Estilos CSS

Tabalhando com seletores

Nesta primeira parte da nossa aula sobre Estilos CSS, vamos explorar o conceito e a aplicação de seletores no desenvolvimento de páginas web. Seletores são ferramentas fundamentais no CSS, pois permitem identificar e aplicar estilos específicos a elementos do HTML. Ao longo deste módulo, vamos aprofundar o conhecimento sobre os diferentes tipos de seletores e como utilizá-los de forma eficiente para criar páginas mais organizadas e visualmente agradáveis.

Introdução aos Seletores

Os seletores são usados para "apontar" quais elementos HTML queremos estilizar. Existem vários tipos de seletores, cada um com suas particularidades e usos específicos. Começaremos pelos seletores básicos, que incluem os seletores de tipo, de classe, de ID e de atributo.

Seletor de Tipo: este seletor aplica estilos a todos os elementos de um determinado tipo. Por exemplo, o seletor `h1` estiliza todos os títulos de nível 1 (`<h1>`) em uma página. Sua sintaxe é simples: basta colocar o nome do elemento e definir os estilos desejados.

Seletor de Classe: utilizado para aplicar estilos a todos os elementos que compartilham uma mesma classe. Para definir um seletor de classe, usamos um

ponto (.) seguido do nome da classe. Isso nos permite aplicar estilos específicos a grupos de elementos, como botões ou seções da página.

Seletor de ID: ideal para aplicar estilos a um elemento único na página, o seletor de ID é identificado por um símbolo de cerquilha (#) seguido do nome do ID. Diferente das classes, um ID deve ser único dentro de uma página HTML, garantindo que os estilos sejam aplicados apenas a um único elemento específico.

Seletor de Atributo: este seletor permite estilizar elementos baseando-se em atributos específicos. Por exemplo, podemos usar `[type="text"]` para aplicar estilos a todos os campos de texto em um formulário. A sintaxe é simples: o nome do atributo entre colchetes, seguido do valor que queremos estilizar.

Seletores Avançados

Além dos seletores básicos, o CSS oferece seletores mais avançados que permitem maior precisão ao aplicar estilos.

Seletor de Filho Direto: este seletor identifica o primeiro nível de descendência de um elemento. Por exemplo, `ul > li` estiliza apenas os itens de lista (`<li>`) que são filhos diretos de uma lista não ordenada (`<ul>`), ignorando elementos mais profundos na hierarquia.

Seletor de Descendentes: diferente do seletor de filho direto, este seletor aplica estilos a todos os descendentes de um elemento, independentemente da profundidade na hierarquia. Um exemplo seria `div p`, que estiliza todos os parágrafos (`<p>`) dentro de uma `div`.

Seletor de Pseudo-Classe: as pseudo-classes aplicam estilos baseados no estado de um elemento, como `a:hover`, que estiliza links (`<a>`) quando o cursor do mouse passa sobre eles. Isso permite criar interações dinâmicas e feedback visual para os usuários.

Seletor de Pseudo-Elemento: os pseudo-elementos, como `p::first-line`, permitem estilizar partes específicas de um elemento, como a primeira linha de um parágrafo. Isso adiciona um nível extra de detalhamento e personalização ao design.

Aplicando Estilos com Seletores

Veja o código de exemplo e o leia sobre as aplicações utilizadas nele:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Estoque</title>
  <style>
    body {
      font-size: 20px;
      font-family: sans-serif;
      margin: 20px;
      background-color: #f4f4f9;
    }
    h2 {
```

```
    color: #333;
}

table {
    border-collapse: collapse;
    margin-bottom: 20px;
}

table, th, td {
    border: 1px solid #ccc;
}

th, td {
    padding: 8px;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}

fieldset {
    max-width: 600px;
    background-color: #fff;
```

```
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
fieldset table {
    width: 100%;
    border: none;
}
fieldset table td {
    padding: 8px;
    border: none;
}
input[type="text"], input[type="number"], input[type="date"] {
    width: 100%;
    padding: 8px;
    margin: 4px 0;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
```

