Desenvolvimento Web no lado cliente – CSS 2023/1

GSI019 - Programação para Internet

Prof. Dr. Rafael D. Araújo

rafael.araujo@ufu.br

http://www.facom.ufu.br/~rafaelaraujo

UFU 45 ANOS

Linguagem de estilos CSS

- **C**ascading **S**tyle **S**heets - Folhas de Estilo em Cascata
- Linguagem de estilos que define o layout de documentos Web
  - Não é linguagem de programação
- Controla o posicionamento e formatação dos elementos HTML
  - fontes, cores, espaçamentos, bordas etc.
  - regras baseadas na seleção dos elementos
- Separa o conteúdo (responsabilidade do HTML) da apresentação gráfica
- É uma especificação do W3C

Exemplo

Página com código HTML puro, sem CSS

Index

- Item A
- Item B
- Item C
- Item D
- Item E

1. Item A
2. Item B
3. Item C
4. Item D
5. Item E

Fig. 1 - Legenda da Figura

Seção 1

Lorem ipsum dolor sit amet. Sit vero voluptates sit accusamus eveniet sit incidunt iste sit unde beatae eos voluptas laudantium. Aut fuga consectetur est ullam officiis in repellat reprehenderit eos temporibus velit non exercitationem minus et facilis cumque quo dolorem similique! (ver Seção 3).

Seção 2

Aut dolores provident sit omnis quaerat ea cumque doloremque. Ea repudiandae nemo in atque cupiditate eum labore recusandae. In quas dolorem et quisquam tempore est repudiandae expedita aut dolores nihil est omnis voluptas et libero rerum.

Seção 3

Quo Quis quos sit voluptatem sunt ea eaque fuga et enim repellat At atque aperiam qui quia aliquam ut dolor adipisci. Aut quaerat odit et neque nemo ea rerum molestiae nam doloremque mollitia rem impedit voluptatem ut ullam consequatur. Aut voluptas labore ea aspernatur quam aut esse minima! Quo reiciendis autem eos saepe eligendi qui earum excepturi nam ducimus facere.

Página com o mesmo código HTML, com CSS incluído

Index

- *Item A*
- *Item B*
- *Item C*
- *Item D*
- *Item E*

1. *Item A*
2. *Item B*
3. *Item C*
4. *Item D*
5. *Item E*

Fig. 1 - Legenda da Figura

Seção 1

Lorem ipsum dolor sit amet. Sit vero voluptates sit accusamus eveniet sit incidunt iste sit unde beatae eos voluptas laudantium. Aut fuga consectetur est ullam officiis in repellat reprehenderit eos temporibus velit non exercitationem minus et facilis cumque quo dolorem similique! (ver Seção 3).

Seção 2

Aut dolores provident sit omnis quaerat ea cumque doloremque. Ea repudiandae nemo in atque cupiditate eum labore recusandae. In quas dolorem et quisquam tempore est repudiandae expedita aut dolores nihil est omnis voluptas et libero rerum.

Seção 3

Quo Quis quos sit voluptatem sunt ea eaque fuga et enim repellat At atque aperiam qui quia aliquam ut dolor adipisci. Aut quaerat odit et neque nemo ea rerum molestiae nam doloremque mollitia rem impedit voluptatem ut ullam consequatur. Aut voluptas labore ea aspernatur quam aut esse minima! Quo reiciendis autem eos saepe eligendi qui earum excepturi nam ducimus facere.

Três formas de inserir CSS no HTML

- Embutido no próprio elemento HTML (*inline*)
  - Atributo `style`
  - O uso deve ser evitado (difícil manutenção)
- Folha de estilos embutida na página (interno)
  - Utiliza o elemento `<style>` dentro do `<head>`
  - Estilos específicos da página, não compartilhados
- Folha de estilos em arquivo separado (externo)
  - Utiliza o elemento `<link>` para referenciar um arquivo com código CSS
  - Várias páginas podem utilizar o código CSS do arquivo
  - Melhor separação entre conteúdo e estilos

Embutido no próprio elemento HTML (inline)

Atributo para inserção de CSS *inline*

Código CSS *inline*

```
<p style="font-size: 14pt; color: blue;">
    Texto em azul com fonte tamanho 14 pontos
</p>
```

O código CSS afetará apenas o elemento em questão. **Não é uma boa prática**.

