

Programação Web Responsiva

CSS

Prof. Bruno Azevedo

UNIP – Universidade Paulista



CSS

- Recordando...
- HTML trata da **Estrutura** e do **Conteúdo** do documento.
- CSS trata da **Apresentação** do documento.

Apresentação do Site

```
<h2>Login</h2>
<form>
  <label for="idusuario">Nome de Usuário:</label><br>
  <input type="text" id="idusuario"><br>
  <label for="idsenha">Senha:</label><br>
  <input type="password" id="idsenha"><br><br>
  <input type="submit" value="Enviar">
</form>
```

- O código acima gera o seguinte formulário que não é visualmente muito interessante.

Login

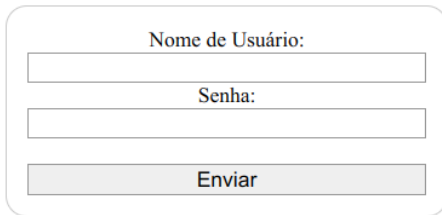
Nome de Usuário:

Senha:

Apresentação do Site

- Com CSS podemos gerar algo assim.

Login



Nome de Usuário:

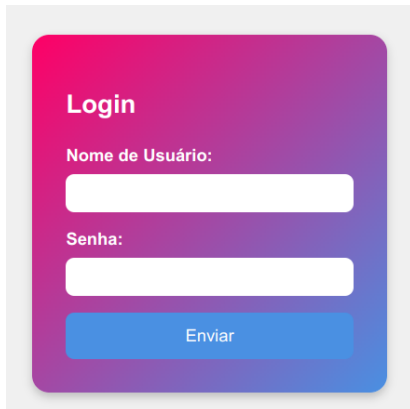
Senha:

Enviar

- Mas podemos fazer algo visualmente ainda mais interessante.

Apresentação do Site

- Por exemplo:



Login

Nome de Usuário:

Senha:

Enviar

Apresentação do Site

Login

Nome de Usuário:

Senha:

Enviar

Login

Nome de Usuário:

Senha:

Enviar

Login

Nome de Usuário:

Senha:

Enviar

- O código HTML permanece **inalterado** em todos os três formulários acima.
- Apenas adicionamos código em CSS. A estrutura é a mesma.

CSS

- CSS é a sigla para Cascading Style Sheets (ou Folha de Estilos em Cascata, em tradução livre).
- É o padrão para definir a apresentação de documentos escritos em HTML.
- Apresentação refere-se a como o documento é exibido ao usuário.
- CSS é uma linguagem distinta de HTML com sua própria sintaxe.

CSS

- Para exemplos do potencial de CSS, visitem o site www.csszengarden.com.
- O CSS Zen Garden é um projeto que permitiu¹ que designers criassem diferentes estilos visuais **para a mesma página HTML**, usando apenas CSS.

¹ Não é atualizado desde 2013.

Exemplos do Zen Garden

- <https://csszengarden.com/>
- <https://csszengarden.com/213/>
- <https://csszengarden.com/221/>
- <https://csszengarden.com/204/>
- <https://csszengarden.com/206/>
- <https://csszengarden.com/219/>
- <https://csszengarden.com/216/>
- <https://csszengarden.com/215/>

CSS - Folhas de Estilo

- Uma folha de estilos contém uma ou mais regras de estilo que determinam a **aparência dos elementos HTML**.
- Cada **regra CSS** é composta por um seletor e uma ou mais declarações.
- O exemplo a seguir demonstra duas regras:
 - A primeira regra define que todos os títulos <h1> do documento serão verdes.
 - A segunda regra define que os parágrafos (<p>) utilizarão a fonte Arial com tamanho 18px.

```
h1 {  
    color: green;  
}  
p {  
    font-size: 18px;  
    font-family: 'Arial';  
}
```

CSS - Regras

- A estrutura de uma regra CSS é composta por dois elementos principais:
 - Seletor: o elemento HTML que será estilizado.
 - Declaração: define as **propriedades e seus valores**, indicando como o elemento deve ser exibido.

```
seletor {  
  propriedade1: valor1;  
  propriedade2: valor2;  
  propriedade3: valor3;  
}
```

CSS

- Um exemplo completo para visualizarmos:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        background-color: blue;
      }
      h1 {
        color: white;
      }
      p {
        font-family: verdana;
        font-size: 20px;
        color: red;
      }
    </style>
  </head>
  <body>
    <h1> Um título personalizado! </h1>
    <p> Um parágrafo personalizado! </p>
  </body>
</html>
```

Seletores

- Declarações são descritas como um par **propriedade/valor**.

