Introdução ao CSS

Neste capítulo, vamos explorar o mundo do CSS (Cascading Style Sheets), uma linguagem de estilo que permite a você controlar a apresentação e o design de suas páginas web. Você aprenderá o que é CSS, por que é importante e como separar a apresentação do conteúdo.

O que é CSS?

O que é CSS?

O **CSS**, ou **Cascading Style Sheets** (Folhas de Estilo em Cascata), é uma linguagem de estilo usada para controlar a apresentação visual de elementos em documentos HTML. Ele desempenha um papel crucial no design e na formatação de páginas web. Vamos aprimorar a explicação sobre o CSS:

Separação de Conteúdo e Estilo

O CSS foi desenvolvido para separar a estrutura (HTML) do design (estilo) de uma página web. Isso oferece várias vantagens:

1. **Manutenção mais fácil**: Alterações no estilo podem ser feitas em um único local, afetando todas as ocorrências dos elementos afetados, em vez de ter que editar cada elemento individualmente.
2. **Reutilização**: Você pode criar regras de estilo que se aplicam a vários elementos, economizando tempo e esforço.
3. **Acessibilidade**: A separação de estilo facilita a criação de páginas acessíveis, pois permite que os leitores de tela e outros dispositivos interpretem o conteúdo de maneira mais eficaz.

Como o CSS Funciona

O CSS funciona por meio de **seletores** e **declarações**. Um seletor identifica um elemento HTML específico, enquanto as declarações definem como esse elemento deve ser estilizado. Aqui está um exemplo simples:

```
/* Um seletor que estiliza todos os parágrafos com a classe "destaque" */
p.destaque {
    color: #FF0000; /* Define a cor do texto como vermelha */
    font-size: 16px; /* Define o tamanho da fonte como 16 pixels */
}
```

Neste exemplo, `p.destaque` é o seletor que segmenta todos os parágrafos com a classe "destaque." As declarações dentro das chaves `{}` definem a cor do texto e o tamanho da fonte para esses parágrafos.

Propriedades e Valores

As declarações CSS consistem em **propriedades** e **valores**. As propriedades especificam qual aspecto do elemento deve ser estilizado, como cor, tamanho, margem, etc. Os valores definem os detalhes específicos da propriedade. Por exemplo:

- **Propriedade**: `color`
- **Valor**: `#FF0000` (uma cor vermelha)

Existem centenas de propriedades CSS que podem ser usadas para estilizar elementos, incluindo fonte, espaçamento, bordas, sombras, animações e muito mais.

Cascata e Herança

O termo "em cascata" no CSS refere-se à maneira como as regras de estilo são aplicadas e priorizadas. As regras podem ser definidas em diferentes lugares, como folhas de estilo externas, na seção `<style>` do HTML ou diretamente em um atributo `style` de um elemento. O CSS segue uma ordem de prioridade para determinar qual regra será aplicada quando várias regras se sobrepõem.

Além disso, o CSS herda propriedades de elementos pai para elementos filhos. Isso significa que, se você estilizar um elemento pai, seus filhos podem herdar algumas ou todas as propriedades de estilo.

Evolução do CSS

O CSS evoluiu ao longo do tempo, com a versão mais recente sendo o CSS3. O CSS3 trouxe recursos avançados, como sombras, gradientes, transições, animações e suporte a mídia responsiva, permitindo designs web mais sofisticados e interativos.

Em resumo, o CSS é essencial para o design web moderno, permitindo que os desenvolvedores controlem a aparência e a formatação de páginas HTML. É uma linguagem poderosa e versátil que desempenha um papel fundamental na criação de experiências visuais atraentes na web.

A Separação de Conteúdo e Apresentação

A Separação de Conteúdo e Apresentação

Uma das principais vantagens do CSS é a separação de conteúdo e apresentação. Isso significa que você pode manter o conteúdo (HTML) separado do estilo (CSS), o que torna o código mais organizado e mais fácil de gerenciar. A separação também permite que você faça alterações de estilo em todo o site sem precisar modificar cada página individualmente.

Estilos Inline, Internos e Externos

Estilos Inline, Internos e Externos

Existem três maneiras principais de incluir CSS em uma página:

Estilos Inline: Você pode adicionar estilos diretamente em elementos HTML usando o atributo `style`. Por exemplo:

```
<p style="color: blue; font-size: 16px;">Este é um parágrafo azul com tamanho de fonte 16px.</p>
```

Estilos Internos: Você pode incluir estilos dentro da tag `<style>` na seção `<head>` de um documento HTML. Isso afeta todas as tags dentro da página. Por exemplo:

```
<head>
  <style>
    p {
      color: blue;
      font-size: 16px;
    }
  </style>
</head>
```

Estilos Externos: É uma prática recomendada criar arquivos CSS separados e vinculá-los a várias páginas HTML. Isso mantém o código limpo e reutilizável. Por exemplo:

```
<head>
  <link rel="stylesheet" type="text/css" href="estilos.css">
</head>
```

Sintaxe CSS

Seletores, Propriedades e Valores

CSS segue uma sintaxe específica. Aqui está um exemplo simples de uma regra CSS:

```
seletor {
    propriedade: valor;
}
```

- **Seletor**: Indica a quem a regra se aplica. Pode ser uma tag HTML, uma classe, um ID ou outros seletores avançados.
- **Propriedade**: Define o que será estilizado, como `color` para a cor do texto ou `font-size` para o tamanho da fonte.
- **Valor**: Especifica o valor da propriedade, como `blue` para a cor ou `16px` para o tamanho da fonte.

