentro do elemento.

* center → Centraliza a imagem.
* top, bottom, left, right → Alinha conforme indicado.
* Valores como 50% 50% posicionam a imagem em coordenadas percentuais.

background-repeat → Controla a repetição da imagem.

* no-repeat → Não repete a imagem.
* repeat → Repete tanto na horizontal quanto na vertical.
* repeat-x ou repeat-y → Repete apenas em um eixo.

background-attachment → Define se a imagem de fundo se move com a rolagem.

* fixed → A imagem fica fixa enquanto o conteúdo rola.
* scroll → A imagem rola junto com o conteúdo.
* local → A imagem rola apenas dentro do elemento.

box-decoration-break → controla como bordas, fundos e decoração se comportam em elementos quebrados

* slice (padrão) → aplica borda e fundo só na caixa inteira
* clone → replica borda e padding em cada linha quebrada
* EFEITOS VISUAIS

Filter: Aplica efeitos visuais como blur, brilho, contraste, etc.

* Blur → desfoca
* brightness → brilho
* contrast → contraste
* drop-shadow → sombra (tipo box-shadow)
* grayscale → preto e branco
* hue-rotate → gira as cores
* invert → inverte as cores
* opacity → transparência
* saturate → intensidade das cores
* sepia → efeito envelhecido
* url → usa SVG como filtro
* none → remove filtros
* box-shadow: Aplica sombra ao redor do elemento.
* text-shadow: Aplica sombra ao redor do texto.
  + - Exemplo: box-shadow: 5px 10px 15px rgba(128, 105, 64, 0.706)
    - 5px deslocamento **horizontal** (pra direita)
    - 10px deslocamento **vertical** (pra baixo)
    - 15px **desfoque** (blur)
* opacity: altera a transparência visual do elemento, sem mudar o layout.
* Outline: Cria um contorno ao redor de um elemento sem afetar o layout (não ocupa espaço).

obs:  Propriedade shorthand outline: width + style + color

🎯 **4. BORDAS E DECORAÇÃO**

* BORDAS E CANTOS
* border – Atalho pra criar borda completa
* border-radius – Arredonda todos os cantos
* border-color – Cor da borda
* border-width – Espessura da borda
* border-top – Borda em cima
* border-right – Borda à direita
* border-bottom – Borda embaixo
* border-left – Borda à esquerda
* border-style – Estilo da borda (solid, dashed...)
* border-top-left-radius – Arredonda canto superior esquerdo
* border-top-right-radius – Arredonda canto superior direito
* border-bottom-right-radius – Arredonda canto inferior direito

obs:  Propriedade shorthand border: width + style + color  
obs:  Propriedade shorthand tamanho: top + right + bottom + left

* TRANSFORMAÇÕES

Move, gira ou redimensiona elementos (div, img, button, span, p, h1 até section, article, header, footer, main etc)

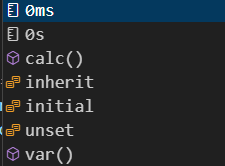
transform: Propriedade principal para aplicar transformações. O **correto é sempre usar transform:** seguido da(s) **função(ões) de transformação**, como scale, rotate, translate, etc

* matrix → Aplica múltiplas transformações 2D de uma vez com valores matemáticos (avançado)
* matrix3d → Versão 3D do matrix(), mais complexa
* none → Remove qualquer transformação aplicada
* perspective → Define profundidade para efeitos 3D
* rotate → Gira o elemento no plano 2D
* rotate3d → Gira em 3D, usando eixo X, Y e Z
* rotateX → Gira em torno do eixo X (efeito "cabecear")
* rotateY → Gira em torno do eixo Y (efeito "virar de lado")
* rotateZ → Gira em torno do eixo Z (mesmo que rotate(), plano 2D)
* scale → Aumenta ou diminui o tamanho geral
* scale3d → Escala diferente nos 3 eixos (X, Y, Z)
* scaleX→ Escala só na horizontal
* scaleY → Escala só na vertical
* scaleY Escala só na vertical
* translate → Move o elemento no eixo X e/ou Y (ex: translate(10px, 20px))
* scale: Aumenta ou diminui o tamanho do elemento.
* TRANSIÇÕES E ANIMAÇÕES

Transition: Permite que mudanças de CSS aconteçam **de forma suave**, ao invés de instantaneamente.

