

# Guia Completo sobre Cascading Style Sheets

Um Guia Detalhado e Simples para CSS3

Viktor Krauss



Data de Publicação: September 3, 2024

# Contents

<b>1</b>	<b>Capítulo 1 - Introdução</b>	<b>6</b>
1.1	O que é CSS?	6
1.2	Importância do CSS	6
1.3	Evolução para CSS3	7
1.4	Objetivo do Guia	7
1.5	HTML5 e CSS3 - O casal perfeito	8
1.5.1	HTML5	8
1.5.2	CSS3 e HTML5	8
1.5.3	Integração do HTML5 e CSS3	8
<b>2</b>	<b>Capítulo 2 - Seletores</b>	<b>9</b>
2.1	Seletores de Tipo	9
2.2	Seletores de Classe	9
2.3	Seletores de ID	10
2.4	Seletores de Atributo	10
2.5	Seletores Pseudo-Classe	10
2.6	Seletores Pseudo-Elementos	10
2.7	Seletores Combinadores	11
<b>3</b>	<b>Capítulo 3 - Propriedades de Layout</b>	<b>12</b>
3.1	Box Model	12
3.2	Display	13
3.3	Position	13
3.4	Float	14
3.5	Clear	15
<b>4</b>	<b>Capítulo 4 - Flexbox</b>	<b>16</b>
4.1	Conceitos Básicos	16
4.1.1	Contêiner Flexível	16
4.1.2	Itens Flexíveis	16
4.2	Propriedades do Contêiner Flexível	17

4.2.1	Propriedade ‘flex-direction’	17
4.2.2	Propriedade ‘flex-wrap’	17
4.2.3	Propriedade ‘flex-flow’	18
4.2.4	Propriedade ‘justify-content’	18
4.2.5	Propriedade ‘align-items’	18
4.2.6	Propriedade ‘align-content’	19
4.3	Propriedades dos Itens Flexíveis	19
4.3.1	Propriedade ‘flex-grow’	20
4.3.2	Propriedade ‘flex-shrink’	20
4.3.3	Propriedade ‘flex-basis’	20
4.3.4	Propriedade ‘flex’	20
4.3.5	Propriedade ‘align-self’	20
<b>5</b>	<b>Capítulo 5 - Grid Layout</b>	<b>21</b>
5.1	Conceitos Básicos	21
5.1.1	Contêiner Grid	21
5.1.2	Itens Grid	21
5.2	Propriedades do Contêiner Grid	22
5.2.1	Propriedade ‘grid-template-columns’ e ‘grid-template-rows’	22
5.2.2	Propriedade ‘grid-template-areas’	22
5.2.3	Propriedade ‘grid-column’ e ‘grid-row’	22
5.2.4	Propriedade ‘grid-gap’	23
5.3	Propriedades dos Itens Grid	23
5.3.1	Propriedade ‘justify-self’	23
5.3.2	Propriedade ‘align-self’	24
5.3.3	Propriedade ‘grid-auto-columns’ e ‘grid-auto-rows’	24
<b>6</b>	<b>Capítulo 6 - Estilização de Texto</b>	<b>25</b>
6.1	Propriedades de Fonte	25
6.1.1	Propriedade ‘font-family’	25
6.1.2	Propriedade ‘font-size’	25
6.1.3	Propriedade ‘font-weight’	26
6.1.4	Propriedade ‘font-style’	26
6.1.5	Propriedade ‘font-variant’	26
6.2	Propriedades de Cor	26
6.2.1	Propriedade ‘color’	26
6.2.2	Propriedade ‘background-color’	27
6.3	Propriedades de Alinhamento	27
6.3.1	Propriedade ‘text-align’	27
6.3.2	Propriedade ‘text-indent’	27
6.3.3	Propriedade ‘line-height’	27

6.4	Propriedades de Espaçamento . . . . .	27
6.4.1	Propriedade ‘letter-spacing’ . . . . .	28
6.4.2	Propriedade ‘word-spacing’ . . . . .	28
6.4.3	Propriedade ‘text-transform’ . . . . .	28
<b>7</b>	<b>Capítulo 7 - Cores e Imagens</b>	<b>29</b>
7.1	Cores . . . . .	29
7.1.1	Nomes de Cores . . . . .	29
7.1.2	Valores Hexadecimais . . . . .	29
7.1.3	Valores RGB . . . . .	30
7.1.4	Valores RGBA . . . . .	30
7.1.5	Valores HSL . . . . .	30
7.1.6	Valores HSLA . . . . .	30
7.2	Imagens . . . . .	30
7.2.1	Propriedade ‘background-image’ . . . . .	31
7.2.2	Propriedade ‘background-repeat’ . . . . .	31
7.2.3	Propriedade ‘background-size’ . . . . .	31
7.2.4	Propriedade ‘background-position’ . . . . .	31
7.2.5	Propriedade ‘opacity’ . . . . .	31
7.3	Imagens Responsivas . . . . .	32
<b>8</b>	<b>Capítulo 8 - Bordas e Sombras</b>	<b>33</b>
8.1	Bordas . . . . .	33
8.1.1	Propriedade ‘border’ . . . . .	33
8.1.2	Propriedade ‘border-width’ . . . . .	33
8.1.3	Propriedade ‘border-style’ . . . . .	34
8.1.4	Propriedade ‘border-color’ . . . . .	34
8.1.5	Propriedade ‘border-radius’ . . . . .	34
8.2	Sombras . . . . .	34
8.2.1	Propriedade ‘box-shadow’ . . . . .	34
8.2.2	Propriedade ‘text-shadow’ . . . . .	35
8.3	Efeitos Combinados . . . . .	35
8.3.1	Exemplo Combinado . . . . .	35
<b>9</b>	<b>Capítulo 9 - Transições e Animações</b>	<b>36</b>
9.1	Transições . . . . .	36
9.1.1	Propriedade ‘transition’ . . . . .	36
9.1.2	Propriedade ‘transition-property’ . . . . .	36
9.1.3	Propriedade ‘transition-duration’ . . . . .	37
9.1.4	Propriedade ‘transition-timing-function’ . . . . .	37
9.1.5	Propriedade ‘transition-delay’ . . . . .	37