Folha de estilos embutida na página (interno)

```
<html>
    <head>
        <style>
            p {
                font-size: 14pt;
                color: blue;
            }
        </style>
    </head>
    <body>
        <p>...</p>
    </body>
</html>
```

Código CSS embutido no cabeçalho do documento HTML

O código CSS será aplicado a todo o documento HTML. Separa melhor o código CSS do HTML, mas ainda **não consegue ser reutilizado em outras páginas**.

CSS em folha de estilos separada

```
<html>
    <head>
        <link rel="stylesheet" href="css/meuestilo.css">
    </head>
    <body>
        <p>...</p>
    </body>
</html>
```

- Provê melhor separação de conteúdo (HTML) e estilos (CSS)
- Permite que várias páginas utilizem o mesmo código CSS
- Maior facilidade de manutenção

Arquivo meuestilo.css

```
p {
    font-size: 14pt;
    color: blue;
}
```

Validação do código CSS

- Exibição adequada no navegador não é garantia de código correto
  - O navegador pode ocultar erros e inconsistências
- Código fora da especificação pode trazer problemas diversos
  - Apresentação inconsistente e imprevisível nos navegadores
- Ferramenta oferecida pelo W3C para validação de código CSS
  - https://jigsaw.w3.org/css-validator/

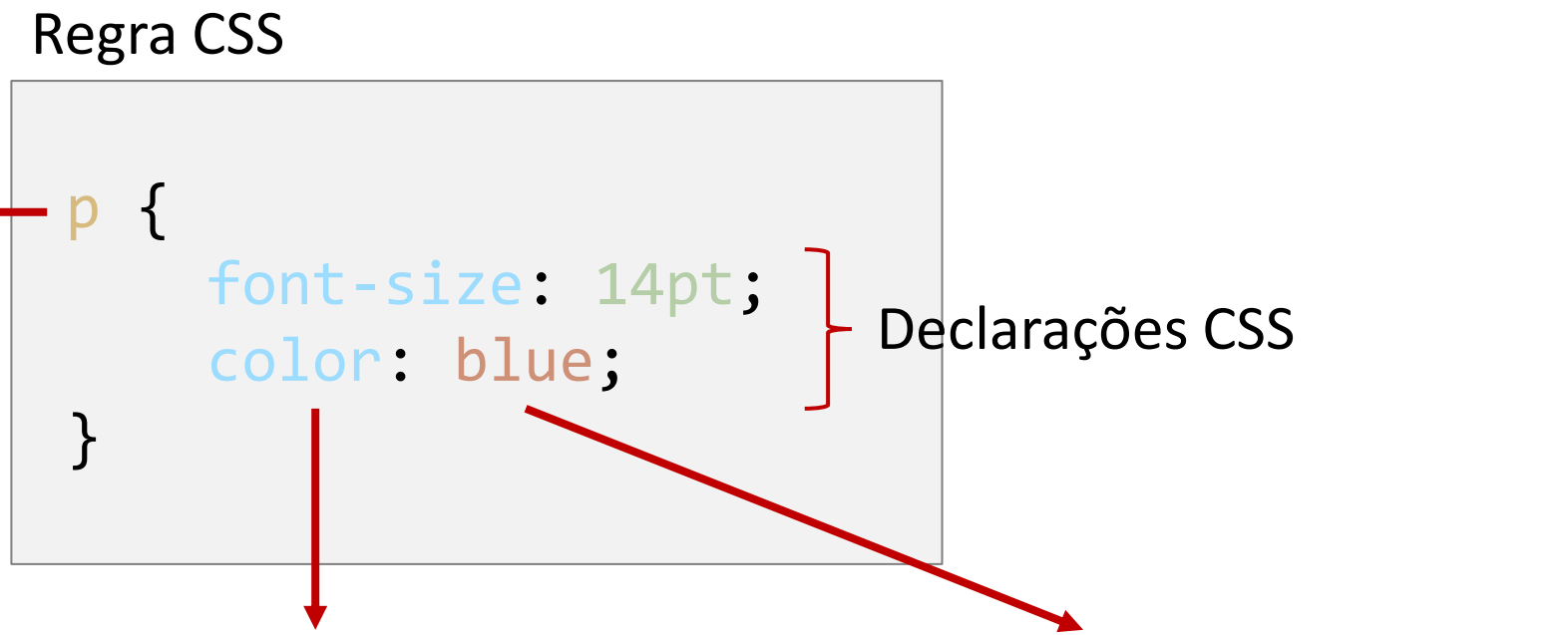
CSS e cache do navegador

- O navegador pode armazenar estilos CSS em memória
- Mudanças nos estilos CSS podem não ter efeito imediato
- Se preciso, tecle Ctrl+F5 para forçar o navegador a recarregar os estilos
- Outra possibilidade é excluir os dados de navegação (cache) do navegador

Regra, Seletor e Propriedades

Seletor da regra

Indica quais partes da página serão afetadas pela regra CSS. Neste exemplo, o seletor da regra é um seletor de elemento indicando que todos os elementos p devem ser alterados com os estilos da regra



Múltiplas regras e seletores

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

p {
        font-size: 20pt;
        color: blue;
}

h1 {
        font-family: Verdana;
}
```

Afetará o corpo do documento HTML