* transition-property: Qual propriedade CSS será animada
* transition-duration: Quanto tempo dura a transição
* transition-delay: Tempo de espera antes de começar
* transition (shorthand): Atalho para juntar todas as propriedades

Observação: As opções da tabela abaixo aparecem para todos os tipos de transição listados acima

* 0ms / 0s – Sem atraso na transição
* calc() – Permite calcular o tempo dinamicamente Ex: calc(2s + 1s)
* inherit – Herda o valor do elemento pai
* initial – Volta ao valor padrão do CSS (que é 0s)
* unset – Remove valor: herda se possível, ou usa padrão (0s)
* var() – Usa uma variável CSS

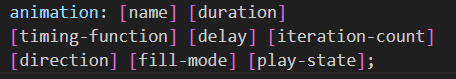
transition-timing-function: Ritmo da transição

* linear – Velocidade constante
* ease – Devagar → rápido → devagar
* ease-in – Começa devagar
* ease-out – Termina devagar
* ease-in-out – Começa e termina devagar

Animation: Atalho que combina várias propriedades

* animation-name – Nome da animação criada com @keyframes
* animation-duration – Tempo de duração (ex: 2s)
* animation-timing-function – Define o ritmo (ex: ease, linear)
* animation-delay – Tempo de espera antes de iniciar
* animation-iteration-count – Quantas vezes repete (ex: infinite, 2)
* animation-direction – Sentido da animação (ex: normal, reverse)
* animation-fill-mode – Mantém o estilo no início/fim (ex: forwards)
* animation-play-state – Pausa ou continua (ex: paused, running)
* animation-composition – Como animações se combinam (pouco usada)
* animation-range – Intervalo de tempo da animação (ainda experimental)
* animation-range-end – Define o fim do intervalo (experimental)

**A ordem correta dos valores no shorthand animation**



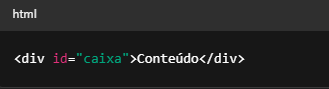
🎯 **5. INTERAÇÃO E ESTADO**

* USANDO O ID COM CSS

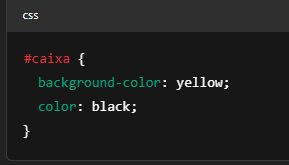
**Id:** É um identificador único que você dá a um elemento HTML para poder estilizar ou manipular ele individualmente com CSS ou JavaScript.

* Um **ID** deve ser único na página — não repita o mesmo ID em vários elementos!
* Serve para destacar ou tratar um único item.
* Ideal para coisas específicas (ex: #menu-principal, #banner, #botao-login).

No HTML você usa o atributo id assim: **id= " "**

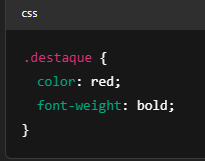
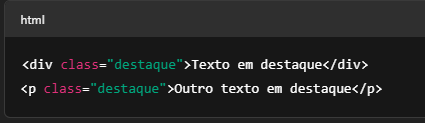


No CSS você usa o atributo id assim: **#caixa {}**



**class:** É uma maneira de agrupar e aplicar estilos a vários elementos ao mesmo tempo. Diferente do **id**, pode ser repetido em quantos elementos você quiser.

* Pode usar a mesma **class** em vários elementos.
* Um elemento pode ter várias **classes** ao mesmo tempo, separadas por espaço.
* Serve para criar estilos reaproveitáveis, tipo botões, caixas, textos, etc.
* No HTML você usa o atributo assim: **class= " "**
* No cCSS você usa o atributo com um ponto “.” + o nome da calss, exemplo: **class= "destaque"** no html, então no CSS vai ficar. destaque

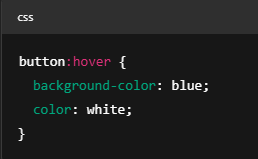


* PSEUDOCLASSES E PSEUDOELEMENTOS

**PSEUDOCLASSES** É um recurso do CSS (não do HTML) que permite aplicar estilos a um elemento quando ele está em um estado específico ou em uma posição específica no HTML.

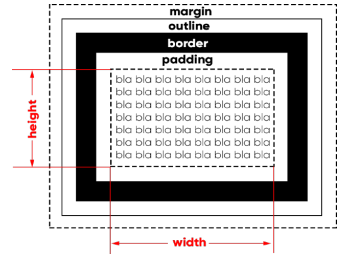
Ela sempre começa com dois pontos **:**

🧪 Exemplos de pseudoclasses:

* **:hover** Quando o mouse passa por cima
* **:active** Enquanto o elemento está sendo clicado
* **:focus** Quando o campo (input, textarea) está em foco
* **:checked** Quando checkbox ou radio está marcado
* **:first-child** Seleciona o primeiro filho de um elemento
* **:last-child** Seleciona o último filho
* **:nth-child**(n) Seleciona o filho número n (ex: :nth-child(2))
* **:not**(seletor) Seleciona o que não corresponde ao seletor especificado
* **:disabled** Seleciona um input desativado
* **:enabled** Seleciona um input ativado
* **:visited** Links que já foram clicados
* **:link** Links que ainda não foram visitados
* 🧱 Exemplo prático ABAIXO: Isso muda o botão quando o mouse passa por cima.