9.2	Animações . . . . .	37
9.2.1	Propriedade ‘animation’ . . . . .	37
9.2.2	Propriedade ‘@keyframes’ . . . . .	38
9.2.3	Propriedade ‘animation-name’ . . . . .	38
9.2.4	Propriedade ‘animation-duration’ . . . . .	38
9.2.5	Propriedade ‘animation-timing-function’ . . . . .	38
9.2.6	Propriedade ‘animation-delay’ . . . . .	39
9.2.7	Propriedade ‘animation-iteration-count’ . . . . .	39
9.2.8	Propriedade ‘animation-direction’ . . . . .	39
<b>10</b>	<b>Capítulo 10 - Responsividade e Media Queries</b>	<b>40</b>
10.1	Introdução à Responsividade . . . . .	40
10.2	Media Queries . . . . .	40
10.2.1	Sintaxe Básica . . . . .	40
10.2.2	Media Queries para Diferentes Dispositivos . . . . .	41
10.3	Propriedades Avançadas de Media Queries . . . . .	41
10.3.1	Orientação . . . . .	42
10.3.2	Resolução . . . . .	42
10.4	Exemplo Completo . . . . .	42
<b>11</b>	<b>Capítulo 11 - Transformações</b>	<b>44</b>
11.1	Transformações Básicas . . . . .	44
11.1.1	Propriedade ‘transform’ . . . . .	44
11.1.2	Transformação ‘translate’ . . . . .	44
11.1.3	Transformação ‘scale’ . . . . .	45
11.1.4	Transformação ‘rotate’ . . . . .	45
11.1.5	Transformação ‘skew’ . . . . .	45
11.1.6	Transformações Múltiplas . . . . .	45
11.2	Transformações 3D . . . . .	45
11.2.1	Propriedade ‘perspective’ . . . . .	46
11.2.2	Transformação ‘rotateX’ e ‘rotateY’ . . . . .	46
11.2.3	Transformação ‘translateZ’ . . . . .	46
11.2.4	Transformações 3D Combinadas . . . . .	46
<b>12</b>	<b>Capítulo 12 - Pseudo-Classes e Pseudo-Elementos Avançados</b>	<b>47</b>
12.1	Pseudo-Classes . . . . .	47
12.1.1	Pseudo-Classe ‘:nth-child’ . . . . .	47
12.1.2	Pseudo-Classe ‘:not’ . . . . .	48
12.1.3	Pseudo-Classe ‘:nth-of-type’ . . . . .	48
12.1.4	Pseudo-Classe ‘:last-child’ . . . . .	48
12.2	Pseudo-Elementos . . . . .	48

12.2.1	Pseudo-Elemento ‘::before’ . . . . .	48
12.2.2	Pseudo-Elemento ‘::after’ . . . . .	49
12.2.3	Pseudo-Elemento ‘::first-line’ . . . . .	49
12.2.4	Pseudo-Elemento ‘::first-letter’ . . . . .	49
12.3	Exemplos Combinados . . . . .	50

# Chapter 1

## Capítulo 1 - Introdução

### 1.1 O que é CSS?

CSS, sigla para Cascading Style Sheets (Folhas de Estilo em Cascata), é uma linguagem de estilo utilizada para definir a apresentação de documentos HTML e XML (incluindo variantes como SVG e XHTML). Criado por Håkon Wium Lie e introduzido em 1996 pelo W3C (World Wide Web Consortium), o CSS permite a separação da estrutura do conteúdo da sua apresentação visual.

A principal função do CSS é controlar a aparência dos elementos de uma página web, incluindo a disposição, cores, fontes e espaçamento. Com o CSS, é possível criar layouts complexos e responsivos que se adaptam a diferentes tamanhos de tela e dispositivos.

### 1.2 Importância do CSS

A utilização do CSS oferece várias vantagens significativas:

- **Separação de Conteúdo e Apresentação:** Permite que o conteúdo HTML seja mantido separado das regras de apresentação, facilitando a manutenção e atualização do site.
- **Reutilização de Estilos:** Com o CSS, você pode aplicar o mesmo estilo a vários elementos, evitando a duplicação de código.
- **Melhor Desempenho:** A separação de estilos e conteúdo pode melhorar o tempo de carregamento das páginas, pois o CSS pode ser armazenado em um arquivo separado e carregado uma vez.

- **Controle Avançado de Layout:** Oferece ferramentas para criar layouts avançados, responsivos e adaptáveis a diferentes dispositivos e tamanhos de tela.

## 1.3 Evolução para CSS3

CSS3 é a terceira versão da linguagem CSS e traz uma série de melhorias e novas funcionalidades em comparação com seu antecessor, CSS2. Introduzido oficialmente em 1999, o CSS2 foi a base para o CSS3, que começou a ser desenvolvido no início dos anos 2000.

As principais inovações do CSS3 incluem:

- **Seletores Avançados:** Novos seletores e pseudo-classes permitem uma seleção mais precisa e poderosa dos elementos HTML.
- **Layout e Design:** Introdução de novos módulos como Flexbox e Grid Layout, que facilitam a criação de layouts complexos e responsivos.
- **Efeitos Visuais:** Novas propriedades para adicionar efeitos visuais como sombras, bordas arredondadas, e gradientes.
- **Transições e Animações:** Permite a criação de animações e transições suaves entre estados de elementos.
- **Web Fonts:** Suporte para fontes personalizadas através da regra '@font-face'.

CSS3 é dividido em vários módulos, cada um com um conjunto específico de funcionalidades e propriedades. Essa modularidade permite que os navegadores implementem partes do CSS3 de maneira independente, promovendo a evolução contínua da linguagem.

A compreensão das funcionalidades básicas e avançadas do CSS e CSS3 é essencial para o desenvolvimento web moderno, pois permite criar interfaces ricas e dinâmicas que oferecem uma experiência de usuário aprimorada.

## 1.4 Objetivo do Guia

Este guia visa proporcionar um entendimento detalhado e acessível sobre o CSS e CSS3. Cada capítulo abordará um aspecto específico da linguagem, fornecendo exemplos práticos e explicações claras para ajudar no aprendizado e na aplicação dos conceitos em projetos reais.



## 1.5 HTML5 e CSS3 - O casal perfeito

### 1.5.1 HTML5

O advento do HTML5 representou um marco no desenvolvimento de aplicações web, ampliando as possibilidades de criação de conteúdos interativos e acessíveis. Conforme aponta Pilgrim [pilgrim2010html5], o HTML5 introduziu uma série de novas funcionalidades que simplificaram a incorporação de mídia, como áudio e vídeo, diretamente nas páginas web, eliminando a necessidade de plugins externos, como o Flash, que anteriormente desempenhava esse papel. Além disso, o HTML5 trouxe tags semânticas, como `<header>`, `<footer>`, `<article>` e `<section>`, que contribuem para uma estruturação mais lógica e coesa dos documentos HTML. Essa semântica aprimorada não só facilita o desenvolvimento, como também melhora a acessibilidade das páginas para motores de busca e tecnologias assistivas, conforme ressaltado por Freeman e Robson [freeman2014head].

### 1.5.2 CSS3 e HTML5

Complementando as inovações do HTML5, o CSS3 proporcionou um avanço significativo nas possibilidades de estilização de páginas web. Segundo Meyer [meyer2011css3], o CSS3 introduziu funcionalidades que permitiram a criação de efeitos visuais complexos, como animações e transições, que antes dependiam de linguagens como JavaScript. Além disso, com a chegada de módulos como Flexbox e Grid Layouts, o design responsivo tornou-se uma prática mais acessível e eficaz, permitindo que os elementos das páginas web se ajustem automaticamente a diferentes tamanhos de tela e dispositivos. Essas ferramentas são essenciais em um cenário onde o acesso à internet por dispositivos móveis ultrapassou o uso tradicional de desktops, exigindo layouts que ofereçam uma experiência consistente e de alta qualidade ao usuário.