Afetará todos os parágrafos p

Afetará todos os títulos h1

Agrupando seletores

```
h1, h2, h3 {
        font-size: 20pt;
        font-family: Verdana;
        color: blue;
}
```

Ao invés de criar várias regras com os **mesmos estilos**, pode-se criar uma única regra agrupando os seletores.

Tipos de seletores

Tipo	Código
Seletor de tipo de elemento	`p { }`
Seletor de ID	`#par1 { }`
Seletor de filho	`x > y { }`
Seletor de descendente	`x y { }`
Seletor de irmão adjacente	`x + y { }`
Seletor de irmão geral	`x ~ y { }`
Seletor de atributo	`x[atributo] { }`
Seletor de classe	`.classe1 { }`

Seletor de tipo de elemento

```
<html>
    <head>
        <style>
            p {
                color: blue;
            }
        </style>
    </head>
    <body>
        <p>...</p>
        <p>...</p>
    </body>
</html>
```

Elementos afetados

O seletor de tipo de elemento é um seletor básico usado para escolher todos os elementos de um tipo específico de marcador HTML.

Afeta todos os elementos daquele tipo.

Seletor de ID (`#idDoElemento`)

```
<html>
    <head>
        <style>
            #paragrafo1 {
                font-size: 14pt;
                color: blue;
            }
        </style>
    </head>
    <body>
        <p id="paragrafo1">...</p>
        <p>...</p>
    </body>
</html>
```

Elemento afetado

O seletor de ID pode ser utilizado quando se deseja aplicar estilos a apenas um elemento em particular. Utiliza-se **#** seguido do **id** do elemento.

Afeta apenas o elemento que tem o **id** indicado no seletor.

Seletor de filho (x > y)

```
...
        <style>
            p > a {
                color: red;
            }
        </style>
...
    <body>
        <a href="#">Link 1</a>
        <p class="classe1">
            <a href="#">Link 2</a>
            <a href="#">Link 3</a>
        </p>
    </body>
</html>
```

Elementos afetados

O seletor de filho tem a sintaxe **x > y** e afeta todos os elementos **y** que **são filhos** de elementos **x**.

Seletor de descendente (x y)