🎯 **6. ESTRUTURA E ORGANIZAÇÃO**

* ANATOMIA DAS CAIXAS



Box-Level (Bloco):

* Ocupa toda a largura disponível
* Começa em nova linha
* Permite width, height, margin, padding

**Exemplos**

* **<address>** – Define informações de contato (autor ou dono do documento).
* **<blockquote>** – Exibe uma citação longa, geralmente com recuo.
* **<canvas>** – Área de desenho para gráficos, jogos ou imagens via JavaScript.
* **<dd>** – Define a descrição de um termo em uma lista (<dl>).
* **<div>** – Contêiner genérico para agrupar elementos (usado com CSS/JS).
* **<dl>** – Lista de definição (termo e descrição).
* **<dt>** – Define um termo em uma lista de definição.
* **<fieldset>** – Agrupa campos relacionados dentro de um formulário.
* **<figcaption>** – Legenda de uma figura (<figure>).
* **<figure>** – Contêiner de mídia (imagem, gráfico, etc.) com legenda.
* **<form>** – Contêiner para elementos de formulário.
* **<h1> a <h6>** – Títulos, indo do mais importante (<h1>) ao menos importante (<h6>).
* **<hr>** – Linha horizontal que separa conteúdos tematicamente.
* **<li>** – Item de uma lista (<ul>, <ol>).
* **<noscript>** – Mostra conteúdo alternativo se o JavaScript estiver desativado.
* **<ol>** – Lista ordenada (com números).
* **<p>** – Parágrafo de texto.
* **<pre>** – Texto pré-formatado (mantém espaços e quebras de linha).
* **<table>** – Tabela de dados.
* **<tfoot>** – Rodapé de uma tabela (geralmente para totais).
* **<ul>** – Lista não ordenada (com marcadores).
* **<video>** – Incorpora um vídeo na página.

Inline-Level (Em linha):

* Ocupam apenas o espaço necessário ao seu conteúdo.
* Não quebram a linha, ficam "na mesma linha" que outros elementos inline.
* Não respeitam largura e altura (essas propriedades não funcionam neles por padrão).
* **<a>** – Cria um hiperlink para outra página, seção ou site.
* **<abbr>** – Indica uma abreviação; pode exibir o significado com title.
* **<acronym>** – (Obsoleta) Usada para acrônimos, hoje se usa <abbr>.
* **<b>** – Deixa o texto em **negrito**, sem dar ênfase semântica.
* **<bdo>** – Controla a direção do texto (ex: da direita para esquerda).
* **<br>** – Quebra de linha (pula para a próxima linha).
* **<button>** – Cria um botão interativo.
* **<cite>** – Refere-se a uma fonte, citação de obra, livro, site etc.
* **<code>** – Mostra código em estilo monoespaçado.
* **<dfn>** – Marca a definição de um termo.
* **<em>** – Dá ênfase ao texto (geralmente aparece *itálico*).
* **<i>** – Texto em itálico, sem ênfase semântica.
* **<img>** – Exibe uma imagem.
* **<input>** – Campo de entrada em formulário (texto, botão, etc.).
* **<kbd>** – Representa entrada do teclado (estiliza como tecla).
* **<label>** – Rótulo de um campo de formulário, ligado por for="id".
* **<map>** – Define áreas clicáveis dentro de uma imagem.
* **<object>** – Incorpora conteúdo externo (PDF, vídeo, plugin...).
* **<output>** – Mostra resultados calculados por scripts ou formulários.
* **<q>** – Define citação curta (usa aspas automáticas).
* **<samp>** – Representa saída de um programa ou sistema.
* **<script>** – Adiciona ou executa código JavaScript.
* **<select>** – Cria uma lista suspensa (dropdown).
* **<small>** – Mostra texto menor (ex: observações).
* **<span>** – Contêiner genérico inline (usado com CSS/JS).
* **<strong>** – Dá ênfase forte ao texto (normalmente negrito com valor semântico).
* **<sub>** – Texto subscrito (abaixo da linha, tipo fórmulas).
* **<textarea>** – Campo de texto com várias linhas.
* **<var>** – Representa uma variável em contextos como matemática ou código.