### 1.5.3 Integração do HTML5 e CSS3

A integração entre HTML5 e CSS3 constitui, portanto, uma base sólida e eficiente para o desenvolvimento web contemporâneo. Juntas, essas tecnologias permitem a criação de sites que são não apenas esteticamente agradáveis, mas também funcionais, acessíveis e de fácil manutenção. A sinergia entre HTML5 e CSS3 garante que as páginas web ofereçam uma experiência de usuário otimizada em qualquer dispositivo, promovendo uma internet mais inclusiva e interativa. De acordo com Refsnes [refsnes2015html5], essa combinação reflete uma evolução natural das linguagens de marcação e estilização, que continua a moldar o futuro da web, alinhando-se às necessidades tanto dos desenvolvedores quanto dos usuários.

# Chapter 2

## Capítulo 2 - Seletores

Os seletores no CSS (Cascading Style Sheets) são utilizados para aplicar estilos a elementos HTML específicos. Aqui, discutiremos os principais tipos de seletores e suas utilizações.

### 2.1 Seletores de Tipo

Os seletores de tipo aplicam estilos a todos os elementos de um tipo específico. Por exemplo, o seletor ‘p’ aplica estilos a todos os parágrafos.

```
p {  
    color: blue;  
    font-size: 14px;  
}
```

### 2.2 Seletores de Classe

Os seletores de classe aplicam estilos a elementos com uma classe específica. Por exemplo, o seletor ‘.destaque’ aplica estilos a todos os elementos com a classe “destaque”.

```
.destaque {  
    background-color: yellow;  
    font-weight: bold;  
}
```

## 2.3 Seletores de ID

Os seletores de ID aplicam estilos a um elemento com um ID específico. O ID deve ser único dentro da página.

```
#especial {  
    border: 2px solid red;  
}
```

## 2.4 Seletores de Atributo

Os seletores de atributo aplicam estilos a elementos com um atributo específico. Por exemplo, o seletor '[type="text"]' aplica estilos a todos os elementos de entrada com o tipo "text".

```
input[type="text"] {  
    border-radius: 4px;  
    padding: 5px;  
}
```

## 2.5 Seletores Pseudo-Classe

Os seletores pseudo-classe aplicam estilos a elementos em um estado específico. Por exemplo, o seletor ':hover' aplica estilos quando o cursor está sobre o elemento.

```
a:hover {  
    color: red;  
}
```

## 2.6 Seletores Pseudo-Elementos

Os seletores pseudo-elemento aplicam estilos a partes específicas de um elemento. Por exemplo, '::before' adiciona conteúdo antes do conteúdo do elemento.

```
p::before {  
    content: "Nota: ";  
    font-weight: bold;  
}
```

## 2.7 Seletores Combinadores

Os seletores combinadores aplicam estilos com base na relação entre os elementos. Por exemplo, o combinador de descendentes aplica estilos a todos os elementos ‘p’ dentro de um elemento ‘div’.

```
div p {  
    color: green;  
}
```

# Chapter 3

## Capítulo 3 - Propriedades de Layout

O layout de uma página web é fundamental para a criação de uma boa experiência de usuário. Neste capítulo, exploraremos as principais propriedades de layout no CSS, incluindo o Box Model, Display, Position, Float e Clear.

### 3.1 Box Model

O Box Model é uma das partes fundamentais do layout em CSS. Cada elemento na página é considerado uma caixa, que é composta por quatro áreas principais:

- **Content:** A área onde o conteúdo do elemento é exibido.
- **Padding:** Espaço entre o conteúdo e a borda do elemento.
- **Border:** A borda ao redor do padding.
- **Margin:** Espaço fora da borda do elemento, separando-o de outros elementos.

A seguir, um exemplo básico do Box Model:

```
.box {  
    width: 200px;  
    padding: 20px;  
    border: 5px solid black;  
    margin: 15px;  
}
```

## 3.2 Display

A propriedade ‘display’ controla como os elementos são exibidos na página. Os valores mais comuns para a propriedade ‘display’ são:

- **block**: O elemento é exibido como um bloco, ocupando toda a largura disponível e começando em uma nova linha.
- **inline**: O elemento é exibido na mesma linha que os elementos adjacentes, ocupando apenas a largura necessária.
- **inline-block**: O elemento é exibido como um bloco, mas permite que elementos adjacentes sejam exibidos na mesma linha.
- **flex**: O elemento é exibido como um contêiner flexível, permitindo o layout flexível de seus filhos.
- **grid**: O elemento é exibido como um contêiner de grid, permitindo um layout baseado em grade para seus filhos.
- **none**: O elemento não é exibido.

Exemplo de uso da propriedade ‘display’:

```
.container {  
    display: flex;  
    justify-content: center;  
}  
.item {  
    display: inline-block;  
    margin: 10px;  
}
```

## 3.3 Position

A propriedade ‘position’ define como um elemento é posicionado na página. Os valores principais para ‘position’ são:

- **static**: O posicionamento é normal, o elemento é posicionado de acordo com o fluxo do documento.
- **relative**: O elemento é posicionado relativamente à sua posição original.

- **absolute:** O elemento é posicionado relativamente ao seu contêiner mais próximo com posição diferente de ‘static’.
- **fixed:** O elemento é posicionado fixamente em relação à janela de visualização e não se move com a rolagem da página.
- **sticky:** O elemento é tratado como ‘relative’ até que ele atinja um limite de rolagem definido, após o qual se torna ‘fixed’.

Exemplo de uso da propriedade ‘position’:

```
.relative-box {
  position: relative;
  top: 10px;
  left: 20px;
}
.absolute-box {
  position: absolute;
  top: 50px;
  right: 50px;
}
```

## 3.4 Float

A propriedade ‘float’ permite que um elemento flutue para a esquerda ou direita dentro de seu contêiner, permitindo que o texto e outros elementos o envolvam. Os valores principais para ‘float’ são:

- **left:** O elemento flutua para a esquerda.
- **right:** O elemento flutua para a direita.
- **none:** O elemento não flutua.

Exemplo de uso da propriedade ‘float’:

```
.float-left {
  float: left;
  margin: 10px;
}
.float-right {
  float: right;
  margin: 10px;
}
```

## 3.5 Clear

A propriedade ‘clear’ é usada para controlar o comportamento dos elementos seguintes em relação aos elementos que flutuam. Os valores principais para ‘clear’ são:

- **left:** O elemento não pode aparecer ao lado de elementos flutuantes à esquerda.
- **right:** O elemento não pode aparecer ao lado de elementos flutuantes à direita.
- **both:** O elemento não pode aparecer ao lado de elementos flutuantes nem à esquerda nem à direita.
- **none:** O elemento pode aparecer ao lado de elementos flutuantes.

Exemplo de uso da propriedade ‘clear’:

```
.clear-both {  
    clear: both;  
}
```



# Chapter 4

## Capítulo 4 - Flexbox

O Flexbox, ou Flexible Box Layout, é um modelo de layout CSS que facilita a criação de layouts flexíveis e responsivos. Introduzido com CSS3, o Flexbox permite distribuir espaço entre itens de um contêiner e alinhar itens de forma eficiente, mesmo quando seu tamanho é desconhecido ou dinâmico.

### 4.1 Conceitos Básicos

O Flexbox é baseado em dois conceitos principais: o contêiner flexível e os itens flexíveis.