```
p {  
    color: blue;  
}
```

- Um seletor pode ter mais de uma declaração.

```
p {  
    font-size: 30px;  
    color: blue;  
}
```

- **Cada declaração deve terminar com ponto e vírgula (;)**, permitindo a separação entre diferentes declarações.

Seletores

- CSS ignora espaços em branco e novas linhas.
- Ambos códigos abaixo resultam na **mesma estilização**.

```
p {font-size:large;font-family:sans-serif;}
```

```
p {  
    font-size:large;  
    font-family:sans-serif;  
}
```

Seletores

- Existem diferentes tipos de seletores e um deles é o **seletor de tipo de elemento**.
- Outros tipos de seletores são o **seletor de id** e o **seletor de classe**.
- Estes selecionam os elementos de acordo com seus ids e classes, respectivamente.

Seletores e Tipos de Elementos

- Vamos selecionar um **tipo de elemento** e estilizá-lo:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        font-family: monospace;
        font-size: 30px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <p>Um parágrafo personalizado!</p>
  </body>
</html>
```


Seletores e Classes e IDs

- O atributo **class** é usado para estilizar múltiplos elementos ao mesmo tempo.
- O atributo **id** permite estilizar um único elemento de forma exclusiva.

```
<div id="noticia-principal">  
  <h1 class="titulo">Nova Descoberta Científica</h1>  
  <p class="resumo">Pesquisadores anunciam um avanço  
    revolucionário na medicina.</p>  
</div>
```

- O **id** noticia-principal identifica exclusivamente a notícia mais importante da página.
- A **class** titulo estiliza o título da notícia.
- A **class** resumo estiliza o parágrafo que resume a notícia.

Seletores e IDs

- Vamos selecionar um elemento **através de seu id** e estilizá-lo:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #meuid {
        font-family: monospace;
        font-size: 30px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <p> Um parágrafo não-personalizado. </p>
    <p id="meuid"> Um parágrafo personalizado. </p>
  </body>
</html>
```

Seletores e Classes

- Vamos selecionar um elemento **através de sua classe** e estilizá-lo:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .minhaclasse {
        font-family: monospace;
        font-size: 30px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <p> Um parágrafo não-personalizado. </p>
    <p class="minhaclasse"> Um parágrafo personalizado. </p>
  </body>
</html>
```

Seletores, Classes e IDs

Resumindo:

- Usamos o nome do elemento para estilizá-lo.

```
p {  
    font-family: monospace;  
    font-size: 30px;  
    color: blue;  
}
```

- Se estamos estilizando um ID, usamos o símbolo # seguido do nome do identificador.

```
#meuid {  
    font-family: monospace;  
    font-size: 30px;  
    color: blue;  
}
```

- Se estamos estilizando uma classe, usamos o símbolo . seguido do nome da classe.

```
.minhaclasse {  
    font-family: monospace;  
    font-size: 30px;  
    color: blue;  
}
```

Adicionando Estilos

- Existem diferentes modos para adicionarmos estilos CSS à uma página HTML, conheceremos três modos:
 - *Inline.*
 - Interno
 - Externo.

Adicionando Estilos

- O primeiro método é a adição de estilos na linha do elemento HTML que desejamos alterar, utilizando o atributo `style`.

```
<h1 style="color: red"> Esse título está na cor vermelha!!</h1>
```

- O segundo método é a adição do código CSS **diretamente** no código HTML, usando a tag `<style>`, que deve ser colocada dentro do `<head>` do documento HTML.

```
<head>  
  <style>  
    /* Estilos CSS diretamente no documento HTML. */  
  </style>  
</head>
```

Adicionando Estilos

- O terceiro método envolve o uso de um arquivo com sufixo .css, que é *linkado* ao documento HTML.

```
<head>  
  <link rel="stylesheet" href="/caminho/estilo.css">  
</head>
```

- No arquivo .css, definimos as **regras de estilização** que serão aplicadas ao nosso site.

