

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>



Linguagem de estilos CSS

- Cascading Style Sheets - 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*

Regra CSS

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

Declarações CSS

Propriedade CSS

Indica qual característica de apresentação será ajustado

Valor da propriedade a ser alterada

Múltiplas regras e seletores

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



Afetar  o corpo do documento HTML

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



Afetar  todos os par grafos p

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



Afetar  todos os t tulos 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	<code>p { }</code>
Seletor de ID	<code>#par1 { }</code>
Seletor de filho	<code>x > y { }</code>
Seletor de descendente	<code>x y { }</code>
Seletor de irmão adjacente	<code>x + y { }</code>
Seletor de irmão geral	<code>x ~ y { }</code>
Seletor de atributo	<code>x[atributo] { }</code>
Seletor de classe	<code>.classe1 { }</code>

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>
```


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

Elementos afetados

Seletor de descendente (x y)

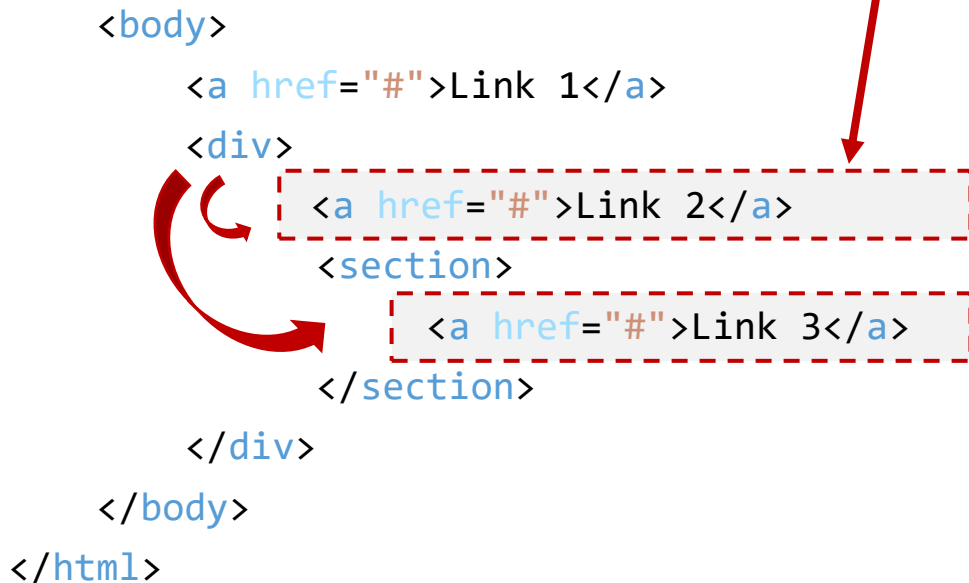
...

```
<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)

```
<body>
  <section>
    <p>Parágrafo 1</p>
  </section>
  <p>Parágrafo 2</p>
  <section>
    <p>Parágrafo 3</p>
  </section>
  <p>Parágrafo 4</p>
</body>
```

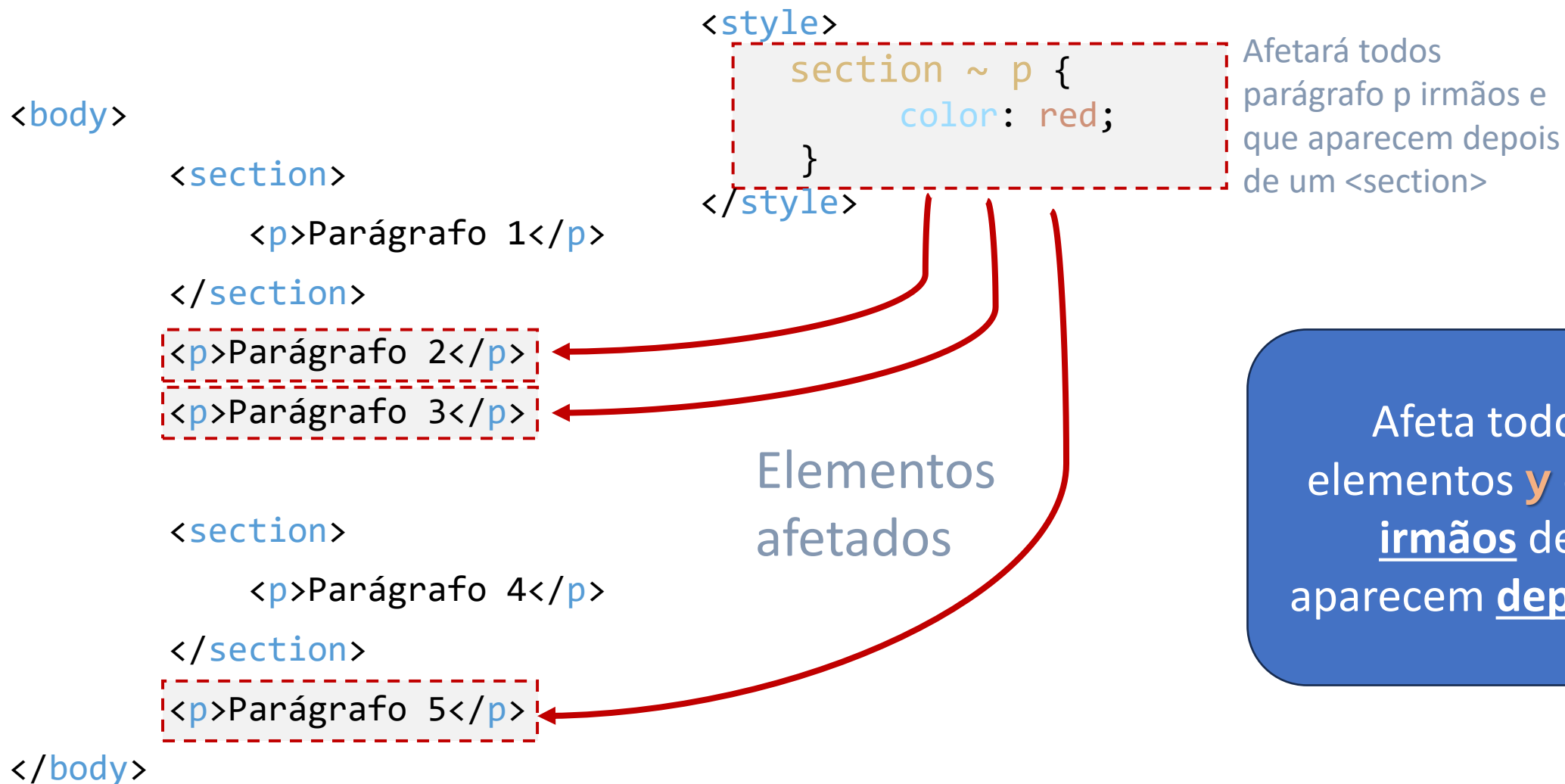
```
<style>
  section + p {
    color: red;
  }
</style>
```

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

Elementos afetados

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

Seletor de irmão geral (x ~ y)



Seletor de atributo (x[atributo])

...

```
<style>
```

```
  input[required]{  
    background-color: blue;  
  }
```

```
</style>
```

Seleciona os elementos
<input> que possuem o
atributo required

...

