Classes Utilitárias

Aprenda a usar todas as classes utilitárias auxiliares do WPF, desde o seu conceito até a aplicação.

- Position
- Offset
- Z-Index
- Overflow
- Text-Align
- Word-Break
- Text-Wrap
- Word-Wrap
- Line-Height
- Font-Size
- Text-Decoration

Position

O atributo **position** é retratado de uma maneira bem natural e intuitiva pelo WPF. Para atribuir classes utilitárias de posição no seu código, basta escrever o nome do valor do atributo diretamente como uma classe css:

```
<div class="relative">
  <!-- O conteúdo dessa classe recebeu o atributo css 'position: relative' -->
</div>

<div class="absolute">
  <!-- O conteúdo dessa classe recebeu o atributo css 'position: absolute' -->
</div>

<div class="sticky">
  <!-- O conteúdo dessa classe recebeu o atributo css 'position: sticky' -->
</div>

<!-- Isso funciona para qualquer outro tipo de atribuição de posicionamento. -->
```

Abaixo segue uma lista com todas as posições suportadas pelo framework:

Classe Utilitária	Comportamento
`static`	*Posicionamento padrão, segue o fluxo normal do documento.*
`sticky`	*Alterna entre* `relative` *e* `fixed`*, fixando o elemento ao atingir um ponto de rolagem definido.*
`relative`	*Posicionado relativo à sua posição original. Permite ajus* `s` `t` `t` `nt`, `tom` e `left`.
`absolute`	*Posicionado em relação ao ancestral mais próximo com po* `o não static`*, sai do fluxo normal.*
`fixed`	*Posicionado em relação à janela do navegador, permanece fixo durante a rolagem.*

Offset

As classes utilitárias de offsets no **WPF** permitem ajustar a posição de elementos com as propriedades CSS top, right, bottom e left. Elas são projetadas para serem intuitivas, seguindo uma sintaxe clara e flexível, compatível com unidades como `%`, `px`, `dvh` e `dvw`.

Como Funcionam as Classes de Offsets

As classes de offsets seguem o padrão: (propriedade)(valor)(unidade), onde:

- **Propriedade**: `top`, `right`, `bottom` ou `left`.
- **Valor**: Um número inteiro (positivo ou negativo) que define o deslocamento.
- **Unidade**: Opcional, pode ser `%`, `px`, `dvh` (altura dinâmica da viewport) e `dvw` (largura dinâmica da viewport). Se omitida, o valor é tratado como unitless (ex.: `top(10)` equivale a `top: 10;`).

Isso significa que uma classe válida segue este formato:

```
<div class="fixed top(50%) left(20px)">
  <!-- Esse elemento html definiu os seguintes atributos css - top: 50%; left: 20px; -->
</div>

<div class="absolute right(-10dvw) bottom(100)">
  <!-- Esse elemento html definiu os seguintes atributos css - right: -10dvw; bottom: 100; -->
</div>
```

Usando Funções de Classe para Centralização

As classes utilitárias de centralização do **WPF** simplificam o alinhamento de elementos horizontalmente, verticalmente ou em ambos os eixos. Elas combinam as propriedades left, top e transform para posicionar elementos com precisão, especialmente em elementos com position: `absolute`, `fixed` ou `sticky`.

Como Funcionam as Classes de Centralização

As classes de centralização seguem uma sintaxe simples e intuitiva: **center(tipo)**, onde o tipo define o eixo de centralização:

As classes de centralização **são ideais para elementos com position:** `absolute`, `fixed` ou `sticky`, **dentro ou fora de um contêiner pai com position semelhante**. Certifique-se de que o elemento pai tenha dimensões definidas de alguma maneira (ex.: width e height) para que a centralização funcione corretamente.

