Conteúdo

***Módulo 7:***

*Propriedade Display*

*Propriedade Max-Width*

*Propriedade Position*

*Propriedade Overflow*

*Propriedade Float*

*Propriedade Opacity*

*Propriedade Text Shadow e Box Shadow*

*Efeito Dropdown*

*Efeito Gradiente*

*Efeitos de Texto*

*Especificidade CSS*

*Regra Important*

***Propriedade Display***

*A propriedade display no CSS é uma das mais fundamentais e poderosas, utilizada para controlar o layout dos elementos na página. Ela define se um elemento deve ser exibido ou não e como deve ser organizado junto aos outros elementos. Aqui estão alguns valores comuns para a propriedade display e exemplos de como eles são usados:*

*1.* ***block***

*Faz com que o elemento ocupe toda a largura disponível, independentemente de sua largura real. Elementos* ***block*** *sempre começam em uma nova linha.*

*Exemplo:* ***div, p, e section são por padrão elementos block.***

***2. inline***

*Os elementos não começam em uma nova linha e apenas ocupam a largura necessária. É útil para elementos dentro de um parágrafo ou frase, como span ou a.*

***3. inline-block***

*Combina aspectos de inline e block. O elemento não começa em uma nova linha, mas você pode definir a largura e a altura, o que não é possível com elementos puramente inline.*

***4. none***

*O elemento é completamente removido do fluxo do documento. Ele não ocupa espaço e é como se não estivesse no HTML.*

***5. flex***

*Permite que você aplique um layout flexível aos seus elementos. É um modelo de layout projetado para estruturas complexas e alinhamentos precisos dentro de um container.*

***6. grid***

*Facilita o trabalho com layouts bidimensionais (linhas e colunas). grid oferece um controle mais fino sobre como os elementos são posicionados e dimensionados em relação um ao outro.*

*Cada um desses valores permite diferentes maneiras de organizar e apresentar os elementos de uma página, tornando a propriedade `display` essencial para o design e layout web.*

*Exemplos:*

*https://codepen.io/uchoamaster/pen/qBwYJXd*

*Neste exemplo, você pode ver como diferentes elementos são afetados pela propriedade **`display`**:*

- ***Block**: Ocupa toda a largura disponível, independentemente de seu conteúdo, e sempre começa em uma nova linha.*
- ***Inline**: Não inicia uma nova linha e ocupa apenas o espaço necessário para seu conteúdo.*
- ***Inline-Block**: Semelhante ao **`inline`** em não começar uma nova linha, mas permite a definição de largura e altura.*
- ***None**: O elemento não é exibido e não ocupa espaço no layout.*
- ***Flex**: O container flexível ajusta os itens filhos de acordo com as propriedades **`flex`**.*
- ***Grid**: O container de grade organiza os itens filhos em um layout de grade bidimensional.*

*Espero que este exemplo ajude a ilustrar como cada propriedade **`display`** funciona!*

### *Propriedade Max-Width*

*A propriedade `max-width` no CSS é usada para definir a largura máxima de um elemento. Isso significa que o elemento pode ser menor que o valor definido, mas nunca maior. Essa propriedade é particularmente útil para criar designs responsivos, pois permite que um elemento se ajuste ao tamanho da tela ou do contêiner pai, mantendo uma largura máxima para evitar que o layout se quebre em telas grandes.*

*Aqui está um exemplo prático de como `max-width` pode ser usado para controlar o layout de um site:*

*https://codepen.io/uchoamaster/pen/vYMjVjx*

*Neste exemplo, o `.container` tem uma `max-width` de 600px, o que significa que seu conteúdo pode ocupar até 600px de largura, mas não mais que isso. Se a tela do dispositivo for maior que 600px, o contêiner permanecerá com 600px de largura e será centralizado horizontalmente devido ao `margin: 0 auto;`. Em telas menores que*

*600px, o contêiner se ajustará à largura da tela, mantendo as margens e o preenchimento definidos, garantindo que o conteúdo permaneça legível e acessível, independentemente do tamanho da tela.*