```
}

input[type="submit"], input[type="reset"] {
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

input[type="submit"] {
    background-color: #4CAF50;
}

input[type="reset"] {
    background-color: #df0026;
}

input[type="submit"]:hover {
    background-color: #45a049;
}

input[type="reset"]:hover {
```

```
        background-color: #B80E22;
    }
  </style>
</head>
<body>
  <h2>Tabela de Estoque de Produtos</h2>
  <table>
    <!-- Conteúdo da tabela -->
  </table>
  <fieldset>
    <!-- Conteúdo do fieldset -->
  </fieldset>
</body>
</html>
```

Agora que entendemos os diferentes tipos de seletores, podemos começar a aplicá-los na prática. Por exemplo, podemos definir estilos globais para o corpo da página (`body`), ajustando o tamanho da fonte, a família tipográfica, as margens e a cor de fundo. Podemos, então, refinar ainda mais o design aplicando estilos específicos a títulos (`h2`), tabelas, células de cabeçalho e dados, e até mesmo aos botões de envio e reset de um formulário.

Um aspecto importante ao trabalhar com seletores é entender como combinar múltiplos seletores para aplicar estilos de forma eficiente. Por exemplo, usando u vírgula, podemos aplicar o mesmo estilo a vários elementos de uma só vez, como `table, th, td { border: 1px solid #CCC; }`, que define uma borda uniforme para tabelas, células de cabeçalho e células de dados. Além disso, podemos especificar propriedades como `padding` e `text-align` para garantir que o conteúdo das tabelas esteja bem organizado e legível.

Podemos também utilizar seletores de pseudo-classes para adicionar interações dinâmicas. Por exemplo, ao passar o mouse sobre os botões de envio e reset, podemos mudar a cor de fundo desses botões usando `:hover`, o que destaca visualmente a interação do usuário, tornando a experiência mais intuitiva e agradável. Veja a visualização da nossa tabela após essa aplicação:

Figura 1 - Tabela de Estoque de Produtos

Fonte: elaborado pelo autor

Nesta primeira parte da aula, exploramos os conceitos fundamentais dos seletores CSS e como eles são usados para estilizar elementos HTML. Seletores são poderosas ferramentas que, quando bem utilizadas, permitem criar layouts sofisticados e interfaces de usuário interativas e visualmente atraentes. Na próxima parte da nossa aula, continuaremos explorando outros aspectos do CSS, como alinhamento de fontes, layout e posicionamento de elementos.

Estilos e alinhamentos de fontes

Nesta segunda parte da nossa aula, focaremos nos estilos e no alinhamento de fontes, elementos essenciais para garantir uma boa apresentação visual em pági. web. As fontes, além de transmitir informações, desempenham um papel fundamental na estética e na legibilidade de uma página. Por isso, o CSS oferece diversas propriedades que nos permitem personalizar fontes de maneira detalhada.

Propriedades de Fontes

Para aplicar estilos às fontes no CSS, utilizamos uma série de propriedades que começam com o prefixo "font". Cada uma dessas propriedades tem uma função específica na personalização das fontes:

font-family: define a família de fontes a ser utilizada. Por exemplo, ao aplicar `font-family: 'Sans Serif';`, estamos instruindo o navegador a utilizar uma fonte da família Sans Serif, que é conhecida por suas linhas limpas e ausência de serifas. Essa escolha é crucial para definir o estilo geral do texto em uma página.

font-size: controla o tamanho da fonte. Um valor comum é `font-size: 16px`, que define um tamanho de 16 pixels para o texto. O tamanho da fonte é um fator importante na acessibilidade e na legibilidade do conteúdo, especialmente em dispositivos móveis.

font-weight: determina a espessura da fonte, permitindo, por exemplo, que um texto seja exibido em negrito (`font-weight: bold`). Podemos também definir valores numéricos que variam de 100 a 900 para ajustar o peso da fonte de forma mais precisa, sendo 400 o peso normal e 700 equivalente ao negrito.

font-style: altera o estilo da fonte, como a aplicação de itálico (`font-style: italic`). Essa propriedade é útil para destacar partes do texto, como citações ou termos técnicos, conferindo um dinamismo à leitura.

Essas propriedades permitem uma ampla gama de personalizações, possibilitando que cada elemento da página tenha uma aparência distinta e adequada ao seu contexto.

Alinhamento de Texto

Além das propriedades de fonte, o CSS também oferece opções para alinhar o texto dentro de um elemento. O alinhamento é uma parte crucial do layout, pois influencia diretamente a forma como o conteúdo é apresentado ao usuário. Entre as principais propriedades de alinhamento, destacam-se:

text-align: controla o alinhamento horizontal do texto. Com `text-align`, podemos alinhar o texto à esquerda, à direita, ao centro ou justificar, dependendo do efeito desejado. Esta propriedade é frequentemente utilizada para centralizar títulos ou alinhar parágrafos de forma uniforme.

line-height: define a altura da linha do texto, controlando o espaçamento vertical entre as linhas. Por exemplo, `line-height: 1.5;` aumenta o espaçamento, tornando o texto mais legível, especialmente em blocos grandes de texto.

vertical-align: ajusta o alinhamento vertical do texto em relação ao elemento que o contém. É útil para alinhar texto em células de uma tabela ou ao lado de imagens, garantindo que o conteúdo fique visualmente equilibrado.

text-indent: indenta a primeira linha de um parágrafo, criando um espaço antes do início do texto. Esta propriedade é comum em contextos onde a formatação tradicional de texto é necessária, como em artigos ou documentos formais.

Aplicando na Prática

Agora que entendemos as propriedades básicas de estilo e alinhamento de fontes, podemos aplicá-las de maneira prática. Vamos supor que estamos estilizando um formulário com botões de submissão e reset. Podemos definir o tamanho da fonte dos botões para `16px` usando `font-size`, e garantir que todo o conteúdo dentro da `body` use a fonte Sans Serif, criando um layout limpo e coerente.

Podemos também ajustar o peso da fonte de um título `h2` para negrito com `font-weight: bold`, garantindo que ele se destaque do restante do texto. É importante notar que, em alguns casos, as mudanças podem ser sutis, especialmente se o elemento já possui um estilo predefinido.

O alinhamento dos textos também deve ser considerado. Podemos, por exemplo, centralizar botões usando `text-align:lef;` para garantir que estejam visualmente alinhados com outros elementos da página. Veja o exemplo abaixo:

```
body {
    font-size: 20px;
    font-family: sans-serif;
    margin: 20px;
    background-color: #f4f4f9;
}
h2 {
    color: #333;
    font-weight: bold;
}
```

```
table {
    border-collapse: collapse;
    margin-bottom: 20px;
}

table, th, td {
    border: 1px solid #ccc;
}

th, td {
    padding: 8px;
    text-align: left;
}
```

Nesta segunda parte da aula, exploramos as propriedades de estilo e alinhamento de fontes no CSS, elementos fundamentais para a criação de páginas web visualmente agradáveis e funcionais. O controle sobre as fontes e o alinhamento do texto permite personalizar a experiência do usuário, tornando o conteúdo mais acessível e atraente.

Layout e posicionamento de elementos

Neste tópico da nossa aula, vamos comentar sobre o conceito de layout e o posicionamento de elementos em uma página web. O layout é uma parte

fundamental do design de páginas HTML, pois permite organizar visualmente os elementos, garantindo que a apresentação seja clara e acessível aos usuários.

O que é um Layout?

O layout em CSS refere-se à maneira como os elementos são organizados em uma página web. Isso inclui a disposição dos blocos de texto, imagens, botões, entre outros elementos. O CSS oferece diversas técnicas e propriedades para controlar essa disposição, permitindo que os elementos sejam posicionados de forma precisa e consistente.

Existem várias abordagens para criar layouts em CSS, mas nesta aula vamos focar em quatro principais: blocos, linhas, flexbox e grid.

Blocos Verticais: quando organizamos elementos em blocos verticais, eles são posicionados um abaixo do outro. Esta é a configuração padrão para elementos de bloco em HTML, como parágrafos (`<p>`) e divs (`<div>`). Esse tipo de layout é ideal para criar seções distintas em uma página.

Linhas: diferente dos blocos verticais, os elementos em linha são posicionados lado a lado. Isso é comum em textos com imagens, onde você deseja que a imagem fique à direita ou à esquerda do texto. Elementos em linha, como `<span>` e `<img>`, permitem criar layouts mais dinâmicos e interativos.

Flexbox: o Flexbox é uma técnica de layout que oferece um controle mais flexível sobre o alinhamento e distribuição dos elementos em uma página, em qualquer direção (horizontal ou vertical). Ele é especialmente útil para criar layouts responsivos, onde os elementos se ajustam automaticamente ao tamanho da tela. No entanto, é importante tomar cuidado ao utilizar o