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

Elementos
afetados

Seletor de atributo (x[atributo])

...

```
<style>
```

```
  input[type="text"]{  
    background-color: blue;  
  }
```

```
</style>
```

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

...

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

Elementos
afetados

Seletor de atributo (x[atributo])

...

```
<style>
```

```
  input[type="text"]{  
    background-color: blue;  
  }
```

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

```
</style>
```

Elemento afetado

...

```
<form>
```

```
  <label for="nomePessoa">Nome:</label>
```

```
  <input type="text" id="nomePessoa" required>
```

```
  <label for="emailPessoa">Email:</label>
```

```
  <input type="email" id="emailPessoa">
```

```
</form>
```

...

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
<code>:link</code>	Define o estilo inicial do link	<code>a:link { color: blue; }</code> links não visitados aparecerão em azul
<code>:visited</code>	Define o estilos de links já visitados	<code>a:visited { color: gray; }</code> links visitados aparecerão em cinza
<code>:hover</code>	Define o estilo de exibição do elemento quando o ponteiro do mouse está sobre o mesmo	<code>a:hover { color: gray; }</code> links aparecerão sem o <u>underline</u> quando o ponteiro do mouse for posicionado sobre ele
<code>:active</code>	Define o estilo de exibição do elemento quando ativado (botão do mouse pressionado sobre o mesmo)	<code>a:active { color: gray; }</code> 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)	<code>input:valid { border-color: blue; }</code> campos válidos aparecerão com borda azul
:invalid	Permite definir o estilo de campos de formulário quando estão no estado inválido (conteúdo no formato incorreto)	<code>input:invalid { border-color: red; }</code> campos inválidos aparecerão com borda vermelha
:checked	Permite definir o estilo de campos do tipo radio, checkbox ou option quando selecionado	<code>radio:checked { border-color: green; }</code> campos de opção selecionados aparecerão com borda verde
:focus	Permite definir o estilo do campo quando estiver em foco (campo em edição)	<code>input:focus { border-color: green; }</code> 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
<code>:first-child</code>	Permite estilizar o elemento que é o primeiro filho do elemento pai	<code>li:first-child { color: blue; }</code> altera o 1º item de cada lista
<code>:last-child</code>	Permite estilizar o elemento que é o último filho do elemento pai	<code>li:last-child { color: blue; }</code> altera o último item de cada lista
<code>:nth-child</code>	Permite estilizar o elemento que é o n-ésimo filho do elemento pai	<code>li:nth-child(2) { color: blue; }</code> altera o 2º item de cada lista
<code>:first-of-type</code>	Permite alterar a primeira ocorrência de um elemento dentro de seu container	<code>p:first-of-type { color: blue; }</code> altera a cor do 1º parágrafo do container (1º parágrafo dentro de um section, 1º parágrafo dentro de um article etc.)

Pseudo-elementos

- Permite selecionar uma parte específica de um elemento
- Sintaxe geral: **seletor::valor**

```
p::first-line {  
  text-transform: uppercase;  
}
```

↪ A primeira linha de cada parágrafo será apresentada com letras maiúsculas

```
p::after {  
  content: 'exemplo after';  
}
```

↪ Insere o texto 'exemplo after' como um pseudo-elemento **depois do conteúdo** do parágrafo

OBS: Os pseudo-elementos `::after` e `::before` não podem ser utilizados em elementos sem conteúdo como `` ou `<input>`

```
p::selected {  
  color: green;  
}
```

O texto que o usuário selecionar nos parágrafos aparecerá na cor verde com fundo preto

```
input::placeholder {  
  color: red;  
}
```

↪ Altera a cor dos textos de placeholder dos campos input

```
p::before {  
  content: 'exemplo before';  
}
```

↪ Insere o texto 'exemplo before' como um pseudo-elemento **antes do conteúdo** do parágrafo

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

font: *italic* *small-caps* *condensed* *bold* *16px* */1.5* *Arial*

font-style *font-variant* *font-stretch* *font-weight* *font-size* *line-height* *font-family*

obrigatório *obrigatório*

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 `#AAF55`)
 - 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