*Essa propriedade é uma ferramenta valiosa para criar designs responsivos, pois ajuda a manter o layout flexível e adaptável a diferentes tamanhos de tela, melhorando a experiência do usuário em dispositivos móveis e desktops.*

***Propriedade Position***

*A propriedade `position` no CSS é utilizada para especificar como um elemento é posicionado na página. Ela permite que você posicione elementos fora do fluxo normal do documento, o que pode ser bastante útil para criar layouts específicos ou efeitos visuais, como pop-ups, sticky headers, ou elementos sobrepostos. Existem vários valores que `position` pode assumir:*

## *Valores da Propriedade `position`*

- *`static`: Este é o valor padrão. Os elementos são posicionados de acordo com o fluxo normal do documento.*
- *`relative`: O elemento é posicionado em relação à sua posição original sem ser removido do fluxo do documento. Você pode usar `top`, `right`, `bottom`, e `left` para movê-lo.*
- *`absolute`: O elemento é removido do fluxo do documento e posicionado em relação ao seu contêiner posicionado mais próximo (não `static`). Ele pode ser movido com `top`, `right`, `bottom`, e `left`.*
- *`fixed`: O elemento é removido do fluxo do documento e posicionado em relação à viewport. Ele permanece fixo ao rolar a página.*
- *`sticky`: Uma mistura de `relative` e `fixed`. O elemento é tratado como `relative` até que a página seja rolada a um ponto específico, momento em que ele se torna `fixed`.*

## *Exemplo Prático*

*Vamos criar um exemplo que mostra alguns desses comportamentos:*

*https://codepen.io/uchoamaster/pen/yLrjREZ*

*Neste exemplo, você verá um elemento com `position: relative` movido em relação à sua posição normal, um elemento `absolute` posicionado em relação ao `<body>` (já que não há outro contêiner posicionado mais próximo), um elemento `fixed` que fica*

*fixo na tela enquanto você rola, e um* ***`sticky`****, que adere ao topo da viewport ao rolar para baixo.*

*Esse exemplo ilustra como a propriedade* ***`position`*** *pode ser utilizada para controlar o layout e o posicionamento dos elementos na sua página web, oferecendo uma grande flexibilidade para design de interfaces e interações visuais.*

***Propriedade Overflow***

*A propriedade* ***`overflow`*** *no CSS determina o que acontece com o conteúdo que excede os limites de um elemento. Se um elemento contém mais conteúdo do que pode caber em sua área especificada,* ***`overflow`*** *permite controlar se esse conteúdo extra deve ser recortado, exibido com barras de rolagem ou se deve ser exibido fora do elemento. É uma propriedade essencial para gerenciar o layout de conteúdos que variam em tamanho, como textos dinâmicos ou imagens.*

## *Valores da Propriedade* ***`overflow`***

- ***`visible`****: Valor padrão. O conteúdo que excede o tamanho do elemento é mostrado fora dele.*
- ***`hidden`****: O conteúdo que excede o tamanho do elemento é recortado e não visível.*
- ***`scroll`****: Barras de rolagem são adicionadas ao elemento para permitir a visualização do conteúdo oculto, independentemente de haver conteúdo excedente ou não.*
- ***`auto`****: O navegador decide se deve adicionar barras de rolagem para visualizar o conteúdo excedente.*

*A propriedade* ***`overflow`*** *pode ser especificada para ambos os eixos (horizontal e vertical) usando* ***`overflow-x`*** *e* ***`overflow-y`****, respectivamente, permitindo um controle mais refinado.*

## *Exemplo Prático*

*Aqui está um exemplo prático que demonstra o uso da propriedade* `overflow`*:*

*https://codepen.io/uchoamaster/pen/LYvmgvb*

*Neste exemplo, criamos quatro contêineres com a mesma dimensão (****`width: 200px; height: 100px;`****) mas com diferentes valores para a propriedade* ***`overflow`****. Dentro de cada contêiner, há um parágrafo (*`<p>`*) que é mais largo que o contêiner,*

*demonstrando como cada valor de `overflow` afeta a visualização do conteúdo excedente.*