```
...
        <style>
            div a {
                color: red;
            }
        </style>
...
    <body>
        <a href="#">Link 1</a>
        <div>
            <a href="#">Link 2</a>
            <section>
                <a href="#">Link 3</a>
            </section>
        </div>
    </body>
</html>
```

Elementos afetados

Afeta todos os elementos **y** que **estão dentro** de elementos **x**, mesmo que tenha outros elementos aninhados entre eles.

Seletor de irmão adjacente (x + y)

```
<style>
    section + p {
        color: red;
    }
</style>
```

Afetará todo parágrafo p que está imediatamente depois de um <section>

```
<body>
    <section>
        <p>Parágrafo 1</p>
    </section>
    <p>Parágrafo 2</p>

    <section>
        <p>Parágrafo 3</p>
    </section>
    <p>Parágrafo 4</p>
</body>
```

Elementos afetados

Afeta todo elemento **y** que aparece **imediatamente depois** (próximo elemento irmão) de elementos **x**.

Seletor de irmão geral (x ~ y)

```
<style>
    section ~ p {
        color: red;
    }
</style>
```

Afetará todos parágrafo p irmãos e que aparecem depois de um <section>

```
<body>
    <section>
        <p>Parágrafo 1</p>
    </section>
    <p>Parágrafo 2</p>
    <p>Parágrafo 3</p>

    <section>
        <p>Parágrafo 4</p>
    </section>
    <p>Parágrafo 5</p>
</body>
```

Elementos afetados

Afeta todos os elementos **y** que **são irmãos** de **x** e aparecem **depois** de **x**.

Seletor de atributo (`x[atributo]`)

```
...
    <style>
      input[required]{
          background-color: blue;
      }
    </style>
...
    <form>
       <label for="nomePessoa">Nome:</label>
       <input type="text" id="nomePessoa" required>

       <label for="emailPessoa">E-mail:</label>
       <input type="email" id="emailPessoa" required>

       <label for "cidadePessoa">Cidade:</label>
       <input type="text" id="cidadePessoa">
    </form>
...
```

Seleciona os elementos <input> que possuem o atributo required

Elementos afetados

Seleciona todos os elementos **x** de acordo com alguma condição envolvendo seus atributos.

Seletor de atributo (`x[atributo]`)

```
...
    <style>
      input[type="text"]{
          background-color: blue;
      }
    </style>
...
    <form>
       <label for="nomePessoa">Nome:</label>
       <input type="text" id="nomePessoa" required>

       <label for="emailPessoa">E-mail:</label>
       <input type="email" id="emailPessoa" required>

       <label for "cidadePessoa">Cidade:</label>
       <input type="text" id="cidadePessoa">
    </form>
...
```

Seleciona os elementos <input> que possuem o atributo type = text

Elementos afetados

Seleciona todos os elementos **x** de acordo com alguma condição envolvendo seus atributos.

Seletor de atributo (`x[atributo]`)

```
...
    <style>
      input[type="text"]{
          background-color: blue;
      }
    </style>
...
    <form>
       <label for="nomePessoa">Nome:</label>
       <input type="text" id="nomePessoa" required>

       <label for="emailPessoa">Email:</label>
       <input type="email" id="emailPessoa">
    </form>
...
```

Seleciona os elementos <input> que possuem o atributo type = text

Elemento afetado

Seleciona todos os elementos **x** de acordo com alguma condição envolvendo seus atributos.

Seletor de classe (`.nomeDaClasse`)