#### 4.1.1 Contêiner Flexível

Um contêiner flexível é criado aplicando a propriedade ‘display: flex’ ou ‘display: inline-flex’ a um elemento. O ‘display: flex’ cria um contêiner flexível em bloco, enquanto ‘display: inline-flex’ cria um contêiner flexível em linha.

```
.container {  
    display: flex;  
}
```

#### 4.1.2 Itens Flexíveis

Os itens dentro do contêiner flexível são os filhos diretos e podem ser manipulados usando várias propriedades Flexbox.

## 4.2 Propriedades do Contêiner Flexível

As propriedades do contêiner flexível controlam a disposição e o alinhamento dos itens dentro do contêiner.

### 4.2.1 Propriedade ‘flex-direction’

Define a direção dos itens dentro do contêiner. Os valores possíveis são:

- **row**: Os itens são dispostos na horizontal (padrão).
- **column**: Os itens são dispostos na vertical.
- **row-reverse**: Os itens são dispostos na horizontal, mas na ordem inversa.
- **column-reverse**: Os itens são dispostos na vertical, mas na ordem inversa.

Exemplo:

```
.container {  
  display: flex;  
  flex-direction: row; /* ou column, row-reverse, column-reverse */  
}
```

### 4.2.2 Propriedade ‘flex-wrap’

Define se os itens devem ser quebrados em várias linhas ou colunas, caso não caibam no contêiner.

- **nowrap**: Todos os itens ficam em uma única linha (padrão).
- **wrap**: Os itens são quebrados em várias linhas.
- **wrap-reverse**: Os itens são quebrados em várias linhas, mas na ordem inversa.

Exemplo:

```
.container {  
  display: flex;  
  flex-wrap: wrap; /* ou nowrap, wrap-reverse */  
}
```

### 4.2.3 Propriedade ‘flex-flow‘

Atalho para ‘flex-direction‘ e ‘flex-wrap‘.

Exemplo:

```
.container {  
    display: flex;  
    flex-flow: row wrap; /* ou column nowrap, row-reverse wrap-reverse, etc. */  
}
```

### 4.2.4 Propriedade ‘justify-content‘

Controla o alinhamento dos itens ao longo do eixo principal.

- **flex-start**: Alinha os itens ao início do contêiner (padrão).
- **flex-end**: Alinha os itens ao final do contêiner.
- **center**: Alinha os itens ao centro do contêiner.
- **space-between**: Distribui os itens uniformemente com o primeiro item alinhado ao início e o último ao final.
- **space-around**: Distribui os itens uniformemente com espaços iguais ao redor dos itens.
- **space-evenly**: Distribui os itens uniformemente com espaços iguais entre todos os itens.

Exemplo:

```
.container {  
    display: flex;  
    justify-content: center; /* ou flex-start, flex-end, space-between, space-around */  
}
```

### 4.2.5 Propriedade ‘align-items‘

Controla o alinhamento dos itens ao longo do eixo transversal.

- **stretch**: Os itens se estendem para preencher o contêiner (padrão).
- **flex-start**: Alinha os itens ao início do eixo transversal.

- **flex-end**: Alinha os itens ao final do eixo transversal.
- **center**: Alinha os itens ao centro do eixo transversal.
- **baseline**: Alinha os itens ao longo da linha de base do texto.

Exemplo:

```
.container {
  display: flex;
  align-items: center; /* ou stretch, flex-start, flex-end, baseline */
}
```

#### 4.2.6 Propriedade ‘align-content’

Controla o alinhamento das linhas de itens dentro do contêiner quando há várias linhas.

- **stretch**: Estica as linhas para preencher o contêiner (padrão).
- **flex-start**: Alinha as linhas ao início do contêiner.
- **flex-end**: Alinha as linhas ao final do contêiner.
- **center**: Alinha as linhas ao centro do contêiner.
- **space-between**: Distribui as linhas uniformemente com a primeira linha alinhada ao início e a última ao final.
- **space-around**: Distribui as linhas uniformemente com espaços iguais ao redor das linhas.

Exemplo:

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: space-around; /* ou stretch, flex-start, flex-end, center, space-between */
}
```

### 4.3 Propriedades dos Itens Flexíveis

As propriedades dos itens flexíveis controlam o comportamento dos itens dentro de um contêiner flexível.

### 4.3.1 Propriedade ‘flex-grow’

Define a proporção de espaço disponível que um item deve ocupar em relação aos outros itens.

```
.item {  
    flex-grow: 1; /* ou qualquer valor numérico */  
}
```

### 4.3.2 Propriedade ‘flex-shrink’

Define a proporção de quanto um item deve encolher em relação aos outros itens quando há falta de espaço.

```
.item {  
    flex-shrink: 1; /* ou qualquer valor numérico */  
}
```

### 4.3.3 Propriedade ‘flex-basis’

Define o tamanho inicial de um item antes de qualquer espaço extra ser distribuído.

```
.item {  
    flex-basis: 200px; /* ou qualquer valor de comprimento */  
}
```

### 4.3.4 Propriedade ‘flex’

Atalho para as propriedades ‘flex-grow’, ‘flex-shrink’ e ‘flex-basis’.

```
.item {  
    flex: 1 1 200px; /* ou valores para flex-grow, flex-shrink e flex-basis */  
}
```

### 4.3.5 Propriedade ‘align-self’

Permite o alinhamento de um item específico ao longo do eixo transversal, sobrepondo a configuração do contêiner.

```
.item {  
    align-self: center; /* ou auto, flex-start, flex-end, baseline, stretch */  
}
```

# Chapter 5

## Capítulo 5 - Grid Layout

O Grid Layout, ou CSS Grid Layout, é um sistema de layout bidimensional que permite criar layouts complexos de maneira eficiente. Introduzido com o CSS3, o Grid Layout oferece um controle preciso sobre as linhas e colunas de um contêiner, facilitando a criação de layouts responsivos e alinhamentos complexos.

### 5.1 Conceitos Básicos

O Grid Layout é baseado em dois conceitos principais: o contêiner grid e os itens grid.

#### 5.1.1 Contêiner Grid

Um contêiner grid é criado aplicando a propriedade ‘display: grid’ ou ‘display: inline-grid’ a um elemento. O ‘display: grid’ cria um contêiner grid em bloco, enquanto ‘display: inline-grid’ cria um contêiner grid em linha.

```
.container {  
    display: grid;  
}
```

#### 5.1.2 Itens Grid

Os itens dentro do contêiner grid são os filhos diretos e podem ser manipulados usando várias propriedades do Grid Layout.

## 5.2 Propriedades do Contêiner Grid

As propriedades do contêiner grid definem o comportamento das linhas e colunas do grid.

### 5.2.1 Propriedade ‘grid-template-columns’ e ‘grid-template-rows’

Define a estrutura das colunas e linhas do grid.

- **grid-template-columns**: Define a largura das colunas do grid.
- **grid-template-rows**: Define a altura das linhas do grid.

Exemplo:

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr; /* Define três colunas com frações */  
  grid-template-rows: 100px auto 100px; /* Define três linhas com alturas fixas e auto */  
}
```

### 5.2.2 Propriedade ‘grid-template-areas’

Define um layout de áreas nomeadas dentro do grid.

```
.container {  
  display: grid;  
  grid-template-areas:  
    "header header header"  
    "main main sidebar"  
    "footer footer footer";  
}
```

### 5.2.3 Propriedade ‘grid-column’ e ‘grid-row’

Define em que coluna e linha um item deve começar e terminar.