- ***Visible**: O conteúdo que ultrapassa os limites é mostrado fora do contêiner.*
- ***Hidden**: O conteúdo que não cabe é ocultado.*
- ***Scroll**: Barras de rolagem são adicionadas, permitindo a visualização de todo o conteúdo.*
- ***Auto**: Barras de rolagem aparecem apenas se necessário, baseando-se na necessidade de exibir o conteúdo excedente.*

*Este exemplo ilustra claramente como a propriedade `overflow` pode ser utilizada para controlar a apresentação de conteúdos que excedem o espaço de um elemento.*

***Propriedade Float***

*A propriedade `float` no CSS é utilizada para posicionar elementos ao longo do eixo horizontal de seu contêiner pai, permitindo que os elementos de texto e outros elementos em linha fluam ao redor deles. Originalmente, `float` foi projetado para o layout de imagens e texto em jornais e revistas, permitindo que o texto envolvesse imagens ou caixas de informação de forma harmoniosa.*

## *Valores da Propriedade `float`*

- *`left`: O elemento é deslocado para a esquerda do contêiner, e o texto e os elementos em linha fluem ao redor do seu lado direito.*
- *`right`: O elemento é deslocado para a direita do contêiner, e o texto e os elementos em linha fluem ao redor do seu lado esquerdo.*
- *`none`: Valor padrão. O elemento não flutua e é exibido em seu fluxo normal na página.*
- *`inherit`: O elemento herda o valor da propriedade `float` do seu elemento pai.*

*A propriedade `float` pode causar um comportamento indesejado quando elementos subsequentes não são adequadamente "limpos" ou "clear". Isso acontece porque o elemento flutuante é removido do fluxo normal do documento, afetando como os elementos seguintes são posicionados. Para evitar isso, é comum usar a propriedade `clear` em elementos subsequentes, que impede que eles sejam dispostos ao lado do elemento flutuante.*

## *Exemplo Prático*

*Aqui está um exemplo que demonstra o uso de todas as propriedades de `float`, incluindo uma demonstração da propriedade `clear`:*

https://codepen.io/uchoamaster/pen/poBVQeo

*Neste exemplo, `float: left;` e `float: right;` são usados para alinhar os elementos à esquerda e à direita, respectivamente, permitindo que o texto flua ao redor deles. O elemento com `float: none;` não flutua e aparece em seu fluxo normal no documento. Por fim, o elemento com `clear: both;` é posicionado abaixo de todos os elementos flutuantes, independentemente do lado em que flutuam, demonstrando como prevenir que o texto e outros elementos fluam ao lado de elementos flutuantes.*

***Propriedade Opacity***

*A propriedade `opacity` no CSS é usada para definir o nível de transparência de um elemento, incluindo todo o seu conteúdo (texto, imagens e filhos). O valor da propriedade `opacity` é um número entre 0.0 (completamente transparente) e 1.0 (completamente opaco). Esse recurso é útil para criar efeitos de sobreposição, destacar ou atenuar elementos na interface do usuário, e para efeitos visuais em elementos gráficos.*

## *Sintaxe*

```
elemento {
  opacity: valor; /* Valor pode ser de 0.0 a 1.0 */
}
```

## *Exemplo Prático*

*Vamos criar um exemplo prático com a propriedade `opacity`, demonstrando diferentes níveis de transparência:*

https://codepen.io/uchoamaster/pen/eYorQav

*Neste exemplo, criamos três divs com a classe `.box` e diferentes classes para os níveis de opacidade: `.opaque` (opaco), `.semi-transparent` (semi-transparente), e `.transparent` (transparente). Cada `.box` tem um fundo azul, mas a opacidade varia, demonstrando o efeito visual que você pode alcançar ajustando o valor de `opacity`.*

- ***Opaco** (1.0): O elemento é completamente opaco e o fundo é sólido.*
- ***Semi-Transparente** (0.5): O elemento é 50% transparente, permitindo que o fundo por trás dele seja parcialmente visível.*

- ***Transparente*** *(0.1): O elemento é altamente transparente, tornando o fundo quase completamente visível através dele.*