Ajustes de tamanho, margem, espaçamento, borda e fundo

CSS Box Model

Content box

- Região de conteúdo do elemento

Border box

- Região incluindo borda, padding (espaçamento) e conteúdo

Borda do elemento

- Limites do border box

Margem do elemento

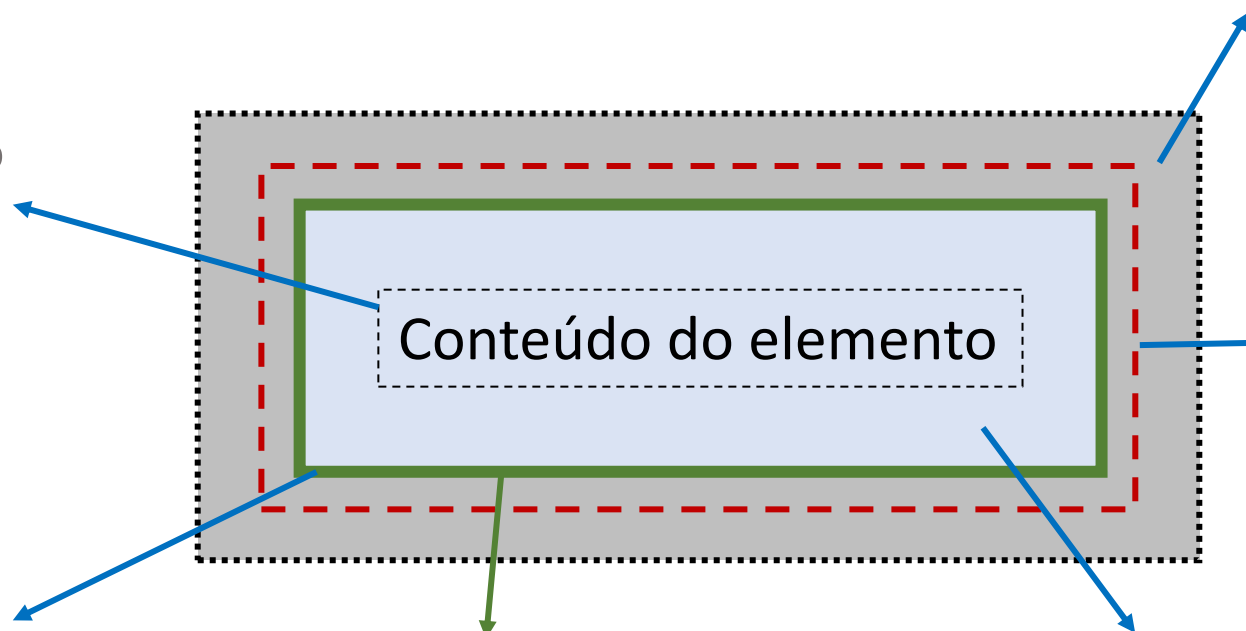
- Externo à borda
- Transparente

Outline do elemento

- Externo à borda
- Não toma espaço próprio

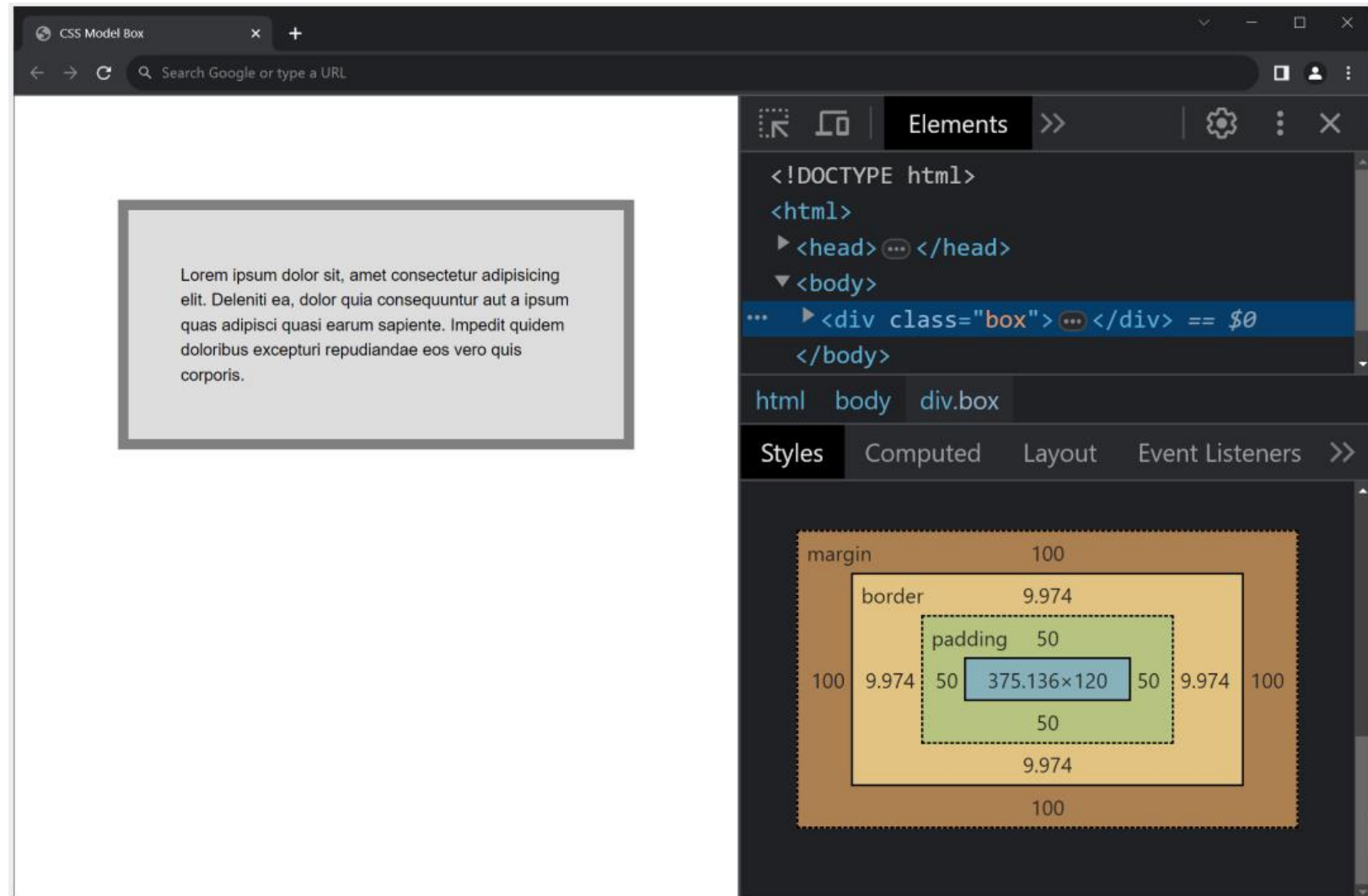
Padding do elemento

- Espaçamento entre o conteúdo e a borda
- Afetado pela cor de fundo
- Dentro da borda



CSS Box Model – Visualização no navegador

- O módulo de desenvolvedor dos navegadores disponibilizam uma área para conferência dos tamanhos da região de conteúdo, margens, paddings e bordas
- No Google Chrome: botão direito do mouse sobre o elemento, selecione inspecionar

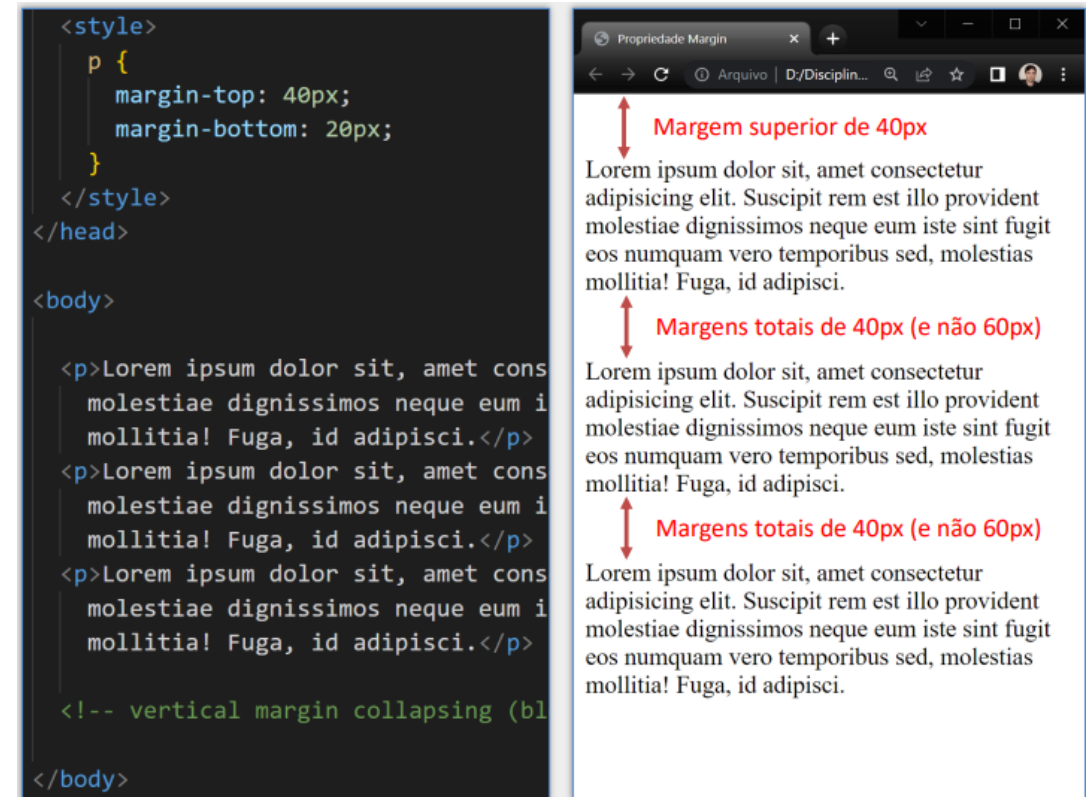
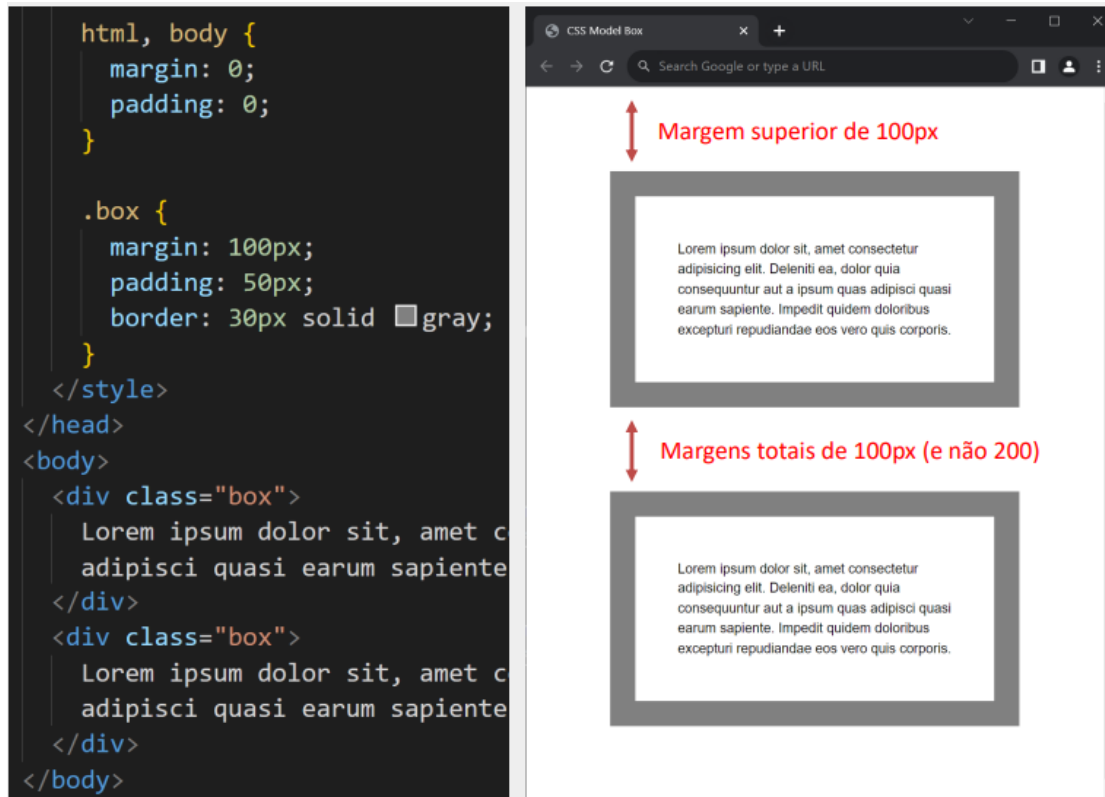