- **grid-column-start**: Define o início da coluna para o item.
- **grid-column-end**: Define o fim da coluna para o item.
- **grid-row-start**: Define o início da linha para o item.

- **grid-row-end**: Define o fim da linha para o item.

Exemplo:

```
.item {
  grid-column: 1 / 3; /* 0 item ocupa da coluna 1 até a coluna 3 */
  grid-row: 2 / 4;    /* 0 item ocupa da linha 2 até a linha 4 */
}
```

### 5.2.4 Propriedade ‘grid-gap’

Define o espaçamento entre as linhas e colunas do grid.

- **grid-row-gap**: Define o espaçamento entre as linhas.
- **grid-column-gap**: Define o espaçamento entre as colunas.
- **grid-gap**: Atalho para ‘grid-row-gap’ e ‘grid-column-gap’.

Exemplo:

```
.container {
  display: grid;
  grid-gap: 10px; /* Espaçamento de 10px entre linhas e colunas */
}
```

## 5.3 Propriedades dos Itens Grid

As propriedades dos itens grid controlam o comportamento dos itens dentro do grid.

### 5.3.1 Propriedade ‘justify-self’

Controla o alinhamento horizontal de um item dentro de sua célula do grid.

- **start**: Alinha o item ao início da célula.
- **end**: Alinha o item ao final da célula.
- **center**: Alinha o item ao centro da célula.
- **stretch**: Estica o item para preencher a célula (padrão).

Exemplo:

```
.item {
  justify-self: center; /* Alinha o item horizontalmente ao centro da célula */
}
```



### 5.3.2 Propriedade ‘align-self‘

Controla o alinhamento vertical de um item dentro de sua célula do grid.

- **start**: Alinha o item ao início da célula.
- **end**: Alinha o item ao final da célula.
- **center**: Alinha o item ao centro da célula.
- **stretch**: Estica o item para preencher a célula (padrão).

Exemplo:

```
.item {  
    align-self: center; /* Alinha o item verticalmente ao centro da célula */  
}
```

### 5.3.3 Propriedade ‘grid-auto-columns‘ e ‘grid-auto-rows‘

Define o tamanho automático das colunas e linhas criadas automaticamente pelo grid.

- **grid-auto-columns**: Define o tamanho das colunas criadas automaticamente.
- **grid-auto-rows**: Define o tamanho das linhas criadas automaticamente.

Exemplo:

```
.container {  
    display: grid;  
    grid-auto-columns: 100px; /* Colunas automáticas com 100px de largura */  
    grid-auto-rows: 100px;    /* Linhas automáticas com 100px de altura */  
}
```

# Chapter 6

## Capítulo 6 - Estilização de Texto

A estilização de texto em CSS permite ajustar a aparência do texto de maneira detalhada. As propriedades de estilização incluem controle de fonte, tamanho, cor, alinhamento e outros aspectos importantes para a formatação de textos.

### 6.1 Propriedades de Fonte

As propriedades de fonte controlam a aparência do texto, incluindo o tipo de fonte, tamanho e estilo.

#### 6.1.1 Propriedade ‘font-family’

Define o tipo de fonte a ser usado para o texto. Pode incluir fontes genéricas como serif, sans-serif e monospace, além de fontes específicas.

```
p {  
    font-family: Arial, sans-serif; /* Define a fonte Arial, com uma fallback para  
}
```

#### 6.1.2 Propriedade ‘font-size’

Define o tamanho da fonte. Pode ser especificado em pixels (px), em (em), rem, porcentagem (

```
p {  
    font-size: 16px; /* Define o tamanho da fonte como 16 pixels */  
}
```

### 6.1.3 Propriedade ‘font-weight’

Define o peso da fonte, que pode ser normal, negrito ou um valor numérico específico.

```
p {  
    font-weight: bold; /* Define a fonte como negrito */  
}
```

### 6.1.4 Propriedade ‘font-style’

Define o estilo da fonte, como normal, itálico ou oblíquo.

```
p {  
    font-style: italic; /* Define o texto como itálico */  
}
```

### 6.1.5 Propriedade ‘font-variant’

Define variantes de fontes como maiúsculas pequenas.

```
p {  
    font-variant: small-caps; /* Define o texto como maiúsculas pequenas */  
}
```

## 6.2 Propriedades de Cor

As propriedades de cor controlam a cor do texto e o fundo.

### 6.2.1 Propriedade ‘color’

Define a cor do texto. Pode ser especificado usando nomes de cores, valores hexadecimais, RGB, RGBA, HSL ou HSLA.

```
p {  
    color: #333; /* Define a cor do texto como um tom de cinza escuro */  
}
```

### 6.2.2 Propriedade ‘background-color‘

Define a cor de fundo do texto.

```
p {  
    background-color: #f0f0f0; /* Define a cor de fundo como um tom claro de cinza */  
}
```

## 6.3 Propriedades de Alinhamento

As propriedades de alinhamento controlam como o texto é posicionado dentro de seu contêiner.

### 6.3.1 Propriedade ‘text-align‘

Define o alinhamento do texto dentro de seu contêiner. Os valores possíveis são left (esquerda), right (direita), center (centro) e justify (justificado).

```
p {  
    text-align: center; /* Alinha o texto ao centro do contêiner */  
}
```

### 6.3.2 Propriedade ‘text-indent‘

Define o recuo do texto na primeira linha.

```
p {  
    text-indent: 20px; /* Define um recuo de 20 pixels para a primeira linha do parágrafo */  
}
```

### 6.3.3 Propriedade ‘line-height‘

Define a altura da linha, que afeta o espaçamento entre linhas de texto.

```
p {  
    line-height: 1.5; /* Define a altura da linha como 1.5 vezes o tamanho da fonte */  
}
```

## 6.4 Propriedades de Espaçamento

As propriedades de espaçamento controlam o espaço ao redor e entre o texto.

### 6.4.1 Propriedade ‘letter-spacing’

Define o espaçamento entre letras.

```
p {  
    letter-spacing: 1px; /* Define o espaçamento entre letras como 1 pixel */  
}
```

### 6.4.2 Propriedade ‘word-spacing’

Define o espaçamento entre palavras.

```
p {  
    word-spacing: 2px; /* Define o espaçamento entre palavras como 2 pixels */  
}
```

### 6.4.3 Propriedade ‘text-transform’

Define a transformação do texto, como maiúsculas, minúsculas ou capitalização.

```
p {  
    text-transform: uppercase; /* Converte todo o texto para maiúsculas */  
}
```

# Chapter 7

## Capítulo 7 - Cores e Imagens

O uso de cores e imagens é essencial para criar interfaces visuais atraentes e funcionais. O CSS fornece várias propriedades para definir cores e manipular imagens em uma página web.