*A propriedade* `opacity` *é uma ferramenta poderosa para designers e desenvolvedores web, permitindo criar uma ampla gama de efeitos visuais e interações na interface do usuário.*

***Propriedade Text Shadow e Box Shadow***

*As propriedades* `text-shadow` *e* `box-shadow` *no CSS permitem adicionar sombras a textos e caixas (elementos), respectivamente, criando profundidade, contraste, e interessantes efeitos visuais em uma página web. Vamos explorar cada uma delas com exemplos práticos.*

## *text-shadow*

*A propriedade* `text-shadow` *aplica sombra ao texto de um elemento. Ela pode receber vários valores para definir o deslocamento horizontal e vertical da sombra, o raio de desfoque e a cor da sombra.*

*https://codepen.io/uchoamaster/pen/YzMLdPd*

*No exemplo acima, o texto "Texto com Sombra" terá uma sombra preta com ligeiro desfoque, deslocada 4 pixels para a direita e 4 pixels para baixo do texto.*

## *box-shadow*

*A propriedade* `box-shadow` *aplica sombra ao elemento. Ela pode aceitar vários valores para definir o deslocamento horizontal e vertical da sombra, o raio de desfoque, o espalhamento da sombra e a cor.*

*https://codepen.io/uchoamaster/pen/ExJLGjo*

*Neste exemplo, a caixa terá uma sombra preta suave em todos os lados, criando a impressão de que a caixa está elevada em relação ao fundo.*

*Ambas as propriedades,* `text-shadow` *e* `box-shadow`*, podem ser utilizadas para adicionar profundidade visual aos elementos de uma página web, melhorando a estética ou destacando certos componentes do design. Experimentar com diferentes valores e cores pode levar a resultados visualmente atraentes.*

***Efeito Dropdown***

*Um efeito dropdown é um componente comum em interfaces de usuário, onde um menu ou uma lista de opções é revelado sob um elemento ao interagir com ele (geralmente, ao passar o mouse por cima ou ao clicar). Vamos criar um exemplo simples de um menu dropdown usando apenas HTML e CSS, aproveitando os seletores de hover para mostrar e esconder o conteúdo do dropdown. Vamos ver como isso pode ser feito:*

*https://codepen.io/uchoamaster/pen/eYorbJE*

*Neste exemplo, o menu dropdown é inicialmente escondido (`display: none;`) e só se torna visível (`display: block;`) quando o usuário passa o mouse sobre o elemento `.dropdown`, graças ao seletor de hover `.dropdown:hover .dropdown-content`. Isso cria um efeito de dropdown puramente com CSS, sem necessidade de JavaScript.*

*Esta abordagem é bastante simples e funciona bem para menus dropdown que não requerem interatividade complexa. Por ser uma solução que depende do hover, pode não ser a mais adequada para todos os dispositivos, especialmente aqueles com telas sensíveis ao toque, onde o conceito de "hover" não se aplica da mesma maneira que em dispositivos com cursor.*

***Efeito Gradiente***

*Os gradientes no CSS são uma maneira de criar transições suaves entre duas ou mais cores especificadas. Eles podem ser usados em diversos elementos, como fundos, textos (com propriedades avançadas ou máscaras) e bordas. Os gradientes podem ser lineares ou radiais.*

## *Gradiente Linear*

*Um gradiente linear transiciona as cores ao longo de uma linha reta. Você pode controlar a direção dessa linha (topo para baixo, esquerda para direita, diagonais, etc.) e as cores usadas.*

*Neste exemplo, o gradiente linear move-se da esquerda (`#ff00ff`) para a direita (`#feb47b`), criando uma transição suave entre as duas cores.*

## *Gradiente Radial*

*Um gradiente radial transiciona as cores a partir de um ponto central, criando um efeito que se assemelha a um círculo que se expande.*

*Neste exemplo, o gradiente radial cria um efeito de círculo que começa com a cor* `#ff7e5f` *e transiciona suavemente para a cor* `#feb47b`.