```
<html>
    <head>
        <style>
            .classe1 {
                font-size: 14pt;
                color: blue;
            }
        </style>
    </head>
    <body>
        <h1 class="classe1">...</h1>
        <p>...</p>
        <p class="classe1">...</p>
    </body>
</html>
```

Elementos afetados

Ideal para situações onde se pretende aplicar os estilos mais de uma vez. Primeiramente, deve-se criar uma **classe** de estilos CSS com os estilos desejados. Utiliza-se o caractere "ponto" seguido do nome da classe.

Afeta todos os elementos que possuem o mesmo nome de classe indicada no atributo `class`.

Seletor de classe (`.nomeDaClasse`)

```
<html>
    <head>
        <style>
            .classe1 {
                font-size: 14pt;
                color: blue;
            }

            .classe2 {
                text-align: center;
            }
        </style>
    </head>
    <body>
        <h1 class="classe1 classe2">...</h1>
    </body>
</html>
```

É possível utilizar mais de uma classe em um mesmo elemento.

Se houver repetição de propriedades, prevalecerão aquelas referenciadas por último (sequencial).

Seletor de classe (`.nomeDaClasse`)

```
<html>
    <head>
        <style>
            p.classe1 {
                font-size: 14pt;
                color: blue;
            }
        </style>
    </head>
    <body>
        <h1 class="classe1">...</h1>
        <p>...</p>
        <p class="classe1">...</p>
    </body>
</html>
```

Elemento afetado

É possível criar um seletor para de uma classe para um tipo de elemento específico.

Atividade 1

- Crie uma página HTML que possua os elementos:
  - `<header>`: uma imagem e um título
  - `<nav>`: lista de menu com 5 itens
  - `<main>`: duas `<section>` com 3 `<article>` em cada
    - Coloque um título para cada seção
    - Dentro de cada artigo, coloque subtítulos e dois parágrafos
  - `<footer>`
- Crie um arquivo CSS separado e inclua no documento HTML
  - Utilize pelo menos uma regra de cada um dos seletores: ID, classe, filho e atributo
  - Podem ser utilizadas quaisquer propriedades

Seletores de pseudo-classes

- Uma pseudo-classe permite alterar o estilo de um elemento quando ele está em um **estado particular** ou quando atende a uma **dada condição**
- Por exemplo, é possível alterar o estilo:
  - dos links que já foram visitados
  - dos campos de formulário com conteúdo inválido
  - dos elementos quando o passo o ponteiro do mouse sobre eles
- Sintaxe: `seletor:pseudo-classe`

Pseudo-classes comumente usadas com links

Pseudo-classe	Descrição	Exemplos
:link	Define o estilo inicial do link	a:link { color: blue; } links não visitados aparecerão em azul
:visited	Define o estilos de links já visitados	a:visited { color: gray; } links visitados aparecerão em cinza
:hover	Define o estilo de exibição do elemento quando o ponteiro do mouse está sobre o mesmo	a:hover { color: gray; } links aparecerão sem o *underline* quando o ponteiro do mouse for posicionado sobre ele
:active	Define o estilo de exibição do elemento quando ativado (botão do mouse pressionado sobre o mesmo)	a:active { color: gray; } links e parágrafos aparecerão em cinza quando o usuário estiver com o botão do mouse pressionado sobre eles

Pseudo-classes comumente usadas com forms

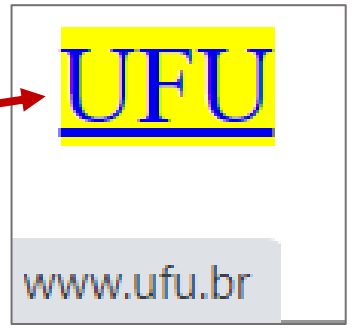
Pseudo-classe	Descrição	Exemplos
:valid	Permite definir o estilo de campos de formulário quando estão no estado válido (conteúdo apropriado)	input:valid { border-color: blue; } campos válidos aparecerão com borda azul
:invalid	Permite definir o estilo de campos de formulário quando estão no estado inválido (conteúdo no formato incorreto)	input:invalid { border-color: red; } campos inválidos aparecerão com borda vermelha
:checked	Permite definir o estilo de campos do tipo radio, checkbox ou option quando selecionado	radio:checked { border-color: green; } campos de opção selecionados aparecerão com borda verde
:focus	Permite definir o estilo do campo quando estiver em foco (campo em edição)	input:focus { border-color: green; } campos que estão com foco (em edição) aparecerão com borda verde

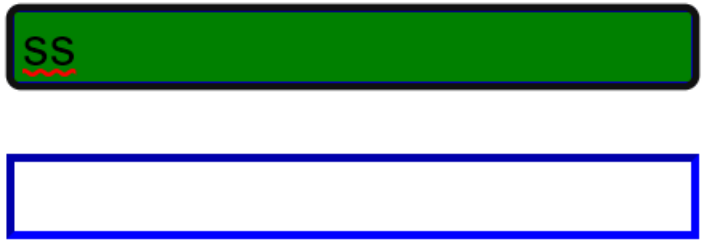
Pseudo-classes - exemplos