### 7.1 Cores

As cores podem ser especificadas de várias maneiras no CSS, incluindo nomes de cores, valores hexadecimais, RGB, RGBA, HSL e HSLA.

#### 7.1.1 Nomes de Cores

O CSS suporta uma lista de nomes de cores predefinidos, como "red", "blue", "green", etc.

```
p {  
    color: blue; /* Define a cor do texto como azul */  
}
```

#### 7.1.2 Valores Hexadecimais

Os valores hexadecimais são especificados com um # seguido por 3 ou 6 dígitos hexadecimais.

```
p {  
    color: #ff5733; /* Define a cor do texto como um tom de laranja */  
}
```

### 7.1.3 Valores RGB

Os valores RGB são especificados como ‘rgb(red, green, blue)’, onde cada componente é um valor entre 0 e 255.

```
p {  
    color: rgb(255, 87, 51); /* Define a cor do texto como um tom de laranja */  
}
```

### 7.1.4 Valores RGBA

Os valores RGBA são semelhantes aos valores RGB, mas incluem um componente alpha para a transparência.

```
p {  
    color: rgba(255, 87, 51, 0.5); /* Define a cor do texto com 50% de opacidade */  
}
```

### 7.1.5 Valores HSL

Os valores HSL são especificados como ‘hsl(hue, saturation, lightness)’, onde “hue” é o matiz em graus, “saturation” é a saturação em porcentagem e “lightness” é a luminosidade em porcentagem.

```
p {  
    color: hsl(9, 100%, 60%); /* Define a cor do texto como um tom de laranja */  
}
```

### 7.1.6 Valores HSLA

Os valores HSLA são semelhantes aos valores HSL, mas incluem um componente alpha para a transparência.

```
p {  
    color: hsla(9, 100%, 60%, 0.5); /* Define a cor do texto com 50% de opacidade */  
}
```

## 7.2 Imagens

O CSS fornece várias propriedades para incorporar e manipular imagens em uma página web.

### 7.2.1 Propriedade ‘background-image‘

Define uma imagem de fundo para um elemento.

```
.container {  
    background-image: url('background.jpg'); /* Define a imagem de fundo como 'backgroun
```

### 7.2.2 Propriedade ‘background-repeat‘

Define se a imagem de fundo deve ser repetida.

```
.container {  
    background-image: url('background.jpg');  
    background-repeat: no-repeat; /* Não repete a imagem de fundo */  
}
```

### 7.2.3 Propriedade ‘background-size‘

Define o tamanho da imagem de fundo.

```
.container {  
    background-image: url('background.jpg');  
    background-size: cover; /* A imagem cobre completamente o contêiner */  
}
```

### 7.2.4 Propriedade ‘background-position‘

Define a posição da imagem de fundo dentro do contêiner.

```
.container {  
    background-image: url('background.jpg');  
    background-position: center; /* Centraliza a imagem de fundo */  
}
```

### 7.2.5 Propriedade ‘opacity‘

Define a opacidade de um elemento, incluindo suas imagens.

```
img {  
    opacity: 0.5; /* Define a opacidade da imagem como 50% */  
}
```



## 7.3 Imagens Responsivas

Para tornar as imagens responsivas, você pode usar a propriedade ‘max-width’.

```
img {  
    max-width: 100%; /* Faz com que a imagem não exceda a largura do seu contêiner */  
    height: auto;    /* Mantém a proporção da imagem */  
}
```

# Chapter 8

## Capítulo 8 - Bordas e Sombras

A estilização de bordas e sombras é fundamental para criar efeitos visuais e destacar elementos em uma página web. CSS fornece várias propriedades para configurar bordas e aplicar sombras de forma flexível.

### 8.1 Bordas

As bordas podem ser personalizadas em termos de estilo, largura e cor.

#### 8.1.1 Propriedade ‘border’

Define a borda de um elemento, combinando largura, estilo e cor em uma única declaração.

```
.box {  
    border: 2px solid #333; /* Define uma borda sólida de 2 pixels de largura e cor */  
}
```

#### 8.1.2 Propriedade ‘border-width’

Define a largura da borda. Pode ser especificado em pixels (px), em (em), rem e outros valores.

```
.box {  
    border-width: 4px; /* Define a largura da borda como 4 pixels */  
}
```

### 8.1.3 Propriedade ‘border-style‘

Define o estilo da borda. Os valores possíveis são solid (sólida), dashed (tracejada), dotted (pontilhada), double (dupla) e outros.

```
.box {  
    border-style: dashed; /* Define a borda como tracejada */  
}
```

### 8.1.4 Propriedade ‘border-color‘

Define a cor da borda.

```
.box {  
    border-color: #ff5733; /* Define a cor da borda como um tom de laranja */  
}
```

### 8.1.5 Propriedade ‘border-radius‘

Define o raio das bordas, permitindo criar bordas arredondadas.

```
.box {  
    border-radius: 10px; /* Define bordas arredondadas com um raio de 10 pixels */  
}
```

## 8.2 Sombras

As sombras podem ser aplicadas a textos e elementos para criar efeitos visuais interessantes.

### 8.2.1 Propriedade ‘box-shadow‘

Aplica uma sombra ao redor de um elemento. Pode ser configurada com deslocamento horizontal e vertical, borrão, espalhamento e cor.

```
.box {  
    box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.3); /* Sombra com deslocamento de 4px  
}
```

### 8.2.2 Propriedade ‘text-shadow‘

Aplica uma sombra ao texto, permitindo criar efeitos de profundidade.

```
p {  
    text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.5); /* Sombra do texto com deslocamento */  
}
```

## 8.3 Efeitos Combinados

Combinar bordas e sombras pode criar efeitos visuais interessantes.

### 8.3.1 Exemplo Combinado

```
.box {  
    border: 3px solid #333; /* Define uma borda sólida de 3 pixels */  
    border-radius: 15px; /* Define bordas arredondadas com um raio de 15 pixels */  
    box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.5); /* Aplica uma sombra ao redor do elemento */  
}
```

# Chapter 9

## Capítulo 9 - Transições e Animações

Transições e animações em CSS permitem criar efeitos visuais dinâmicos e atraentes, melhorando a experiência do usuário e a interatividade das páginas web. Este capítulo aborda as principais propriedades e técnicas para implementar transições e animações.

### 9.1 Transições

As transições permitem a mudança suave entre estados de um elemento, criando um efeito de transição ao alterar propriedades CSS.

#### 9.1.1 Propriedade ‘transition’

A propriedade ‘transition’ é usada para especificar o tempo e o tipo de transição entre os estados de um elemento. Pode ser configurada para várias propriedades.

```
.box {  
    transition: background-color 0.5s ease, transform 0.3s ease-in-out;  
    /* Define transições para a cor de fundo e transformação com diferentes durações */  
}
```

#### 9.1.2 Propriedade ‘transition-property’

Define quais propriedades devem ser animadas durante a transição.

```
.box {  
    transition-property: background-color, transform; /* Define as propriedades a serem animadas */  
}
```

### 9.1.3 Propriedade ‘transition-duration‘

Define a duração da transição.

```
.box {  
    transition-duration: 0.5s; /* Define a duração da transição como 0.5 segundos */  
}
```

### 9.1.4 Propriedade ‘transition-timing-function‘

Define o tipo de função de tempo da transição, como ease, linear, ease-in, ease-out, etc.

```
.box {  
    transition-timing-function: ease-in-out; /* Define a função de tempo da transição */  
}
```

### 9.1.5 Propriedade ‘transition-delay‘

Define o atraso antes do início da transição.

```
.box {  
    transition-delay: 0.2s; /* Define um atraso de 0.2 segundos antes do início da transição */  
}
```

## 9.2 Animações

As animações permitem criar sequências de mudanças em um elemento, proporcionando efeitos mais complexos e variados do que as transições.