## Usando Gradientes em Textos

*Para aplicar um gradiente a um texto, podemos usar a propriedade* `background-clip` *com* `text` *e uma cor de texto transparente.*

https://codepen.io/uchoamaster/pen/OJGZrmV

*Este exemplo aplica um gradiente ao texto, fazendo com que ele tenha uma transição de cores. Note que usamos* `-webkit-background-clip: text;` *e* `color: transparent;` *para garantir que o gradiente preencha apenas o texto e não o fundo ao redor dele.*

*Os gradientes são uma ferramenta poderosa para adicionar visualizações ricas e vibrantes a um design web, e podem ser ajustados para se adequar a qualquer necessidade visual. Experimente diferentes cores e direções para criar efeitos visuais únicos.*

### Efeitos de Texto

*Os efeitos de texto no CSS permitem adicionar visuais interessantes e dinâmicos ao conteúdo textual de uma página web. Vou mostrar alguns exemplos práticos de efeitos de texto, como sombra, gradiente e animação, que podem ser aplicados para destacar certas partes do seu site.*

## 1. Sombra no Texto (`text-shadow`)

*A propriedade* `text-shadow` *adiciona sombra ao texto, podendo criar profundidade ou um efeito de brilho.*

https://codepen.io/uchoamaster/pen/yLrjGjP

*Este exemplo aplica uma sombra preta simples ao texto, dando um efeito de elevação.*

## 2. Texto com Gradiente

*Usando `background-clip` e `text-fill-color`, você pode aplicar um gradiente como cor de texto. Isso requer um pouco de CSS extra para compatibilidade com diferentes navegadores.*

*https://codepen.io/uchoamaster/pen/NWmMezd*

*Este exemplo cria um efeito de gradiente no texto que transita do claro para o escuro.*

## *3. Texto Animado*

*Animações podem ser aplicadas ao texto para criar efeitos visuais dinâmicos, como mudança de cor ou pulsação.*

*https://codepen.io/uchoamaster/pen/rNbvoKv*

*Este código faz com que a cor do texto alterne entre preto e vermelho, criando um efeito pulsante.*

## *4. Texto com Efeito de Máquina de Escrever*

*Este efeito simula o texto sendo digitado na tela, como uma máquina de escrever.*

*https://codepen.io/uchoamaster/pen/gOyzZja*

*Neste exemplo, o efeito de "máquina de escrever" é criado animando a propriedade `width` do texto de 0 a 100% e usando uma animação de "piscar" para o cursor.*

*Esses exemplos demonstram algumas das muitas possibilidades criativas que o CSS oferece para enriquecer a apresentação do texto em páginas web. Experimente e ajuste os valores para se adequar ao estilo e necessidades específicas do seu projeto.*

***Especificidade CSS***

*A especificidade no CSS é uma das regras fundamentais que determinam como os conflitos de estilo são resolvidos quando mais de uma regra pode ser aplicada a um elemento. Basicamente, é um sistema de pontuação que atribui um peso a cada tipo de seletor, e o estilo com a maior pontuação de especificidade é o que prevalece.*

*A pontuação de especificidade é calculada com base em quatro categorias, ordenadas da menos específica para a mais específica:*

*Estilos de Nível de Tipo (por exemplo, seletores de tags como `div`, `p`): 0,0,0,1*

*Estilos de Classe, pseudoclasses e atributos (por exemplo, `.classe`, `:hover`, `[atributo="valor"]`): 0,0,1,0*
*Estilos de ID (por exemplo, `#id`): 0,1,0,0*
*Estilos Inline (definidos diretamente no atributo `style` de um elemento HTML): 1,0,0,0*

*Estilos aplicados diretamente dentro da tag HTML (inline) têm a maior especificidade, enquanto seletores de tipo têm a menor. Importante destacar que a especificidade é uma razão comum para o CSS não se aplicar como esperado, então entender como ela funciona é crucial para o design eficaz de páginas web.*

## *Projeto Prático: Demonstrando Especificidade CSS*

*Vamos criar um exemplo prático para demonstrar como a especificidade afeta a aplicação dos estilos.*

*https://codepen.io/uchoamaster/pen/KKYRbby*

*Neste exemplo, temos três regras CSS aplicadas a um mesmo elemento `div`. Cada regra tem um nível diferente de especificidade:*