Propriedade color

- A propriedade color é usada para definir a cor do texto.
- Pode aceitar valores em nomes de cores (ex: red, blue), hexadecimais (ex: #ff0000), RGB (ex: rgb(255,0,0)) e HSL.
- Exemplo:

```
h1 {  
    color: blue;  
}
```

- A propriedade color afeta apenas o texto, não o fundo do elemento.

Propriedade Background-color

- Define a cor de fundo de um elemento.
- Pode aceitar valores em nomes de cores (ex: red, blue), hexadecimais (ex: #ff0000), RGB (ex: rgb(255,0,0)) e HSL.
- Exemplo:

```
body {  
    background-color: lightblue;  
}
```

```
.minhaNovaClasse {  
    background-color: #f0f0f0;  
}
```

Propriedade padding

- Define o espaçamento interno de um elemento, ou seja, a distância entre o conteúdo e sua borda.
- Pode ser definida em várias unidades: px, %, em, entre outras.
- Exemplo usando o mesmo valor para todos os lados:

```
p {  
    padding: 20px;  
}
```

- Também é possível definir valores diferentes para cada lado, na ordem: top right bottom left:

```
p {  
    padding: 10px 15px 20px 5px;  
}
```

Propriedades width e height

- width define a **largura** de um elemento.
- height define a **altura** de um elemento.
- Ambas podem ser definidas em diversas unidades: px, %, em, rem, vh, vw, entre outras.
- Exemplo:

```
header {  
    width: 100%;  
    height: 80px;  
}
```

- A unidade % se refere ao tamanho do elemento pai.
- Se não forem definidas, o navegador pode calcular os valores automaticamente com base no conteúdo e no contexto.

Propriedade font

- A propriedade `font` é uma forma abreviada para definir várias propriedades de estilo de fonte em uma única linha.
- A sintaxe inclui:
 - `font-family`: define a fonte.
 - `font-size`: define o tamanho do texto.
 - `font-weight`: define a espessura da fonte.
 - `font-style`: define o estilo da fonte (ex: itálico).

- Exemplo:

```
h1 {  
    font: bold 24px 'Arial', sans-serif;  
}
```

- Neste exemplo, a fonte será Arial (ou uma fonte `sans-serif` se Arial não estiver disponível), com tamanho de 24px e espessura `bold`.

Usando Fontes de Terceiros

- Podemos utilizar fontes externas de serviços como **Google Fonts**² para personalizar o design do site.
- Para importar uma fonte, adicionamos um link dentro da seção <head> do HTML:

```
<head>  
  <link rel="stylesheet"  
        href="https://fonts.googleapis.com/css?family=Roboto">  
</head>
```

- Após importar, aplicamos a fonte ao nosso site usando CSS:

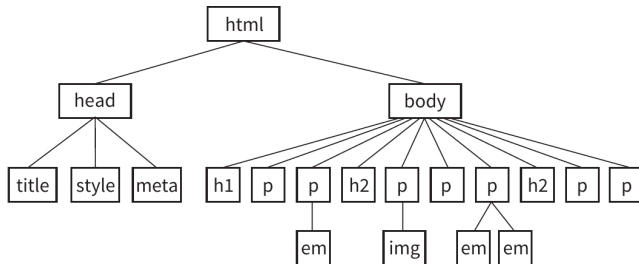
```
body {  
  font-family: 'Roboto', sans-serif;  
}
```

- Sempre defina uma fonte genérica de fallback (como sans-serif) para garantir compatibilidade.

²Tutorial para usar Google Fonts:

Herança

- Elementos podem **herdar** propriedades de elementos que os contém.
- Para entendermos isso, vamos olhar um exemplo de estrutura de um site.

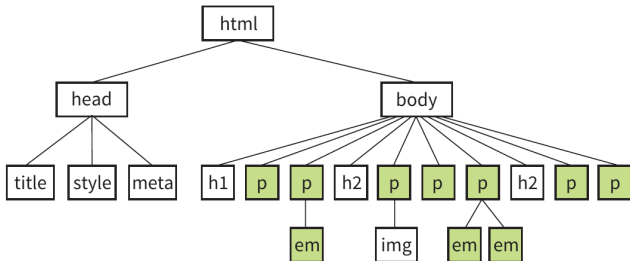


- Um elemento B contido dentro de um elemento A é **filho** de A.
- Analogamente, o A é pai de B.
- Elementos de mesmo pai são irmãos.
- Elementos acima na hierarquia são elementos ancestrais.