No próximo capítulo, exploraremos seletores, propriedades e valores com mais detalhes, mostrando como estilizar efetivamente elementos HTML.

Principais propriedades do CSS - 1

Principais propriedades do CSS

O CSS (Cascading Style Sheets) oferece uma ampla variedade de propriedades que permitem controlar o estilo e a apresentação dos elementos HTML em uma página web. Abaixo, estão algumas das principais propriedades CSS e suas descrições:

1. **`color`**: Define a cor do texto dentro de um elemento.

```
p {

}
```

2. **`font-family`**: Especifica a família de fontes a ser usada para o texto.

```
body {
    font-family: Arial, sans-serif; /* Fontes
}
```

3. **`font-size`**: Controla o tamanho da fonte do texto.

```
h2 {
    font-size: 24px; /* Tamanho da fonte de 24
}
```

4. **`font-weight`**: Define a espessura da fonte, como "normal," "negrito" ou valores numéricos.

```
strong {
    font-weight: bold; /* Texto
}
```

5. **`text-align`**: Alinha o texto horizontalmente (esquerda, direita, centralizado, justificado).

```
div {
    text-align: center; /* Alinha o texto ao
}
```

6. **`text-decoration`**: Controla a decoração do texto, como sublinhado ou tachado.

```
a {
    text-decoration: underline; /* S
}
```

7. **line-height**: Define a altura da linha, afetando o espaço entre as linhas de texto.

```
p {
    line-height: 1.5; /* Espaço entre linhas 1,5 vezes a altura da fonte
}
```

8. **background-color**: Especifica a cor de fundo de um elemento.

```
.destaque {
    background-color: #FFFF00; /* Fundo amarelo para elementos destacados
}
```

9. **border**: Controla as propriedades da borda de um elemento, como largura, estilo e cor.

```
.caixa {
    border: 2px dashed #000; /* Borda de 2 pixels, estilo tracejado, cor preta
}
```

10. **margin**: Define a margem externa de um elemento, afetando o espaço ao seu redor.

```
.margem-superior {
    margin-top: 20px; /* Margem superior de 20
}
```

11. **padding**: Controla o preenchimento interno de um elemento, afetando o espaço entre seu conteúdo e sua borda.

```
.preenchimento-interno {
    padding: 10px; /* Preenchimento de 10 pixels em todas as direções
}
```

12. **width** e **height**: Estabelecem a largura e altura de um elemento.

```
.caixa {


}
```

13. **display**: Define como um elemento é exibido, como "block" (bloco) ou "inline" (em linha).

```
.botao {
    display: inline-block; /* Exibe o elemento como um bloco em linha
}
```

14. **position**: Controla a posição de um elemento na página, com opções como "relative" (relativa) ou "absolute (absoluta).

```
.elemento-absoluto {




}
```

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.

Exemplo de página com CSS

Aqui está um exemplo de uma página HTML que demonstra o uso de estilos inline, interno (no cabeçalho) e externo (com um link para um arquivo CSS). A página também exemplifica o uso de `id` e `class` para aplicar estilos.

Exemplo de arquivo HTML com estilos inline, internos e externos:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de Estilos</title>

    <!-- Estilo interno -->
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
        }

        .titulo {
            color: #4CAF50;
            text-align: center;
        }

        .paragrafo {
            font-size: 18px;
            line-height: 1.6;
            margin: 10px 20px;
        }

        #destaque {
            color: #ff5733;
            font-weight: bold;
```

```
        }
    </style>

    <!-- Link para estilo externo -->
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <!-- Estilo inline aplicado diretamente -->
    <h1 style="color: blue; font-size: 2em;">Exemplo de Estilos em HTML</h1>

    <h2 class="titulo">Estilos Externos, Internos e Inline</h2>

    <p class="paragrafo">Este é um exemplo de <strong>parágrafo</strong> com estilos aplicados através de uma classe chamada <em>paragrafo</em>.</p>

    <p id="destaque" class="paragrafo">Este parágrafo tem um <em>id</em> chamado <strong>destaque</strong>, além da classe <strong>paragrafo</strong>.</p>

    <p style="color: green;">Este parágrafo tem um estilo <strong>inline</strong> aplicado diretamente no elemento, alterando a cor para verde.</p>
</body>
</html>
```

Exemplo de arquivo CSS externo (estilos.css):

```
/* Estilo aplicado via arquivo CSS externo */
body {
    margin: 0;
    padding: 0;
}

h1 {
    background-color: lightgrey;
    padding: 20px;
    text-align: center;
}

p {
    font-family: 'Verdana', sans-serif;
}
```

- **Inline**: O estilo é aplicado diretamente no elemento usando o atributo `style`, como no `<h1>` e no último `<p>`.
- **Interno**: O estilo é definido dentro da tag `<style>` no cabeçalho do HTML, como as regras para `.titulo`, `.paragrafo` e `#destaque`.
- **Externo**: O estilo está em um arquivo CSS separado (`estilos.css`), que é vinculado à página HTML através da tag `<link>`.
- **`id` e `class`**: O `id` (único por página) foi aplicado ao parágrafo com o texto em destaque, enquanto a `class` foi usada para estilizar múltiplos parágrafos.

Esse exemplo mostra as três formas de aplicar estilos e como utilizar `id` e `class`.