Ajustes de margens e paddings

- É possível fornecer até 4 valores para a propriedade **margin**:
 - **margin**: 20px; (ajusta todas as margens em 20px)
 - **margin**: 20px 50px 80px 100px; (superior, direita, inferior e esquerda)
 - **margin**: 20px 40px 20px; (superior, laterais, inferior)
 - **margin**: 20px 40px; (superior/inferior e laterais)
- A mesma sintaxe se aplica à propriedade **padding**
- Há também propriedades para ajuste individual:
 - **margin-left**, **margin-right**, **margin-top** e **margin-bottom**
 - **padding-left**, **padding-right**, **padding-top** e **padding-bottom**

Sobreposição de margens

- Margens entre dois elementos de bloco vizinhos **não se somam** (a maior delas prevalece)
- Esse efeito é conhecido como **sobreposição de margens** (*margin collapsing*)



Ajustes de borda

- A propriedade border permite definir a borda por completo (4 lados)
- Sintaxe mais comum:
 - **border**: **espessura estilo cor**
 - Estilos possíveis: solid, dotted, dashed, double ou none
 - A ordem dos valores não importa
- Exemplos:
 - **border**: 1px solid blue;
 - **border**: 2px double red;
 - **border**: none;
- Há também propriedades para ajuste individual
 - border-top, border-bottom, border-left, border-right
 - border-color, border-style, border-width

Ajustes de borda

```
#p1 {  
  border: 1px solid blue;  
}  
  
#p2 {  
  border-color: red green blue black;  
  border-style: dashed dotted double solid;  
  border-width: 4px 6px 8px 10px;  
}  
  
#p3 {  
  border-color: gray;  
  border-style: solid none;  
  border-width: 2px;  
}  
  
#p4 {  
  border-top: 1px solid blue;  
}
```

Programação para Internet

Programação para Internet

Programação para Internet

Programação para Internet

Borda e fundo arredondados

- **border-radius** permite arredondar a borda e o background dos elementos
- O valor da propriedade indica o raio da curvatura

```
#p1 {  
  border-radius: 10px;  
}  
  
#p2 {  
  border-radius: 10px 20px 40px 60px  
}  
  
#p3 {  
  border-radius: 10px 40px;  
  border: none;  
}  
  
#p4 {  
  border-radius: 20% 50% 50% 20%;  
}
```



The image shows four light gray rounded rectangular boxes stacked vertically, each with a different border-radius value. The first box has a small radius, the second has a larger radius, the third has a very large radius, and the fourth has a radius that is 50% of its width and height, creating a pill-like shape.

Programação para Internet (P1)

Programação para Internet (P2)

Programação para Internet (P3)

Programação para Internet (P4)

Caixa de sombreamento com box-shadow

`box-shadow: 20px 10px;`

Programação para Internet

`box-shadow: 5px 5px gray;`

Programação para Internet

`box-shadow: 5px 5px 5px gray;`

Programação para Internet

`box-shadow: 0 0 10px cyan;`

Programação para Internet

Ajustes de largura e altura

- Para definir a largura e a altura de um elemento pode-se utilizar as propriedades `width` e `height`
- Na maioria dos casos, `width` e `height` definem tamanhos para a região de conteúdo do elemento (abordagem padrão)
 - Largura/altura total de um elemento = margens + bordas + paddings + conteúdo
 - Esse comportamento pode ser alterado com a propriedade `box-sizing`
- A propriedade `width` **não altera** a **largura total** dos elementos de bloco, pois eles continuarão ocupando toda a largura disponível
- A propriedade `height` não altera a altura de alguns elementos de linha (``, `<a>`, ...)

Ajustes de largura e altura

The image shows a browser window with a CSS box model diagram and its implementation in code and a visual representation.

Diagram (Left): A gray box with text "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Deleniti ea, dolor quia consequuntur aut a ipsum quas adipisci quasi earum sapiente. Impedit quidem doloribus excepturi repudiandae eos vero quis corporis." is centered within a 1000px wide viewport. The box has a width of 500px (50% of the viewport). The diagram shows the following dimensions: 100px margin on the left and right, 30px padding on the left and right, 50px border on the left and right, and 500px content width. The total width of the element is 860px.

Code (Right):

```
<body>
  <div class="box">
    Lorem ipsum dolor
    adipisci quasi ear
  </div>
</body>
```

```
.box {
  width: 50%;
  text-align: justify;
  margin: 100px;
  padding: 50px;
  border: 30px solid gray;
  background-color: #ddd;
}
```

Visual Representation (Bottom Right): A diagram showing the box model layers: margin (100px), border (29.922px), padding (50px), and content (500.260x96px). The total width of the element is 860px.

Text (Bottom Left):

A largura da caixa foi ajustada com "width: 50%". Porém, percebe-se que o elemento como um todo ocupa muito mais da metade da largura disponível, pois há o acréscimo das margens, paddings e bordas. Entretanto, a largura da região de conteúdo (500px) corresponde exatamente a metade da largura do container (1000px)

Outros ajustes de tamanho

- Além das propriedades `width` e `height`, há também propriedades para definir a largura mínima, a largura máxima, a altura mínima e a altura máxima de um elemento:
 - `min-width`, `max-width`
 - `min-height`, `max-height`
- Essas propriedades são especialmente importantes em layouts responsivos
- Outros valores possíveis para propriedades de ajuste de tamanho incluem:
 - `max-content`, `min-content` e `fit-content`

Ajustes de fundo - background

- **background-color**
 - Permite ajustar a cor de fundo do elemento (conteúdo e paddings)
 - Ex.: `div { background-color: gray; }`
- **background-image**
 - Permite inserir uma imagem de fundo para o elemento
 - Ex.: `body { background-image: url("images/bgImage.png"); }`
- **background-repeat**
 - Define o modo de repetição da imagem de fundo (caso ela seja menor)
 - Valores: `no-repeat`, `repeat-x`, `repeat-y`, `repeat`
- **background-size**
 - Permite ajustar o tamanho de exibição da imagem de fundo
 - Ex: `background-size: cover` - estica a imagem para ocupar todo o fundo
- **background**
 - Propriedade abreviada que permite definir todos os aspectos de uma vez (exercício)

Ajustes de exibição e posicionamento

- A propriedade `display` altera o modo de apresentação do elemento
- Um elemento de linha pode ser exibido como bloco e vice-versa
- Permite ocultar um elemento, removendo do layout
- Alguns valores possíveis:
 - `none, block, inline, inline-block`
 - `flex, grid, inline-flex, inline-grid`

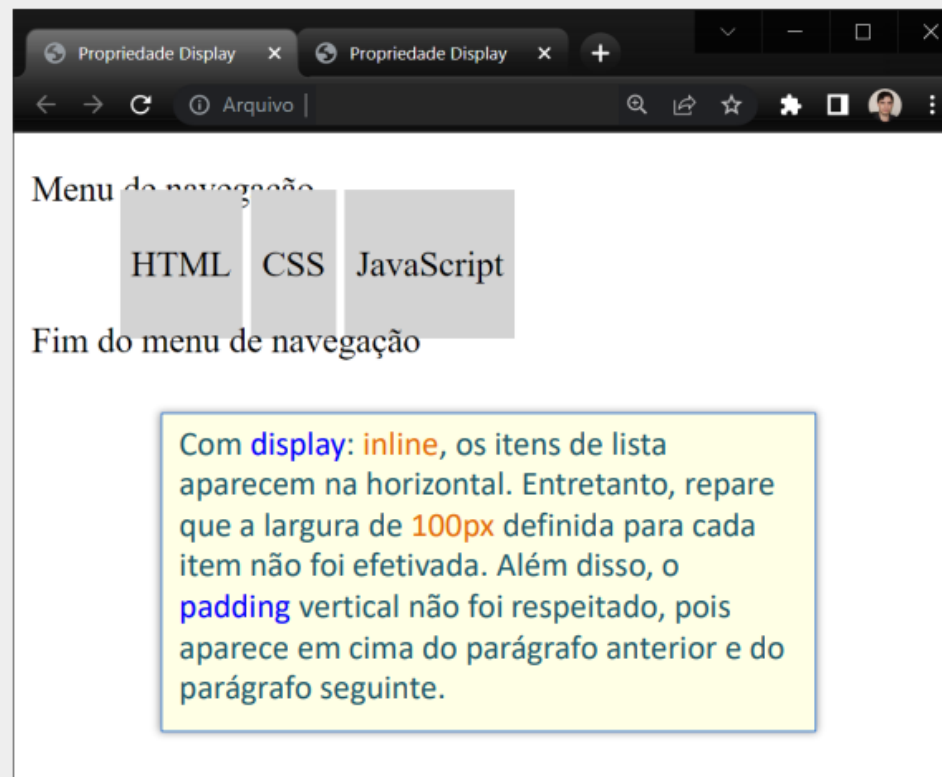
Propriedade `display`

- `display: none`
 - oculta completamente o elemento, removendo o mesmo do layout
 - libera o espaço para outros elementos próximos
- `display: block`
 - exibição em nível de bloco (como o `<div>`)
 - elemento mostrado em nova linha
 - ocupa toda a largura disponível
- `display: inline`
 - exibição em nível de linha (como o ``)
 - `width` e `height` não terão efeito em alguns elementos
 - margens e paddings superiores e inferiores de alguns elementos não são respeitados
- `display: inline-block`
 - exibição em nível de linha
 - possibilidade de usar `width` e `height`
 - margens e paddings superiores e inferiores respeitados

Propriedade `display`

```
<style>
  nav li {
    background-color: lightgray;
    padding: 25px 5px;
    width: 100px;
    display: inline;
  }
</style>
</head>

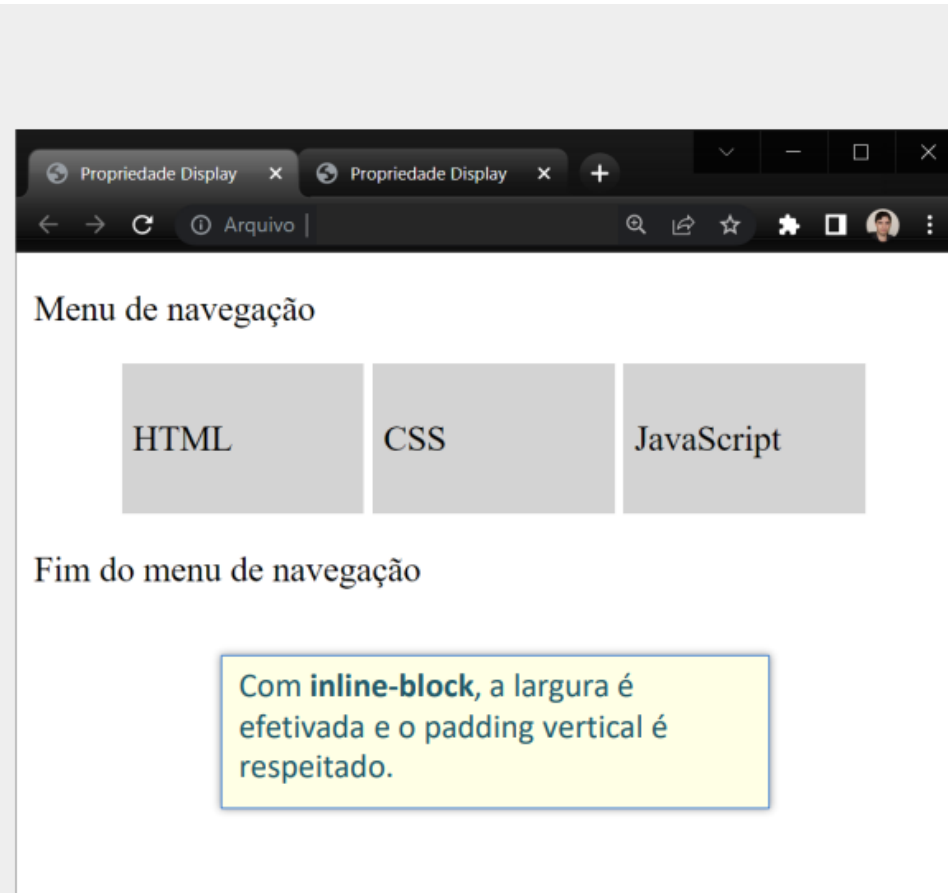
<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>
  </nav>
</body>
```



Propriedade **display**

```
<style>
  nav li {
    background-color: lightgray;
    padding: 25px 5px;
    width: 100px;
    display: inline-block;
  }
</style>
</head>

<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>
  </nav>
</body>
```




Propriedade `visibility`


- Mostra ou oculta um elemento sem alterar o layout
- `visibility: visible`
 - elemento visível, aparecendo normalmente
- `visibility: hidden`
 - elemento oculto, mas continua ocupando espaço no layout
- `visibility: collapse`
 - permite ocultar linhas e colunas de tabelas sem alterar o layout geral
 - as demais linhas e colunas permanecem com seus tamanhos originais

visibility: hidden vs display: none

```
<style>
  div {
    font: 2rem Helvetica;
    display: inline-block;
    width: 100px;
    height: 100px;
    padding: 1rem;
    border-radius: 10px;
    background-color: #eee;
    box-shadow: 0px 0px 5px gray
  }
  .oculto {
    visibility: hidden;
  }
</style>
</head>
<body>
  <div>A</div>
  <div class="oculto">B</div>
  <div>C</div>
</body>
```



```
<style>
  div {
    font: 2rem Helvetica;
    display: inline-block;
    width: 100px;
    height: 100px;
    padding: 1rem;
    border-radius: 10px;
    background-color: #eee;
    box-shadow: 0px 0px 5px gray
  }
  .oculto {
    display: none;
  }
</style>
</head>
<body>
  <div>A</div>
  <div class="oculto">B</div>
  <div>C</div>
</body>
```



visibility: collapse

```
<table>
  <tr>
    <th>Aluno</th>
    <th>Prova 1</th>
    <th>Prova 2</th></tr>
  <tr>
    <td>Fulano</td>
    <td>7,0</td>
    <td>8,0</td></tr>
  <tr>
    <td>Ciclano</td>
    <td>6,0</td>
    <td>9,0</td></tr>
  <tr>
    <th>Media da Turma</th>
    <td>6,5</td>
    <td>8,5</td>
  </tr>
</table>
```

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0
Media da Turma	6,5	8,5

Tabela original com todas as linhas e colunas sendo exibidas

```
.collapse { visibility: collapse; }
.displayNone { display: none; }
</style>
</head>
<body>
  <table>
    ...

    <tr class="collapse">
      <th>Media da Turma</th>
      <td>6,5</td>
      <td>8,5</td></tr>
  </table>
```

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0

Última linha ocultada com a propriedade `visibility: collapse`. Repare que as larguras das colunas permanecem inalteradas.

```
.collapse { visibility: collapse; }
.displayNone { display: none; }
</style>
</head>
<body>
  Propriedade display
  ...

  <tr class="displayNone">
    <th>Media da Turma</th>
    <td>6,5</td>
    <td>8,5</td></tr>
</table>
```

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0

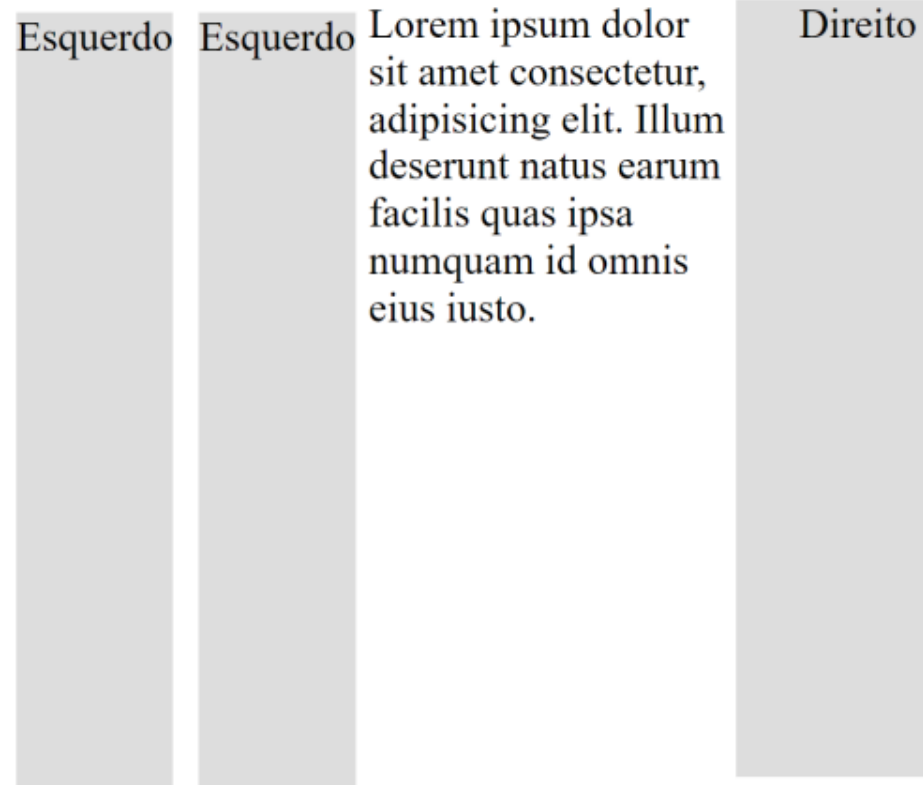
Última linha ocultada com `display: none`. Repare que as larguras das colunas são reajustadas conforme conteúdo (col. **Aluno**)

Propriedade **float**

- Posiciona o elemento no lado esquerdo ou direito de seu container
- Permite que texto e elementos *inline* se posicionem à sua volta
- Alguns valores
 - **left** elemento "flutua" no lado esquerdo do container
 - **right** elemento "flutua" no lado direito do container
 - **none** elemento não "flutua"

Propriedade `float`

```
<style>
  .esquerdo {
    background-color: #ddd;
    width: 15%;
    height: 300px;
    margin: 5px;
    float: left;
  }
  .direito {
    background-color: #ddd;
    width: 25%;
    height: 300px;
    float: right;
  }
</style>
</head>
<body>
  <aside class="esquerdo">Esquerdo</aside>
  <aside class="esquerdo">Esquerdo</aside>
  <aside class="direito">Direito</aside>
  <main><p>Lorem ipsum dolor...</p></main>
</body>
```



Propriedade **overflow**

- Define como o conteúdo do elemento deve ser exibido ao extrapolar a borda
- Propriedade abreviada de **overflow-x** e **overflow-y**
- Alguns valores
 - **visible** : conteúdo sempre visível, ainda que fora dos limites (default)
 - **hidden** : conteúdo cortado, se necessário, para caber no espaço
 - **scroll** : barras de rolagens são sempre apresentadas
 - **auto** : barras de rolagens apresentadas apenas quando necessário

Propriedade **overflow**

Painéis com tamanho fixo (dimensões definidas com **width** e **height**)

The diagram shows three panels, each containing a box with the title "Sistemas de Informação" and two paragraphs of text. The text in each box is identical: "Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos." The first panel has the text fully visible. The second panel has the text hidden, with only the first line visible. The third panel has the text visible with a scrollbar on the right side.

Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

overflow: visible;

Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

overflow: hidden;

Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

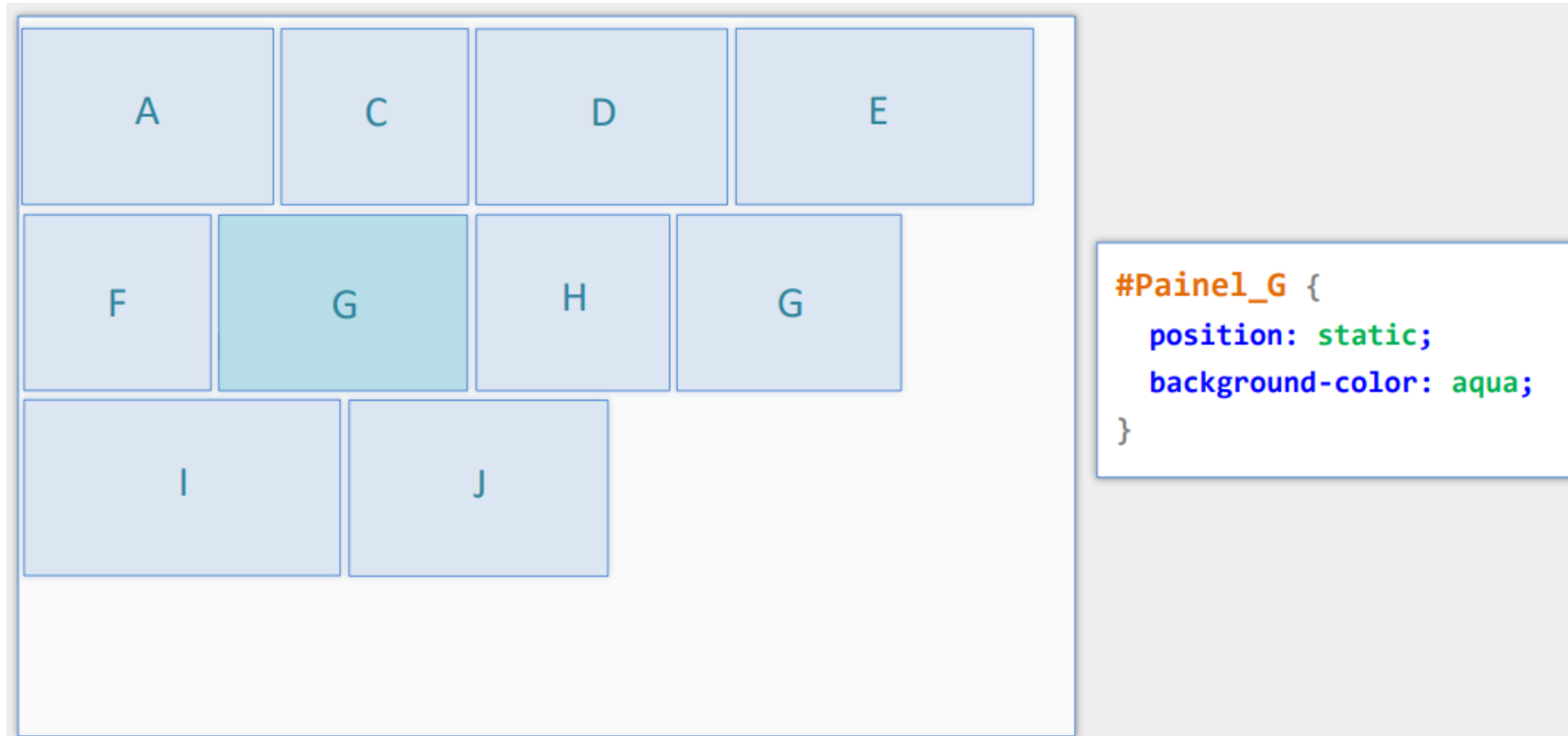
overflow: auto;

Propriedade **position**

- Define como o elemento é posicionado na página
- Normalmente é utilizada em conjunto com as propriedades **top**, **left**, **right** e **bottom**
- Valores possíveis: **static**, **relative**, **absolute**, **fixed**, **sticky**
- O elemento é dito posicionado quando **position** tem valor diferente de **static**
- **position: static**
 - Valor padrão
 - Elemento posicionado de acordo com fluxo normal do documento
- **position: relative**
 - Permite posicionar o elemento de maneira relativa à sua posição normal
 - O elemento é primeiramente posicionado de acordo com o fluxo normal. Em seguida, é deslocado da sua posição com **top**, **left**, **right** e **bottom**
 - O deslocamento não afeta a posição dos elementos à volta