### 9.2.1 Propriedade ‘animation‘

A propriedade ‘animation‘ é usada para definir animações, incluindo nome, duração, função de tempo, e mais.

```
.box {  
    animation: slideIn 1s ease-in-out infinite;  
    /* Define uma animação chamada 'slideIn' com duração de 1 segundo, tipo de easing 'ease-in-out', e repetição infinita */  
}
```

### 9.2.2 Propriedade '@keyframes'

Define os estados intermediários de uma animação, especificando as mudanças que ocorrerão ao longo do tempo.

```
@keyframes slideIn {
  from {
    transform: translateX(-100%);
    /* Começa com o elemento fora da tela à esquerda */
  }
  to {
    transform: translateX(0);
    /* Move o elemento para a posição original */
  }
}
```

### 9.2.3 Propriedade 'animation-name'

Define o nome da animação a ser aplicada.

```
.box {
  animation-name: slideIn; /* Aplica a animação chamada 'slideIn' */
}
```

### 9.2.4 Propriedade 'animation-duration'

Define a duração da animação.

```
.box {
  animation-duration: 1s; /* Define a duração da animação como 1 segundo */
}
```

### 9.2.5 Propriedade 'animation-timing-function'

Define a função de tempo da animação.

```
.box {
  animation-timing-function: ease-in-out; /* Define a função de tempo da animação */
}
```

### 9.2.6 Propriedade ‘animation-delay‘

Define o atraso antes do início da animação.

```
.box {  
    animation-delay: 0.5s; /* Define um atraso de 0.5 segundos antes do início da a  
}
```

### 9.2.7 Propriedade ‘animation-iteration-count‘

Define o número de vezes que a animação deve ser repetida.

```
.box {  
    animation-iteration-count: infinite; /* Define que a animação deve repetir inf  
}
```

### 9.2.8 Propriedade ‘animation-direction‘

Define a direção da animação, como normal, reverse, alternate e alternate-reverse.

```
.box {  
    animation-direction: alternate; /* Define que a animação deve alternar entre o  
}
```



# Chapter 10

## Capítulo 10 - Responsividade e Media Queries

A responsividade é um aspecto crucial do design web moderno, permitindo que o layout e o conteúdo se ajustem a diferentes tamanhos de tela e dispositivos. Media Queries são uma ferramenta essencial para criar designs responsivos, adaptando a aparência da página de acordo com as características do dispositivo.

### 10.1 Introdução à Responsividade

Design responsivo é uma abordagem que permite que o layout da página se ajuste automaticamente para se adequar a diferentes tamanhos de tela, desde desktops grandes até dispositivos móveis pequenos.

### 10.2 Media Queries

Media Queries são regras CSS que aplicam estilos diferentes com base em características do dispositivo, como a largura da tela, a resolução e a orientação.

#### 10.2.1 Sintaxe Básica

A sintaxe básica de uma Media Query consiste em uma expressão condicional que define as condições para aplicar os estilos.

```
@media (min-width: 768px) {  
    /* Estilos aplicados quando a largura da tela é maior ou igual a 768 pixels */  
    .container {  
        width: 750px;  
    }  
}
```

```
}  
}
```

### 10.2.2 Media Queries para Diferentes Dispositivos

Você pode usar Media Queries para aplicar estilos específicos para diferentes tipos de dispositivos, como smartphones, tablets e desktops.

#### Dispositivos Móveis

Para aplicar estilos a dispositivos móveis com uma largura máxima de 600 pixels:

```
@media (max-width: 600px) {  
    body {  
        font-size: 14px; /* Ajusta o tamanho da fonte para dispositivos móveis */  
    }  
}
```

#### Tablets

Para aplicar estilos a tablets com uma largura mínima de 601 pixels e máxima de 1024 pixels:

```
@media (min-width: 601px) and (max-width: 1024px) {  
    .sidebar {  
        display: none; /* Oculta a barra lateral em tablets */  
    }  
}
```

#### Desktops

Para aplicar estilos a desktops com uma largura mínima de 1025 pixels:

```
@media (min-width: 1025px) {  
    .container {  
        width: 960px; /* Define uma largura fixa para grandes telas */  
    }  
}
```

## 10.3 Propriedades Avançadas de Media Queries

Além da largura da tela, Media Queries podem considerar outras características do dispositivo.

### 10.3.1 Orientação

Você pode aplicar estilos com base na orientação do dispositivo (retrato ou paisagem).

```
@media (orientation: portrait) {  
  .container {  
    padding: 10px; /* Adiciona padding em orientação retrato */  
  }  
}  
  
@media (orientation: landscape) {  
  .container {  
    padding: 20px; /* Adiciona padding em orientação paisagem */  
  }  
}
```

### 10.3.2 Resolução

Você pode aplicar estilos com base na resolução da tela.

```
@media (min-resolution: 192dpi) {  
  .high-res {  
    background-image: url('high-res-background.jpg'); /* Define uma imagem de alta resolução */  
  }  
}
```

## 10.4 Exemplo Completo

A seguir, um exemplo completo que combina diferentes Media Queries para criar um layout responsivo.

```
@media (max-width: 600px) {  
  body {  
    font-size: 14px;  
  }  
  .container {  
    padding: 10px;  
  }  
}
```

```
@media (min-width: 601px) and (max-width: 1024px) {  
    .sidebar {  
        display: none;  
    }  
    .container {  
        padding: 20px;  
    }  
}  
  
@media (min-width: 1025px) {  
    .container {  
        width: 960px;  
        margin: 0 auto;  
    }  
}
```

# Chapter 11

## Capítulo 11 - Transformações

As transformações em CSS permitem modificar a posição, tamanho, orientação e outros aspectos dos elementos em uma página web. Essas transformações podem ser aplicadas individualmente ou combinadas para criar efeitos visuais complexos e dinâmicos.

### 11.1 Transformações Básicas

As transformações básicas incluem mover, redimensionar, rotacionar e inclinar elementos.