Classe Utilitária	Comportamento
`center(h)`	*Centraliza horizontalmente o elemento.*
`center(v)`	*Centraliza verticalmente o elemento.*
`center(c)`	*Centraliza em **ambos** os eixos (horizontal e vertical).*

Com essa sintaxe, confira alguns exemplos dessas funções de classe em execução:

```
<div class="fixed top(20px) center(h)">
  <!-- No seguinte elemento html, a seguinte configuração css foi atribuída - top: 20px; left: 50%; transform: translateX(-50%); -->
</div>

<div class="absolute top(5%) center(v)">
  <!-- No seguinte elemento html, a seguinte configuração css foi atribuída - left: 5%; top: 50%; transform: translateY(-50%); -->
</div>

<div class="relative center(c)">
  <!-- No seguinte elemento html, a seguinte configuração css foi atribuída - left: 50%; top: 50%; transform: translate(-50%, -50%); -->
</div>
```

Notas

> **Posicionamento**: As classes de centralização requerem que o elemento tenha position: `absolute`, `fixed` ou `sticky`.

> **Contêiner Pai**: Certifique-se de que o elemento pai tenha algum atributo position não estático e dimensões definidas de alguma maneira.

>

> **Transformações Adicionais**: Certifique-se que, ao usar outros tipos de **configurações de transformação**, que elas não sobrescrevam os seus estilos!

Z-Index

As classes utilitárias de z-index do **WPF** permitem controlar a ordem de empilhamento de elementos no eixo Z, definindo qual elemento aparece acima ou abaixo de outros. Elas são ideais para layouts com sobreposições, como modais, menus ou elementos posicionados com absolute ou fixed.

Como Funcionam as Classes de Z-Index

As classes de z-index seguem o padrão: **z-(valor)**, onde:

- **Valor**: Um número inteiro (positivo ou negativo) que define a ordem de empilhamento.
- **Sintaxe**: A classe é formada por *z-* seguido de um número, como `z-10`, `z--5` ou `z-0`.

Regras CSS Geradas

Cada classe é traduzida diretamente para a propriedade CSS z-index. Por exemplo:

- `z-10` → `z-index: 10;`
- `z--5` → `z-index: -5;`
- `z-0` → `z-index: 0;`

Como Usar

As classes de **z-index** são aplicadas a elementos com position: `absolute`, `relative`, `fixed` ou `sticky`, já que o z-index só afeta elementos posicionados. Use-as para controlar a sobreposição de elementos em layouts complexos.

Exemplo 1: Sobreposição Simples

Posicione um elemento acima de outro:

```
<div class="relative w-100% h-200px">
  <div class="absolute top(0) left(0) bg(c-blue) z-10"><!-- Elemento superior --></div>
  <div class="absolute top(10px) left(10px) bg(c-red) z-5"><!-- Elemento inferior --></div>
</div>
```

Exemplo 2: Z-Index Negativo

Coloque um elemento abaixo do conteúdo padrão:

```
<div class="relative w-100% h-200px">
  <div class="absolute top(0) left(0) bg(c-gray) z--1"><!-- Fundo --></div>
  <div class="absolute top(20px) left(20px) bg(c-white)"><!-- Conteúdo principal --></div>
</div>
```

Notas

> **Valores altos**: Use valores como `z-100` ou superiores para elementos que devem ficar **acima de tudo**, como *modais*.

> **Valores negativos**: Use `z--1` ou **inferiores** para **elementos de fundo ou camadas abaixo do conteúdo principal**.

> O **z-index** só funciona em elementos com position diferente de `static`.

> Valores maiores de **z-index** colocam o elemento mais acima na ordem de empilhamento.

> Evite valores extremamente altos (ex.: `z-999999`) para **manter a manutenção do código simples**.

Overflow

As classes utilitárias de `overflow` do **WPF** controlam como o conteúdo excedente de um elemento é tratado, permitindo gerenciar a visibilidade e o comportamento de rolagem em elementos com dimensões limitadas. Elas são ideais para layouts com *contêineres*, *modais* ou áreas de conteúdo **restritas**.

Como Funcionam as Classes de Overflow

As classes de `overflow` seguem uma sintaxe intuitiva que permite definir o comportamento geral ou por eixo **(x ou y)**, com ou sem tipo específico (`hidden`, `scroll`, `visible`). O padrão `overflow` sem modificadores aplica `overflow: auto;` como valor padrão.

Atributos Válidos	Comportamento
`auto`	*Deixa o navegador encarregado do comportamento. Seu funcionamento pode variar.*
`hidden`	*O conteúdo é cortado e nenhuma barra de rolagem é exibida.*
`scroll`	*O conteúdo é acessível através de barras de rolagem que são exibidas mesmo que o conteúdo não precise.*
`visible`	***Valor padrão**. O conteúdo não é cortado e pode ser renderizado para fora da caixa de conteúdo.*

As classes são traduzidas diretamente para propriedades CSS. Veja alguns exemplos:

- `overflow`→ `overflow: auto;`
- `overflow-x` → `overflow-x: auto;`
- `overflow-y-hidden` → `overflow-y: hidden;`
- `overflow-scroll` → `overflow: scroll;`

Como Usar

As classes de `overflow` são aplicadas a elementos com dimensões definidas (ex.: `width`, `height`, `max-width`, `max-height`) para controlar o comportamento do conteúdo que excede essas dimensões. Elas são úteis para criar áreas roláveis, ocultar conteúdo excedente ou manter visibilidade.

Exemplo 1: Overflow Padrão

Aplica rolagem automática para conteúdo excedente em ambas as direções:

```
<div class="w-300px h-200px overflow">
  <p><!-- Conteúdo longo que excederá o contêiner... --></p>
</div>
```

Exemplo 2: Overflow por Eixo

Habilita rolagem horizontal, mas oculta o conteúdo vertical excedente:

```
<div class="w-300px h-200px overflow-x overflow-y-hidden">
  <p><!-- Conteúdo longo horizontalmente e verticalmente... --></p>
</div>
```

Exemplo 3: Overflow com Tipo Específico

Cria uma área com rolagem vertical forçada e conteúdo horizontal visível:

```
<div class="w-300px h-200px overflow-y-scroll overflow-x-visible">
  <p><!-- Conteúdo com rolagem vertical e visibilidade horizontal... --></p>
</div>
```

Exemplo 4: Combinação com Posicionamento

Usa overflow com classes de posicionamento e centralização:

```
<div class="relative w-100% h-400px overflow-hidden">
  <div class="absolute center(c) w-200px h-200px overflow-y-scroll">
    <p><!-- Conteúdo rolável centralizado... --></p>
  </div>
</div>
```

Notas

> **Dimensões**: Sempre defina `width`, `height`, `max-width` ou `max-height` no elemento para que o `overflow` funcione corretamente.

> **Rolagem Automática**: Use `overflow` ou `overflow-(x|y)` para rolagem apenas quando necessário.

>

Combinações: Combine com classes de posicionamento (`absolute`, `relative`, etc.) e offsets (`top`, `left`, etc.) para layouts complexos.

Forçar Rolagem: Aplicar o tipo `scroll` em um elemento evita o problema de barras de rolagem aparecendo e desaparecendo **quando o conteúdo é dinâmico**. Tenha em mente que **Impressoras podem imprimir o conteúdo vazado**.

As classes `overflow-x` e `overflow-y` são úteis para controle granular em layouts responsivos.

O valor `visible` pode causar sobreposição de conteúdo fora do contêiner, então use com cuidado.

Para áreas roláveis, certifique-se de que o contêiner tenha conteúdo suficiente para ativar a rolagem.

Text-Align

Os alinhadores de texto do **WPF** são classes utilitárias que controlam o alinhamento horizontal de texto em elementos HTML. Eles utilizam a propriedade CSS `text-align` para oferecer flexibilidade e simplicidade no ajuste de layouts textuais.

Como Funcionam os Alinhadores de Texto

Os alinhadores seguem as seguintes definições principais:

Classe Utilitária	Comportamento
`t-left`	*Alinha o texto à esquerda.*
`t-center`	*Centraliza o texto.*
`t-right`	*Alinha o texto à direita.*
`t-start`	*Alinha o texto no início da **direção configurada do texto** .*
`t-end`	*Alinha o texto no final da **direção configurada do texto**.*
`t-justify`	*Justifica o texto.*
`t-justify-all`	*Justifica todas as linhas do texto, **incluindo a última**.*

Exemplos de Uso

Alinhamento Básico

```
<p class="t-left"><!-- Texto à esquerda. --></p>
<p class="t-center"><!-- Texto centralizado. --></p>
<p class="t-right"><!-- Texto à direita. --></p>
```

Alinhamento Dinâmico

```
<p class="t-start"><!-- Texto no início (esquerda em LTR). --></p>
<p class="t-end"><!-- Texto no final (direita em LTR). --></p>
```

Justificação

```
<p class="t-justify"><!-- Texto justificado, exceto a última linha. --></p>
<p class="t-justify-all"><!-- Texto justificado, incluindo a última linha. --></p>
```

Notas

> **Suporte a Idiomas**: Use `t-start` e `t-end` para layouts que precisam se adaptar a direções de texto como *RTL (right-to-left)*.

> **Justify vs. Justify-All**: `t-justify` é ideal para parágrafos normais, enquanto `t-justify-all` é útil para textos curtos que exigem alinhamento completo.

> O valor `justify-all` pode **não ser suportado** em navegadores mais antigos; teste a compatibilidade se necessário.

Word-Break

As classes utilitárias de `word-break` no WPF são projetadas para controlar o comportamento de quebra de palavras em elementos HTML, utilizando a propriedade CSS `word-break`. Elas são ideais para gerenciar textos longos, layouts responsivos ou conteúdos em idiomas que não utilizam espaços entre palavras.

Como Funcionam as Classes de Word-Break

As classes seguem uma sintaxe simples e intuitiva, permitindo ajustar a quebra de palavras de acordo com a necessidade. Veja as classes disponíveis:

Classes Utilitárias	Comportamento
`w-nobreak`	*Usa o **comportamento padrão** de quebra de palavras, onde o texto só é quebrado em **espaços em branco** ou **pontos naturais**. Palavras longas podem ultrapassar os limites do contêiner se não houver espaços suficientes.*
`w-break`	*Permite que palavras longas sejam quebradas em pontos arbitrários para se ajustar ao contêiner, evitando overflow. Ideal para textos em idiomas ocidentais com palavras extensas.*
`w-break-all`	*Quebra o texto em qualquer ponto, inclusive dentro de palavras. Útil para idiomas sem espaços, como chinês ou japonês.*
`w-break-keep-all`	*Impede a quebra dentro de palavras, mantendo-as intactas. Pode causar overflow se as palavras forem longas, sendo ideal para preservar a integridade de palavras em idiomas com espaços.*

Como Usar

Aplique essas classes a elementos de texto como `<p>`, `<div>` ou `<span>`. Elas são especialmente úteis em layouts responsivos ou para textos multilíngues com diferentes necessidades de quebra.

Exemplo 1: Comportamento Padrão

```
<div class="w-300px w-nobreak">
  <p>Palavra longa: Supercalifragilisticexpialidocious</p>
  <!-- A palavra longa pode ultrapassar o contêiner de 300px, pois a quebra só ocorre nos espaços. -->
</div>
```

Exemplo 2: Quebra de Palavras Longas

```
<div class="w-300px w-break">
  <p>Palavra longa: Supercalifragilisticexpialidocious</p>
  <!-- A palavra é quebrada para se ajustar ao contêiner, evitando overflow. -->
</div>
```

Exemplo 3: Quebra em Qualquer Ponto

```
<div class="w-300px w-breakall">
  <p>Texto em chinês: 这是一个很长的句子，没有空格。</p>
  <!-- O texto é quebrado em qualquer ponto, ajustando-se ao contêiner mesmo sem espaços. -->
</div>
```