```
<style>
    a:hover {
        background-color: yellow;
    }

    input:focus {
        background-color: green;
    }

    input:hover {
        border-color: blue;
    }
</style>
```





Outros exemplos de pseudo-classes

Pseudo-classe	Descrição	Exemplos
:first-child	Permite estilizar o elemento que é o **primeiro filho** do elemento pai	li:first-child { color: blue; } altera o 1º item de cada lista
:last-child	Permite estilizar o elemento que é o **último filho** do elemento pai	li:last-child { color: blue; } altera o último item de cada lista
:nth-child	Permite estilizar o elemento que é o **n-ésimo filho** do elemento pai	li:nth-child(2) { color: blue; } altera o 2º item de cada lista
:first-of-type	Permite alterar a primeira ocorrência de um elemento dentro de seu container	p:first-of-type { color: blue; } altera a cor do 1º parágrafo do container (1º parágrafo dentro de um section, 1º parágrafo dentro de um article etc.)

Propriedades para ajustes de texto

- Tipos de fonte

Fonte sans-serif
Sem prolongamentos
Ex.: Arial, Verdana

Fonte serif
Com prolongamentos
Ex.: Times New Roman

Fonte Monospace
Letas com mesma largura de exibição
Ex.: Consolas, Courier New

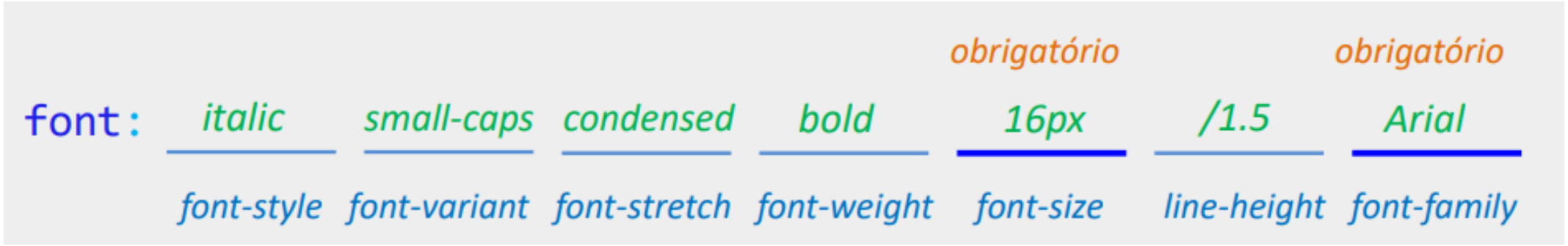
Propriedades para ajustes de texto

O navegador utiliza a primeira (na ordem indicada) que encontrar no computador do usuário. Recomenda-se terminar a lista com o nome de uma família genérica, como **sans-serif** ou **serif**.

Propriedade	Descrição	Exemplo
font-family	Define a fonte em si	font-family: Verdana, Arial, sans-serif
font-style	Define o estilo da fonte	font-style: italic
font-size	Define o tamanho da fonte	font-size: 20px
font-weight	Define a espessura da letra	font-weight: bold
font-variant	Define variantes da fonte	font-variant: small-caps
font-stretch	Estica ou comprime a fonte	font-stretch: expanded
line-height	Define o espaçamento entre linhas	line-height: 1.5

Definir várias propriedades de texto

- Existem propriedades CSS abreviadas, que é um tipo de propriedade que possibilita definir, de uma só vez, várias propriedades relacionadas
- Por exemplo, `font` é uma propriedade abreviada da CSS que nos permite definir, de uma vez, todos os aspectos relacionados à fonte



Regras:
- font-family deve ser o último valor
- font-style, font-variant e font-weight devem vir antes de font-size
- line-height deve vir logo depois de font-size, acompanhado de /

Propriedades para ajustes de texto

Propriedade	Descrição	Exemplo
text-align	Alinhamento horizontal do texto	text-align: left text-align: justify