#### 11.1.1 Propriedade ‘transform’

A propriedade ‘transform’ aplica uma ou mais transformações a um elemento. As transformações são especificadas como valores separados por espaços.

```
.box {  
    transform: rotate(45deg) scale(1.2) translateX(30px);  
    /* Aplica rotação de 45 graus, escala de 1.2 vezes e deslocamento horizontal de 30 pixels */  
}
```

#### 11.1.2 Transformação ‘translate’

A função ‘translate’ move um elemento em relação à sua posição original.

```
.box {  
    transform: translate(50px, 100px);  
    /* Move o elemento 50 pixels para a direita e 100 pixels para baixo */  
}
```

### 11.1.3 Transformação ‘scale’

A função ‘scale’ altera o tamanho de um elemento.

```
.box {  
    transform: scale(1.5);  
    /* Aumenta o tamanho do elemento em 50% */  
}
```

### 11.1.4 Transformação ‘rotate’

A função ‘rotate’ gira um elemento em torno de seu ponto de origem.

```
.box {  
    transform: rotate(30deg);  
    /* Gira o elemento em 30 graus */  
}
```

### 11.1.5 Transformação ‘skew’

A função ‘skew’ inclina um elemento em um ou ambos os eixos.

```
.box {  
    transform: skewX(20deg);  
    /* Inclina o elemento ao longo do eixo X em 20 graus */  
}
```

### 11.1.6 Transformações Múltiplas

Transformações podem ser combinadas para criar efeitos mais complexos.

```
.box {  
    transform: translateY(20px) rotate(45deg) scale(1.2);  
    /* Move o elemento 20 pixels para baixo, gira 45 graus e aumenta o tamanho em 20% */  
}
```

## 11.2 Transformações 3D

CSS também suporta transformações 3D, permitindo criar efeitos de profundidade e perspectiva.

### 11.2.1 Propriedade ‘perspective’

A propriedade ‘perspective’ define a profundidade da perspectiva aplicada a um elemento 3D.

```
.container {  
    perspective: 500px;  
    /* Define a profundidade da perspectiva em 500 pixels */  
}
```

### 11.2.2 Transformação ‘rotateX’ e ‘rotateY’

As funções ‘rotateX’ e ‘rotateY’ giram um elemento ao redor dos eixos X e Y, respectivamente, criando efeitos 3D.

```
.box {  
    transform: rotateX(45deg) rotateY(30deg);  
    /* Gira o elemento 45 graus ao redor do eixo X e 30 graus ao redor do eixo Y */  
}
```

### 11.2.3 Transformação ‘translateZ’

A função ‘translateZ’ move um elemento ao longo do eixo Z, criando um efeito de profundidade.

```
.box {  
    transform: translateZ(100px);  
    /* Move o elemento 100 pixels para fora da tela (em direção ao observador) */  
}
```

### 11.2.4 Transformações 3D Combinadas

Combinando transformações 3D, é possível criar efeitos visuais mais avançados.

```
.box {  
    transform: rotateX(30deg) rotateY(45deg) translateZ(100px);  
    /* Aplica rotações e deslocamento 3D ao elemento */  
}
```

# Chapter 12

## Capítulo 12 - Pseudo-Classes e Pseudo-Elementos Avançados

Pseudo-classes e pseudo-elementos são ferramentas poderosas no CSS que permitem selecionar e estilizar elementos com base em seu estado ou estrutura específica. Este capítulo explora as pseudo-classes e pseudo-elementos mais avançados e suas aplicações.

### 12.1 Pseudo-Classes

Pseudo-classes são usadas para definir o estado de um elemento ou para selecionar elementos com base em sua posição ou interação.

#### 12.1.1 Pseudo-Classe ‘:nth-child’

A pseudo-classe ‘:nth-child’ permite selecionar elementos com base em sua posição dentro de um elemento pai.

```
ul li:nth-child(odd) {
    background-color: #f2f2f2;
    /* Aplica um fundo cinza claro a itens de lista ímpares */
}

ul li:nth-child(even) {
    background-color: #ffffff;
    /* Aplica um fundo branco a itens de lista pares */
}
```



### 12.1.2 Pseudo-Classe ‘:not‘

A pseudo-classe ‘:not‘ permite aplicar estilos a todos os elementos que não correspondem a um seletor específico.

```
button:not(.disabled) {  
    background-color: #4CAF50;  
    /* Aplica um fundo verde a botões que não têm a classe 'disabled' */  
}
```

### 12.1.3 Pseudo-Classe ‘:nth-of-type‘

A pseudo-classe ‘:nth-of-type‘ seleciona elementos com base na sua posição entre elementos do mesmo tipo.

```
p:nth-of-type(2) {  
    font-weight: bold;  
    /* Aplica negrito ao segundo parágrafo dentro de seu elemento pai */  
}
```

### 12.1.4 Pseudo-Classe ‘:last-child‘

A pseudo-classe ‘:last-child‘ seleciona o último filho de um elemento pai.

```
div p:last-child {  
    margin-bottom: 0;  
    /* Remove a margem inferior do último parágrafo dentro de um div */  
}
```

## 12.2 Pseudo-Elementos

Pseudo-elementos permitem estilizar partes específicas de um elemento ou criar novos elementos virtuais.

### 12.2.1 Pseudo-Elemento ‘::before‘

O pseudo-elemento ‘::before‘ é usado para inserir conteúdo antes do conteúdo de um elemento.

```
h1::before {
    content: " ";
    color: #4CAF50;
    /* Adiciona um ícone de triângulo verde antes do conteúdo de cada título h1 */
}
```

### 12.2.2 Pseudo-Elemento ‘::after‘

O pseudo-elemento ‘::after‘ é usado para inserir conteúdo após o conteúdo de um elemento.

```
p::after {
    content: " [Leia mais]";
    color: #888;
    /* Adiciona o texto '[Leia mais]' após o conteúdo de cada parágrafo */
}
```

### 12.2.3 Pseudo-Elemento ‘::first-line‘

O pseudo-elemento ‘::first-line‘ permite estilizar a primeira linha do conteúdo de um bloco.

```
p::first-line {
    font-weight: bold;
    /* Aplica negrito à primeira linha de cada parágrafo */
}
```

### 12.2.4 Pseudo-Elemento ‘::first-letter‘

O pseudo-elemento ‘::first-letter‘ permite estilizar a primeira letra de um bloco de texto.

```
p::first-letter {
    font-size: 2em;
    color: #4CAF50;
    /* Aumenta o tamanho da primeira letra e aplica uma cor verde */
}
```

## 12.3 Exemplos Combinados

Os pseudo-elementos e pseudo-classes podem ser combinados para criar efeitos complexos.

```
ul li:nth-child(odd)::before {  
    content: "● ";  
    color: #4CAF50;  
    /* Adiciona um marcador verde a itens de lista ímpares */  
}
```

# Bibliography

- [1] W3C. (2021). CSS Selectors. <https://www.w3.org/TR/selectors-3/>
- [2] MDN Web Docs. (2024). CSS Selectors. [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Selectors](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors)
- [3] Freeman, E., Robson, E. (2014). *Head First HTML and CSS*. O'Reilly Media.
- [4] Meyer, E. A. (2011). *CSS3: Visual QuickStart Guide*. Peachpit Press.
- [5] Pilgrim, M. (2010). *HTML5: Up and Running*. O'Reilly Media.
- [6] Refsnes, C. (2015). *HTML5 and CSS3, Level Up with Today's Web Technologies*. Addison-Wesley.
- [7] W3C. (2021). CSS Selectors. <https://www.w3.org/TR/selectors-3/>
- [8] MDN Web Docs. (2024). CSS Selectors. [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Selectors](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors)