

# Principais propriedades do CSS - 2

## Principais propriedades do CSS

**15.** `top`, `right`, `bottom`, `left`: Usado com elementos posicionados para definir a distância entre os lados do elemento e seu pai.

```
.elemento-posicionado {  
  position: absolute;  
  top: 20px; /* Distância de 20 pixels a partir do topo */  
  right: 30px; /* Distância de 30 pixels a partir da direita */  
  bottom: 10px; /* Distância de 10 pixels a partir da parte inferior */  
  left: 40px; /* Distância de 40 pixels a partir da esquerda */  
}
```

Neste exemplo, o elemento com a classe `.elemento-posicionado` é posicionado de forma absoluta e deslocado a distâncias específicas do topo, direita, inferior e esquerda em relação ao seu contêiner pai.

**16.** `float`: Permite que um elemento flutue à esquerda ou à direita de seu contêiner, criando layouts de várias colunas.

```
.flutuante-esquerda {  
  float: left;  
}  
  
.flutuante-direita {  
  float: right;  
}
```

Neste exemplo, os elementos com as classes `.flutuante-esquerda` e `.flutuante-direita` flutuam à esquerda e à direita de seu contêiner, criando um layout de duas colunas.

**17.** `clear`: Controla como os elementos devem se comportar em relação aos elementos flutuantes próximos.

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

Neste exemplo, o elemento com a classe `.limpar-flutuante` é definido para "limpar" quaisquer elementos flutuantes à esquerda e à direita, garantindo que ele não seja afetado pela flutuação de outros elementos próximos.

**18. `z-index`:** Define a ordem de empilhamento de elementos posicionados, determinando quais elementos aparecem acima de outros.

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    .caixa1 {  
      width: 100px;  
      height: 100px;  
      background-color: red;  
      position: absolute;  
      top: 10px;  
      left: 10px;  
      z-index: 1; /* Caixa 1 tem z-index 1 */  
    }  
  
    .caixa2 {  
      width: 100px;  
      height: 100px;  
      background-color: blue;  
      position: absolute;  
      top: 20px;  
      left: 20px;  
      z-index: 2; /* Caixa 2 tem z-index 2 */  
    }  
  
    .caixa3 {  
      width: 100px;  
      height: 100px;  
      background-color: green;
```

```
    position: absolute;
    top: 30px;
    left: 30px;
    z-index: 3; /* Caixa 3 tem z-index 3 */
}
</style>
</head>
<body>
    <div class="caixa1">Caixa 1</div>
    <div class="caixa2">Caixa 2</div>
    <div class="caixa3">Caixa 3</div>
</body>
</html>
```

Neste exemplo, temos três caixas coloridas posicionadas uma sobre a outra. Cada caixa tem uma classe CSS associada e uma propriedade `z-index` que define sua ordem de empilhamento. A `caixa1` tem o menor `z-index`, a `caixa2` está no meio e a `caixa3` tem o maior `z-index`. Isso determina a ordem em que as caixas são empilhadas, com a `caixa3` aparecendo no topo das outras duas, devido ao seu `z-index` mais alto.

O uso de `z-index` é útil quando você precisa controlar a sobreposição de elementos posicionados e pode ser especialmente útil em layouts complexos ou na criação de elementos de sobreposição, como menus pop-up ou caixas de diálogo.

**19. `opacity`:** Ajusta a opacidade de um elemento, variando de 0 (totalmente transparente) a 1 (totalmente opaco).

```
.transparencia {
    opacity: 0.5; /* Define a opacidade como 50% (semi-transparente) */
}
```

Neste exemplo, a classe `.transparencia` é usada para definir a opacidade de um elemento. Com `opacity: 0.5;`, o elemento será exibido com 50% de opacidade, tornando-o semi-transparente. Você pode ajustar o valor de `opacity` de 0 (totalmente transparente) a 1 (totalmente opaco) para criar efeitos de transparência em elementos HTML.

**20. `transform`:** Permite aplicar transformações 2D e 3D a elementos, como rotação, escala e translação.

```
.elemento-transform {
    transform: rotate(45deg); /* Rotação de 45 graus */
}
```

Neste exemplo, a classe `.elemento-transform` aplica uma transformação de rotação de 45 graus ao elemento. Você pode usar transformações para realizar rotações, escalas, translações e muito mais.

**21. `transition`**: Controla transições suaves de propriedades, como cores e tamanhos, durante um período de tempo.

```
.botao-transicao {  
  background-color: #FF0000;  
  color: #FFFFFF;  
  transition: background-color 0.5s ease, color 0.5s ease; /* Transição suave de cores em 0,5 segundos */  
}  
  
.botao-transicao:hover {  
  background-color: #00FF00;  
  color: #000000;  
}
```

Neste exemplo, quando você passa o cursor sobre o botão com a classe `.botao-transicao`, a transição suave é ativada, fazendo com que a cor de fundo e o texto mudem gradualmente para as novas cores definidas em 0,5 segundos. O `:hover` indica que a transição ocorrerá ao passar o cursor sobre o botão.

**22. `animation`**: Permite criar animações personalizadas com etapas e durações definidas.

```
@keyframes girar {  
  0% { transform: rotate(0deg); }  
  100% { transform: rotate(360deg); }  
}  
  
.elemento-animacao {  
  animation: girar 4s linear infinite; /* Animação de rotação infinita em 4 segundos */  
}
```

Neste exemplo, uma animação personalizada chamada `girar` é definida usando `@keyframes`. Em seguida, a classe `.elemento-animacao` aplica essa animação de rotação infinita ao elemento, levando 4 segundos para fazer uma rotação completa de 360 graus.

Esses exemplos demonstram como essas propriedades do CSS podem ser usadas para estilizar elementos em uma página web. Cada propriedade oferece controle sobre diferentes aspectos do design e layout, permitindo que você crie páginas web personalizadas e visualmente atraentes.