Exemplo 4: Preservar Palavras

```
<div class="w-300px w-break-keep-all">
  <p>Texto em japonês: これは非常に長い文で、単語を壊さないようにします。</p>
  <!-- As palavras são mantidas intactas, mas o texto pode ultrapassar o contêiner se for muito longo. -->
</div>
```

Dicas de Uso

- **Idiomas sem Espaços**: Use `w-break-all` para textos em *chinês*, *japonês* ou *tailandês*.
- **Preservar Palavras**: Prefira `w-break-keep-all` para idiomas como *português* ou *inglês*, mantendo a legibilidade.
- **Combinações**: Combine com classes de `overflow` (ex.: `overflow-hidden`) para gerenciar excessos de texto.
- **Responsividade**: Use `w-break` com larguras dinâmicas (ex.: `w-100%`) para layouts adaptáveis.
- **Testes**: Verifique o comportamento em diferentes navegadores, especialmente para idiomas específicos.

Notas

O uso de `w-break-all` pode afetar a legibilidade; **aplique com cuidado** e teste o resultado visual.

As classes `w-nobreak` e `w-break` são mais comuns em **idiomas ocidentais**, enquanto `w-break-all` e `w-break-keep-all` atendem a necessidades de **idiomas asiáticos** ou **textos técnicos**.

Text-Wrap

As classes utilitárias de `text-wrap` no **WPF** permitem controlar como o texto é quebrado e exibido em um elemento, utilizando a propriedade CSS `text-wrap`. Elas são ideais para ajustar a apresentação de texto em layouts variados, desde evitar quebras de linha até aplicar modos avançados de formatação.

A propriedade `text-wrap` define o comportamento de quebra de linha do texto dentro de um contêiner. Com as classes do WPF, você pode facilmente aplicar esses comportamentos sem escrever CSS manualmente, garantindo consistência e praticidade no design.

Classes Disponíveis

- `t-nowrap`: Impede que o texto quebre em várias linhas, exibindo-o em uma única linha contínua. Útil para títulos ou textos que devem permanecer compactos, mas pode gerar overflow se o conteúdo for maior que o contêiner.
- `t-wrap`: Permite que o texto quebre em linhas de forma natural, conforme o espaço disponível e as regras de quebra de palavras do navegador.
- `t-wrap-auto`: Deixa o navegador escolher o modo de quebra mais adequado.
- `t-wrap-balance`: Distribui o texto de forma equilibrada entre as linhas, ideal para parágrafos curtos.
- `t-wrap-stable`: Garante consistência no layout durante a renderização, evitando mudanças bruscas.
- `t-wrap-pretty`: Prioriza a legibilidade, ajustando as quebras para um fluxo visual mais agradável.

Exemplos de Uso

Exemplo 1: Texto em Linha Única

```
<span class="t-nowrap">Este texto longo não será quebrado em várias linhas.</span>
```

Exemplo 2: Quebra de Linha Padrão

```
<p class="t-wrap">Este texto será quebrado em linhas conforme o espaço disponível no contêiner.</p>
```

Exemplo 3: Modos Específicos

```
<p class="t-wrap-balance">Este parágrafo será balanceado para linhas de tamanhos similares.</p>
<p class="t-wrap-pretty">Este texto é ajustado para melhor legibilidade e fluidez.</p>
```

Dicas e Notas

> **Overflow**: Use `t-nowrap` com classes de overflow (ex.: `overflow-hidden`) para gerenciar texto que ultrapassa o contêiner.

> **Compatibilidade**: Modos como *balance* e *pretty* podem não ser suportados em navegadores antigos. Teste em seu ambiente alvo.

Word-Wrap

As classes utilitárias de `word-wrap` no **WPF** são usadas para controlar o comportamento de quebra de palavras em elementos HTML, aplicando a propriedade CSS `word-wrap`. Elas ajudam a gerenciar textos longos em contêineres com largura limitada.

Classes Disponíveis

- `w-nowrap`: Impede a quebra de palavras, mantendo o texto em uma única linha. Se o texto for muito longo, ele pode ultrapassar os limites do contêiner, **resultando em overflow**.
- `w-wrap`: Permite que palavras longas sejam quebradas em pontos arbitrários, ajustando o texto ao tamanho do contêiner e **evitando overflow**.