Herança

- Entretanto, nem todo elemento herdará propriedades de outros elementos.
- Existem várias razões para isso.
- Exemplo: considere que temos o seguinte código aplicado ao elemento p:

```
p {  
    font-size: large;  
    font-family: sans serif;  
}
```



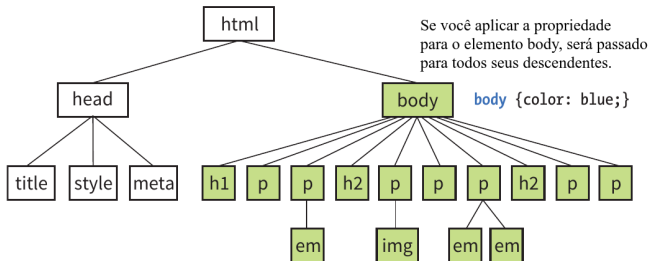
- A imagem que é filho de um dos elementos `p` do documento **não** irá herdar essas propriedades.
- Afinal, tais propriedades não se aplicam a imagens.

Herança

- Devemos usar a herança para nossa vantagem quando escrevendo folhas de estilo.
- Não necessariamente precisaremos definir uma regra para elemento, mas usar a herança para atribuir diretamente propriedades para os elementos filho.
- Por exemplo, se queremos todos os elementos de texto em azul, seria muito trabalhoso escrever regras para todo elemento do documento.

Herança

- Uma alternativa melhor seria escrever uma única regra que aplica a propriedade de cor para o elemento **body**.
- Todos os elementos em body herdarão esse estilo.



A cor mostrará para a imagem apenas se ela tiver borda.

Cascading Style Sheets

- O CSS permite que você aplique várias folhas de estilo ao mesmo documento, ou seja, podem haver conflitos.
- Por exemplo, o que o navegador deve fazer se uma folha de estilo importada diz que todos h1 devem ser vermelhos, mas uma folha de estilo incorporada define h1s como roxos?
- Eles anteciparam esse problema e criaram uma **hierarquia** que é avaliada para definir qual folha de estilo será utilizada.
- Vamos entender isso em mais detalhe.

Cascading Style Sheets

- As regras de CSS são aplicadas primeiro considerando sua prioridade, então sua especificidade, e por último, a ordem de apresentação das regras no documento.

```
/* Regra definida com prioridade importante */  
p {color: blue !important;}
```

- As especificidades relacionam-se ao quanto específica ou geral é a regra. Por exemplo, ela se aplica a todos os parágrafos ou apenas um específico?

```
<!-- Parágrafo sem ID -->  
<p>Este é um parágrafo sem um atributo ID.</p>  
<!-- Parágrafo com ID -->  
<p id="meuparagrafo">Este é um parágrafo com um atributo ID.</p>
```

- Finalmente, temos a ordem de escrita das regras. A regra que vem por último é a aplicada.

```
p {  
    color: red;  
    color: blue;  
    color: green;  
}
```

Atividade Prática 9

- ❶ Criar uma página HTML chamada `estilos.html` contendo:
 - ❶ Incluir os seguintes elementos estilizados com CSS:
 - Um título `<h1>` com uma fonte personalizada do Google Fonts.
 - Um fundo azul claro utilizando `background-color`.
 - Um parágrafo `<p>` com cor de texto diferente e espaçamento interno (`padding`).
 - Um `<header>` com largura de 100% e altura de 80px.
 - Um `<footer>` com cor de fundo escura e texto claro.
 - ❷ Os alunos devem testar as alterações no navegador.
 - ❸ Ampliação: desenvolver dois estilos distintos e indicar a aplicação de cada um em um tipo específico de negócio.
- ❷ Cada aluno deve produzir um relatório de 1 a 3 páginas contendo:
 - Resumo teórico: explicação sobre a estrutura e funcionamento do CSS.
 - Código-fonte comentado: explicação de cada alteração no CSS.