vertical-align	Alinhamento vertical do texto	vertical-align: top vertical-align: middle
text-decoration	Decoração adicional	text-decoration: none text-decoration: underline
text-indent	Recuo de 1ª linha	text-indent: 30px
text-transform	Controle de maiúsculas e minúsculas	text-transform: uppercase text-transform: lowercase text-transform: capitalize
color	Define a cor do texto	color: green

Ajuste de cor

É possível ajustar:

- Pelo nome da cor
  - `color: blue`, `color: darkblue`, `color: lightblue` etc.
- Pelo valor RGB em Decimal (red, green, blue)
  - `color: rgb(0, 120, 255)`
  - `color: rgba(0, 120, 255, 0.5)` (50% translúcido; 0 = totalmente transp., 1 = opaco)
- Pelo valor RGB em Hexadecimal
  - Notação com 6 dígitos. Exemplo: #FF0000
  - Notação com 3 dígitos. Exemplo: #AF5 (equivalente a #AAFF55)
  - Com transparência. Exemplo: #FF000080 (50% translúcido)
- Pelo código HSL (matiz, saturação, luminosidade)
  - `hsl(m, s, l)`
  - `m: 0-360; s: 0-100%, l: 0-100%;`

Unidades de tamanho

- Unidades de tamanho absoluto
  - A unidade px (pixels) é a unidade de tamanho absoluto mais comum
  - Tamanhos absolutos não dependem de tamanhos definidos no elemento pai
  - Ao definir um tamanho utilizando px, o tamanho não será afetado por eventual mudança no tamanho de fonte feita pelo usuário nas configurações do navegador
- Unidades de tamanho relativo
  - Podem depender de outros tamanhos e configurações como aquelas definidas pelo usuário no navegador, do tamanho definido no elemento pai, do tamanho da *viewport* (região visível da página no navegador) etc.

Unidades de tamanho relativo mais comuns

em – relativo ao tamanho da fonte corrente (herdado do elemento pai)
- 2em = dobro da fonte corrente

rem – relativo ao tamanho da fonte do elemento raiz (html)
- 2rem = dobro do tamanho da fonte do elemento raiz

% – em geral, relativo ao elemento pai
- width: 50% – define a largura em 50% da largura do container

vh – relativo à altura da *viewport* (*viewport height*)
- 30vh corresponde a 30% da altura da *viewport*

vw – relativo à largura da viewport (*viewport width*)
- 100vw corresponde a 100% da largura da *viewport*

ch – relativo à largura de um caractere utilizando a fonte do elemento
- width: 10ch – define a largura para comportar até 10 caracteres

Referências

- https://www.w3.org/Style/CSS/
- https://developer.mozilla.org/en-US/docs/Web/CSS
- Agradecimento especial ao Prof. Daniel Furtado pela disponibilização do material: https://furtado.prof.ufu.br