*O seletor de tipo (**`div`**) tem a especificidade mais baixa (0,0,0,1).*
*O seletor de classe (**`.box`**) tem uma especificidade média (0,0,1,0).*
*O seletor de ID (**`#uniqueBox`**) tem a especificidade mais alta entre os seletores de CSS declarados externamente (0,1,0,0).*

*Se você aplicasse um estilo inline ao elemento (como **`style="background-color: violet;"`**), esse estilo teria a maior especificidade de todos (1,0,0,0) e sobrescreveria os demais.*

*Dada essa configuração, a cor de fundo do **`div`** será **`tomato`**, pois o seletor de ID tem a especificidade mais alta entre os declarados no arquivo CSS. Se você remover o estilo de ID, o próximo mais específico, que é o estilo de classe **`.box`** com fundo **`lightgreen`**, será aplicado.*

***Entender a especificidade é crucial para manipular corretamente os estilos em CSS, especialmente em documentos complexos onde múltiplos estilos podem ser aplicados aos mesmos elementos. Experimentar com diferentes seletores e regras pode ajudar a solidificar seu entendimento de como a especificidade afeta o resultado final do estilo aplicado.***

### *Regra Important*

*A regra `!important` no CSS é uma forma de aumentar a especificidade de uma declaração CSS, forçando-a a sobrescrever outras declarações, mesmo que elas tenham uma especificidade normalmente maior. Deve ser usada com moderação, pois seu uso excessivo pode tornar a manutenção do código mais difícil, criando uma cascata de estilos difícil de ser sobreposta e compreendida.*

*Aqui está um exemplo prático de como `!important` pode ser utilizado:*

*https://codepen.io/uchoamaster/pen/poBVYaJ*

*Neste exemplo, mesmo o segundo parágrafo tendo um ID, que normalmente teria uma especificidade maior do que uma classe, o uso de `!important` na declaração da cor vermelha para a classe `.text` faz com que esse estilo tenha prioridade sobre o estilo verde definido pelo ID `#important-text`. Portanto, ambos os textos serão coloridos de vermelho.*

## *Quando usar `!important`?*

*O uso de `!important` é geralmente desaconselhado a não ser que seja absolutamente necessário, pois pode tornar o código CSS difícil de manter e depurar. Alguns casos em que seu uso pode ser justificado incluem:*

- ***Sobrescrever estilos** de folhas de estilo de terceiros ou widgets que você não pode controlar diretamente.*
- ***Aplicar estilos** que devem sempre ter prioridade, independentemente de outros estilos adicionados posteriormente no desenvolvimento do projeto.*
- ***Estilos de tema** ou **acessibilidade** que precisam garantir a aplicação sob qualquer circunstância.*

*Lembre-se de que um bom planejamento e organização do CSS podem, na maioria dos casos, eliminar a necessidade do `!important`, fazendo com que seu uso seja a exceção, não a regra.*

*Agora vamos criar um leiaute para treinar tudo que aprendemos nessa OT, segue abaixo o código e a explicação passo a passo.*

*Vamos criar uma tela de login simples que incorpora vários dos conceitos discutidos hoje, incluindo gradientes, efeitos de texto, e especificidade CSS. A tela terá um formulário de login posicionado sobre um fundo com imagem e gradiente, criando um visual atraente e moderno.*

https://codepen.io/uchoamaster/pen/poBVqBa

*Neste projeto prático, você tem uma tela de login com estilo moderno que utiliza gradientes para adicionar profundidade visual ao fundo, sombra de texto para destacar o título e especificidade CSS para aplicar estilos de forma eficaz ao formulário e seus elementos. A imagem de fundo e a camada de gradiente sobreposta melhoram a estética e focam a atenção no formulário de login.*

*Lembre-se de substituir* `'login-background.jpg'` *no CSS pelo caminho correto da imagem de fundo que você deseja usar. Essa estrutura básica pode ser expandida e personalizada de acordo com as necessidades específicas do seu projeto, adicionando, por exemplo, mensagens de erro ou links para recuperação de senha e registro.*