Exemplos de Uso

Exemplo 1: Impedir Quebra de Palavras

```
<div class="w-300px w-nowrap">
  <p>Supercalifragilisticexpialidocious</p>
  <!-- A palavra longa não será quebrada e pode exceder a largura de 300px do contêiner. -->
</div>
```

Exemplo 2: Permitir Quebra de Palavras

```
<div class="w-300px w-wrap">
  <p>Supercalifragilisticexpialidocious</p>
  <!-- A palavra será quebrada para se ajustar ao contêiner de 300px. -->
</div>
```

Line-Height

As classes utilitárias de `line-height` no **WPF** permitem ajustar a altura das linhas de texto de forma rápida e consistente. Elas controlam o espaçamento entre linhas, melhorando a legibilidade e a estética do texto em parágrafos, títulos e outros elementos.

Como Funcionam as Classes de Line Height

As classes seguem o padrão **lh-[valor][unidade]**, onde:

- **[valor]** é um número (*inteiro* ou *decimal*) que **define o tamanho da altura da linha**.
- **[unidade]** é uma unidade de medida obrigatória: `px`, `rem`, `em` ou `%`.

Regras CSS Geradas

Cada classe é convertida para a propriedade CSS `line-height`. Por exemplo:

- `lh-20px` → `line-height: 20px`;
- `lh-1.5rem` → `line-height: 1.5rem`;
- `lh-150%` → `line-height: 150%`;

Como Usar

Aplique as classes diretamente aos elementos HTML, como `<p>`, `<h1>`, `<li>`, etc., para ajustar o espaçamento entre linhas.

Exemplo 1: Texto de Corpo

```
<p class="lh-1.6rem">
  <!-- Este parágrafo tem uma altura de linha de 1.6rem, ideal para legibilidade em textos longos. -->
</p>
```

Exemplo 2: Títulos

```
<h2 class="lh-1.2em">
  <!-- Este título usa uma altura de linha mais compacta, perfeita para textos curtos. -->
</h2>
```

Notas

> Essas classes exigem uma unidade de medida (`px`, `rem`, `em`, ou `%`). Valores sem unidade (ex.: `lh-1.5`) não são suportados por este conjunto de utilitários.

Font-Size

As classes utilitárias de `font-size` no **WPF** permitem ajustar o tamanho da fonte de elementos HTML de maneira rápida e consistente. Elas utilizam a propriedade CSS `font-size` e suportam unidades absolutas e relativas, como `px`, `em`, `rem` e `%`, oferecendo flexibilidade para diferentes contextos de design.

Como Funcionam as Classes de Font-Size

As classes seguem o padrão **fs-[valor][unidade]**, onde:

- **[valor]** é um número (*inteiro* ou *decimal*) que **define o tamanho da fonte**.
- **[unidade]** é uma unidade de medida obrigatória: `px`, `em`, `rem` ou `%`.

Regras CSS Geradas

Cada classe é traduzida diretamente para a propriedade CSS `font-size`. Por exemplo:

- `fs-16px` → `font-size: 16px`;
- `fs-1.5rem` → `font-size: 1.5rem`;
- `fs-120%` → `font-size: 120%`;

Como Usar

Aplique as classes diretamente a elementos de texto, como `<p>`, `<h1>`, `<span>`, etc., para ajustar o tamanho da fonte. As classes são ideais para criar hierarquias tipográficas claras e consistentes.