Flexbox, pois, se não for configurado corretamente, ele pode causar desorganização na página.

Grid: o CSS Grid é uma ferramenta poderosa para criar layouts em duas dimensões, ou seja, que organizam elementos em linhas e colunas. Diferente das tabelas, que são elementos físicos do HTML, o Grid é uma forma de organizar visualmente os elementos, sem afetar a estrutura do documento HTML. Isso permite criar layouts complexos e flexíveis de maneira eficiente.

Posicionamento de Elementos

Além do layout, o posicionamento dos elementos é crucial para definir como eles serão exibidos na página. No CSS, o posicionamento pode ser controlado através das propriedades `position`, que define como um elemento é posicionado em relação ao seu container ou à página.

Posicionamento Absoluto: com `position: absolute;`, um elemento é posicionado em relação ao seu elemento pai mais próximo que tenha uma posição definida (relativa, absoluta ou fixa). Isso significa que o elemento será removido do fluxo normal do documento e posicionado exatamente onde você definir, utilizando as propriedades `top`, `right`, `bottom` e `left`.

Posicionamento Relativo: o `position: relative;` permite que o elemento seja posicionado em relação à sua posição original no documento. Isso significa que ele ainda ocupa espaço no fluxo normal da página, mas você pode ajustá-lo ligeiramente, movendo-o para cima, para baixo, para a esquerda ou para a direita, sem afetar o posicionamento dos outros elementos.

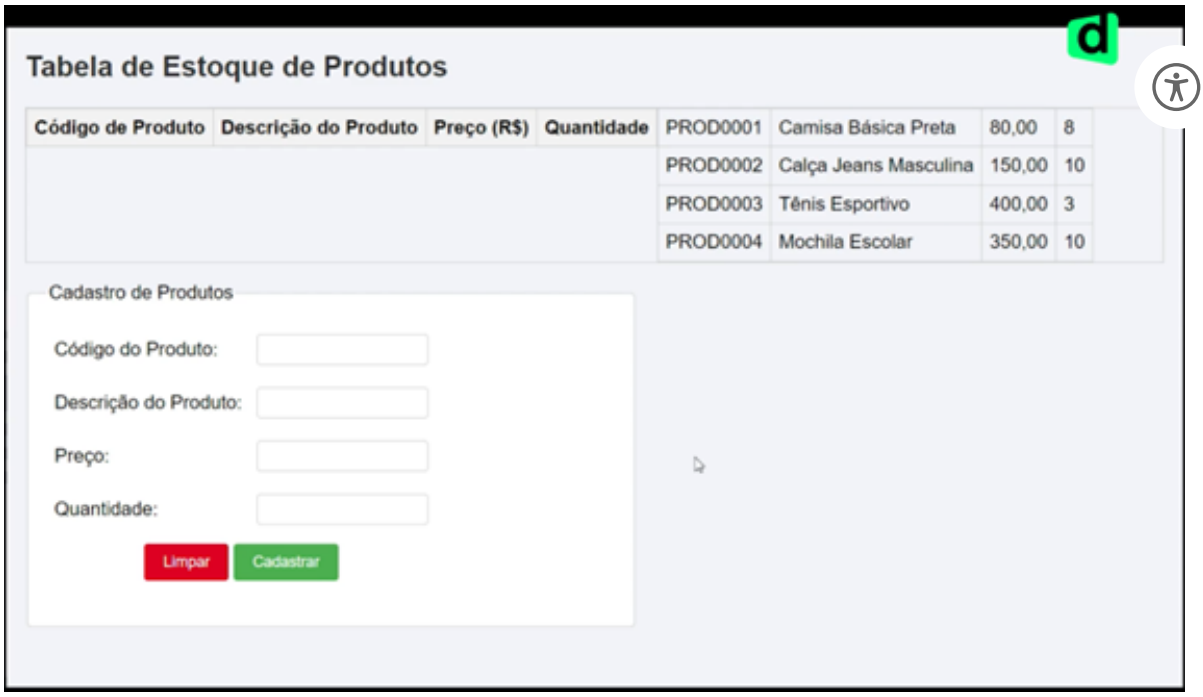
Esses métodos de posicionamento oferecem flexibilidade na organização dos elementos, permitindo criar layouts que se adaptam às necessidades específicas de um projeto.

Aplicando na Prática

Agora que entendemos os conceitos de layout e posicionamento, vamos aplicar esses conhecimentos na prática. Por exemplo, ao trabalhar com uma tabela em HTML, podemos usar a propriedade `display` para definir como os elementos dessa tabela serão organizados. Ao aplicar `display: flex;`, os elementos se reorganizam de maneira flexível, mas é importante lembrar que isso pode desorganizar a tabela se não for feito corretamente. Em casos onde a precisão é crucial, como em tabelas, pode ser melhor deixar que o HTML controle o layout, para evitar problemas de alinhamento. Abaixo estão um exemplo de código implementando a propriedade `display`, e o resultado dessa implementação na nossa tabela:

```
table {
    border-collapse: collapse;
    margin-bottom: 20px;
    display: flex;
}
```

Figura 2 - Implementação da propriedade `display: flex` na Tabela



Fonte: elaborado pelo autor

Aqui, abordamos a importância do layout e do posicionamento dos elementos em CSS. Entender como organizar e posicionar os elementos de forma eficaz é essencial para criar páginas web que não sejam apenas funcionais, mas também esteticamente atraentes. Lembre-se de que, ao trabalhar com layouts e posicionamentos, é fundamental testar as diferentes opções para encontrar a configuração que melhor atende às necessidades do projeto.

Animações e transições

Nesta quarta e última parte da nossa aula, abordaremos as animações e transições, ferramentas que adicionam dinamismo e interatividade às páginas web. O CSS não só permite estilizar elementos, mas também oferece a capacidade de animar e suavizar transições entre diferentes estados de um elemento, tornando a experiência do usuário mais envolvente e fluida.

Introdução às Animações e Transições

As transições e animações em CSS são técnicas poderosas que podem transformar uma página estática em uma experiência interativa e visualmente atraente. Transições permitem que mudanças em propriedades de estilo ocorram de maneira suave e controlada, ao invés de instantaneamente. Isso significa que, ao invés de uma mudança abrupta, como a cor de um botão ao ser clicado, a transição permite que essa alteração aconteça gradualmente, melhorando a usabilidade e a estética da página.

Animações, por outro lado, são mais complexas e permitem criar movimentos e efeitos contínuos que podem durar por um período específico ou indefinido. As animações são ideais para adicionar vida a elementos de uma página, como ícones que se movem ou textos que deslizam, proporcionando uma experiência mais rica ao usuário.

Propriedades de Transições e Animações

Para criar uma transição em CSS, usamos a propriedade `transition`, que nos permite definir quais propriedades serão alteradas, a duração da transição, o tipo de aceleração (ease-in, ease-out, linear, etc.), e um possível atraso para o início da transição.

Exemplo básico de uma transição:

```
button {
    width: 30%;
    transition: width 0.5s;
```

```
}

button:hover {
    width: 40%;
}
```

Neste exemplo, o botão aumenta sua largura de 30% para 40% ao ser "hovered" (quando o cursor passa sobre ele). A transição ocorre em 0.5 segundos, proporcionando um efeito suave e agradável.

Além de `transition`, podemos utilizar a propriedade `animation` para criar animações mais complexas. Animações em CSS envolvem a definição de um conjunto de keyframes que descrevem os estados intermediários da animação. Por exemplo, podemos fazer um elemento mover-se da esquerda para a direita da tela, ou mudar gradualmente sua cor, rotacionar, entre outras possibilidades.

Exemplo básico de uma animação:

```
@keyframes slide {
    from {
        transform: translateX(0);
    }
    to {
```

```
    transform: translateX(100px);
  }
}

div {
  animation: slide 2s forwards;
}
```

Neste exemplo, um `div` se move 100 pixels para a direita ao longo de 2 segundos, uma vez iniciada a animação.

Aplicando na Prática

Vamos aplicar essas técnicas em um exemplo prático. Imagine que você deseja adicionar uma transição aos botões de submissão e reset em um formulário. Ao passar o mouse sobre qualquer um desses botões, você quer que eles aumentem de tamanho para proporcionar uma indicação visual ao usuário de que o botão está ativo.

Primeiro, defina a largura inicial dos botões usando a propriedade `width`, seguida das propriedades `transition-property` e `transition-duration` para especificar a transição desejada. No exemplo, a transição é definida para alterar a largura dos botões de 30% para 40% da largura do elemento pai, com uma duração de 0.5 segundos:

```
input[type="submit"], input[type="reset"] {
```

```
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    width: 30%;
    transition-property: width;
    transition-duration: 0.5s;
}
input[type="submit"]:hover {
    background-color: #45a049;
    width: 40%;
}
input[type="reset"]:hover {
    background-color: #B80E22;
    width: 40%;
}
```

Quando o usuário coloca o cursor sobre qualquer um desses botões, eles expandem de 30% para 40%, criando uma interação visual imediata e destacando botão na interface.

Esse tipo de transição melhora a experiência do usuário ao fornecer feedback visual instantâneo, indicando que o botão está interativo. Essa pequena alteração pode fazer uma grande diferença na percepção do usuário sobre a usabilidade da página.

Nesta aula, exploramos as poderosas ferramentas de animações e transições no CSS, que podem transformar a interação de um usuário com uma página web, tornando-a mais fluida e envolvente. Ao suavizar mudanças de estilo e adicionar movimentos, você pode criar uma experiência mais dinâmica e atraente para os usuários.

Espero que esta aula tenha esclarecido como aplicar transições e animações de maneira eficaz e que você se sinta confiante para implementar esses recursos em seus próprios projetos. Com isso, concluímos nossa aula sobre Estilos CSS, onde abordamos desde seletores e alinhamento de fontes até layouts, posicionamentos e, agora, animações e transições.

Conteúdo Bônus

Para auxiliar nos estudos sobre CSS, falando especialmente de direcionamento e sugestão de referências, recomendo assistir ao vídeo “Como ficar melhor em CSS? Dicas para estudar e se aprofundar em CSS”. Este vídeo é apresentado por Mario Souto, no seu canal no YouTube, Mario Souto - Dev Soutinho, onde ele compartilha dicas e recomendações para quem deseja se tornar mais proficiente em CSS.

Referência Bibliográfica

ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da programação de computadores**. 3.ed. Pearson: 2012.

BRAGA, P. H. **Teste de software**. Pearson: 2016.

GALLOTTI, G. M. A. (Org.). **Arquitetura de software**. Pearson: 2017.

GALLOTTI, G. M. A. **Qualidade de software**. Pearson: 2015.

MEDEIROS, E. **Desenvolvendo software com UML 2.0 definitivo**. Pearson: 2004.

PFLEEGER, S. L. **Engenharia de software**: teoria e prática. 2.ed. Pearson: 2003.

SOMMERVILLE, I. **Engenharia de software**. 10.ed. Pearson: 2019.

Ir para exercício