Exemplo 1: Texto de Corpo

```
<p class="fs-16px">Este parágrafo tem fonte de 16px.</p>
<p class="fs-1.2rem">Este parágrafo tem fonte de 1.2rem, relativo ao elemento raiz.</p>
```

Exemplo 2: Títulos

```
<h1 class="fs-32px">Título Principal (32px)</h1>
<h2 class="fs-24px">Subtítulo (24px)</h2>
```

Exemplo 3: Tamanhos Relativos

```
<div class="fs-1.5em">
  <p>Este texto é 1.5 vezes maior que o pai.</p>
  <p class="fs-80%">Este texto é 80% do tamanho do pai (1.5em * 0.8 = 1.2em do original).</p>
</div>
```

Exemplo 4: Percentuais

```
<p class="fs-150%">Este texto é 150% maior que o tamanho da fonte do pai.</p>
```

Dicas e Melhores Práticas

> **Hierarquia**: Combine diferentes tamanhos para criar uma hierarquia visual clara entre títulos, subtítulos e textos de corpo. Mantenha-se consistente em quais tags html utilizar para quais tamanhos;

> **Acessibilidade**: Evite tamanhos muito pequenos (ex.: `fs-10px`) para garantir legibilidade, especialmente em dispositivos móveis. O *tamanho recomendado para texto é a partir de* **12 pixels de tamanho.**

> **Testes**: Verifique o comportamento em diferentes navegadores e tamanhos de tela para garantir que o texto se ajuste corretamente.

Text-Decoration

As classes utilitárias de `text-decoration` no **WPF** permitem adicionar ou remover decorações de texto, como *sublinhados*, *linhas sobre o texto* ou *tachados*, de **forma rápida e consistente**. Elas utilizam a propriedade CSS `text-decoration` e oferecem uma variedade de estilos para personalizar a aparência do texto.

Como Funcionam as Classes de Text-Decoration

As classes seguem o padrão **t-[estilo]**, onde **[estilo]** é um dos valores permitidos: `underline`, `dashed`, `double`, `line-through`, `overline`, `solid`, `wavy` ou `none`.

Classes Utilitárias	Comportamento
`t-underline`	*Adiciona um **sublinhado sólido** ao texto.*
`t-dashed`	*Adiciona um **sublinhado tracejado** ao texto.*
`t-double`	*Adiciona um **sublinhado duplo** ao texto.*
`t-line-through`	*Adiciona um **tachado** ao texto.*
`t-overline`	*Adiciona uma **linha sobre** o texto.*
`t-solid`	*Adiciona um **sublinhado sólido** (similar a `underline`, mas pode ser usado para consistência).*
`t-wavy`	*Adiciona um **sublinhado ondulado** ao texto.*
`t-none`	***Remove qualquer decoração** de texto.*

Regras CSS Geradas

Cada classe é traduzida diretamente para a propriedade CSS `text-decoration`, seguindo o template **.t-[estilo] { text-decoration: [estilo]; }**. Por exemplo:

- `t-underline` → `text-decoration: underline`;
- `t-dashed` → `text-decoration: dashed`;
- `t-none` → `text-decoration: none`;

Exemplos de Uso

Exemplo 1: Sublinhado Simples

```
<p class="t-underline">Este texto está sublinhado.</p>
```

Exemplo 2: Tachado

```
<p class="t-line-through">Este texto está tachado.</p>
```

Exemplo 3: Sublinhado Ondulado

```
<p class="t-wavy">Este texto tem um sublinhado ondulado.</p>
```

Exemplo 4: Remover Decoração

```
<a href="#" class="t-none">Este link não tem sublinhado.</a>
```

Guia de Boas Práticas

> **Acessibilidade**: Use `t-line-through` para indicar **texto excluído** ou **desatualizado**, mas certifique-se de que o significado seja claro para todos os usuários.

> **Consistência**: Use `t-none` para remover sublinhados de links ou outros elementos que por padrão têm decorações.

> **Navegadores**: Alguns estilos, como `dashed` ou `wavy`, podem ter renderizações diferentes em navegadores antigos. Teste em seu ambiente alvo.

Notas

>

As classes de `text-decoration` são aplicadas diretamente ao elemento e afetam todo o texto dentro dele.

Para decorações mais complexas, como múltiplas linhas ou cores específicas, considere usar CSS personalizado.

O valor `solid` é similar a `underline`, mas pode ser útil para manter a consistência na nomenclatura.