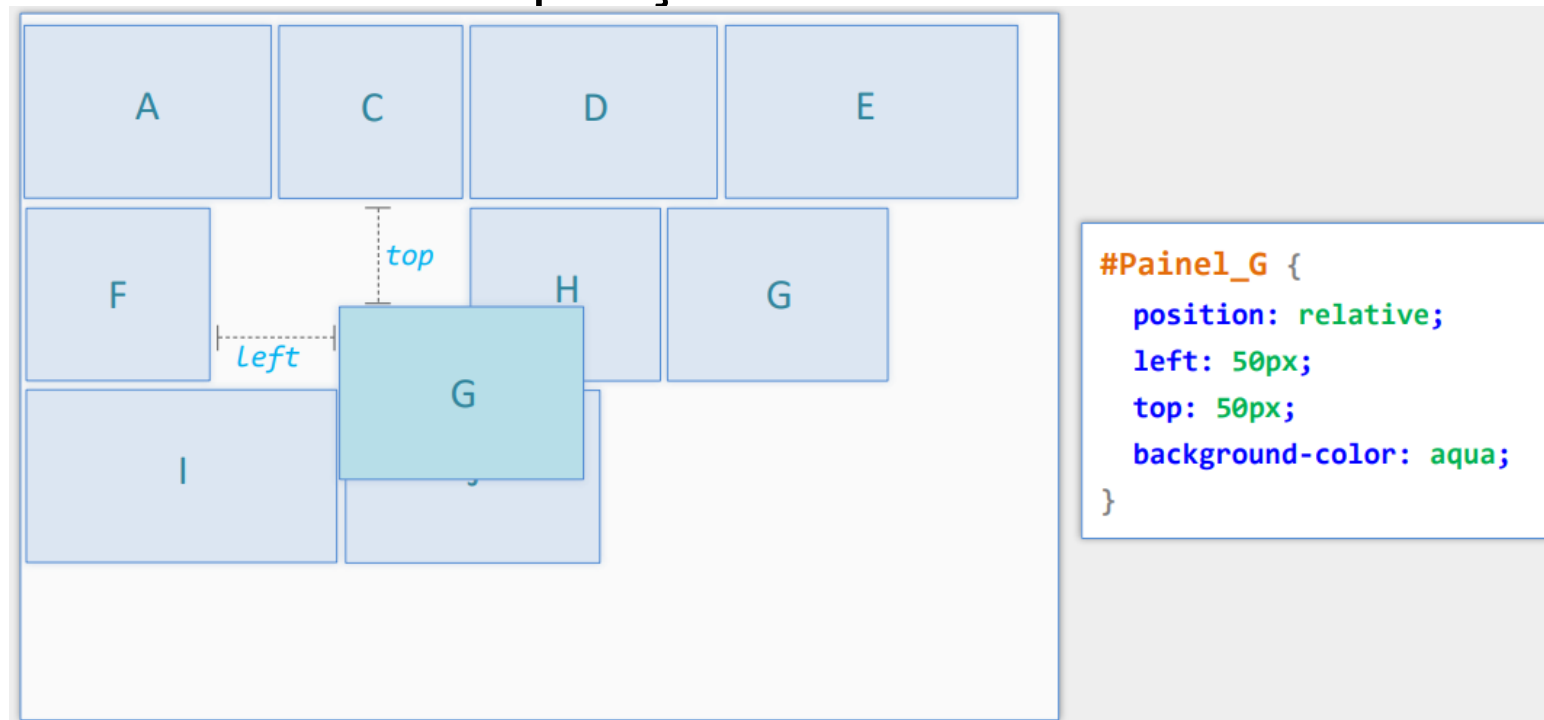
position: static

- Valor padrão
- Elemento posicionado de acordo com fluxo normal do documento



position: relative

- Permite posicionar o elemento de maneira relativa à sua posição normal
- O elemento é primeiramente posicionado de acordo com o fluxo normal. Em seguida, é deslocado da sua posição com **top**, **left**, **right** e **bottom**
- O deslocamento não afeta a posição dos elementos à volta



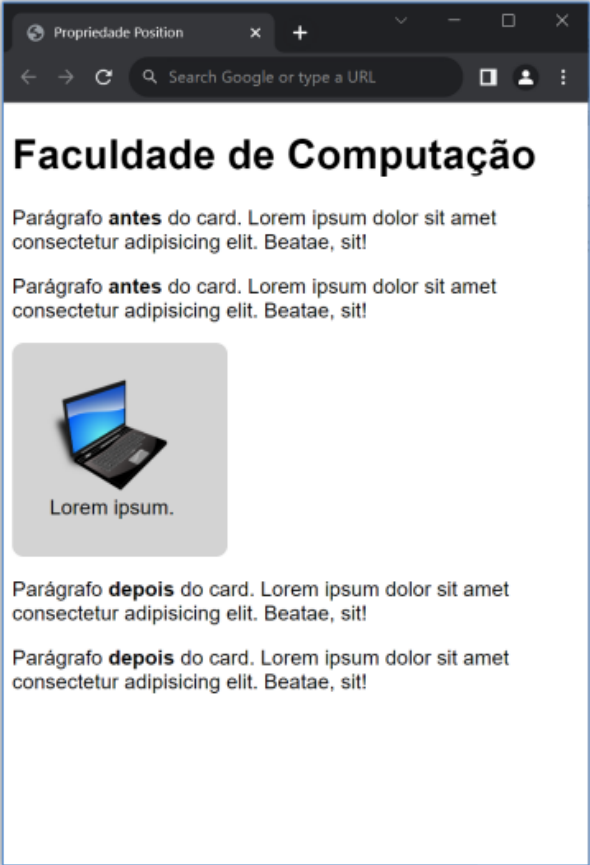
position: absolute

- O elemento é removido do fluxo normal
- **Não** ocupa espaço no layout
- Deve ser posicionado com **top**, **left**, **right** e **bottom**
- O posicionamento é relativo ao **ancestral mais próximo posicionado***, se houver
 - Caso contrário, o posicionamento é relativo ao elemento raiz (<html>)
- Pode ser utilizado para centralizar um elemento de bloco

** Elemento acima na hierarquia que tem a propriedade **position** com valor diferente de **static**.*

position: absolute


```
<head>
  <meta charset="UTF-8">
  <title>Propriedade Position</title>
  <style>
    #caixaCinza {
      width: 120px;
      border-radius: 10px;
      background-color: lightgray;
      padding: 2rem;
    }
  </style>
</head>
<body>
  <main>
    <h1>Faculdade de Computação</h1>
    <p>Parágrafo <b>antes</b> do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!</p>
    <p>Parágrafo <b>antes</b> do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!</p>
    <div id="caixaCinza">
      
      Lorem ipsum.
    </div>
    <p>Parágrafo <b>depois</b> do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!</p>
    <p>Parágrafo <b>depois</b> do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!</p>
  </main>
</body>
```



Faculdade de Computação

Parágrafo **antes** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

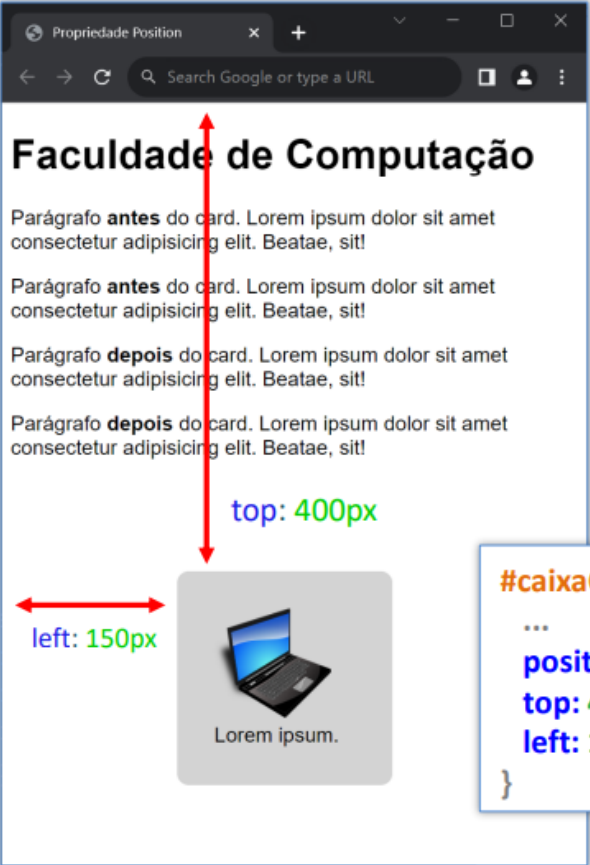
Parágrafo **antes** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!



Lorem ipsum.

Parágrafo **depois** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

Parágrafo **depois** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!



Faculdade de Computação

Parágrafo **antes** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

Parágrafo **antes** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

Parágrafo **depois** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

Parágrafo **depois** do card. Lorem ipsum dolor sit amet
consectetur adipisicing elit. Beatae, sit!

top: 400px

left: 150px

```
#caixaCinza {
  ...
  position: absolute;
  top: 400px;
  left: 150px;
}
```

A caixa em cinza acima é um elemento **div** com posicionamento padrão (**static**). Repare que há dois parágrafos antes e dois parágrafos depois. Veja o código HTML ao lado.

Neste exemplo o posicionamento da caixa é alterado para **absolute** e sua posição é ajustada com relação ao elemento raiz (<html>), pois nenhum outro ancestral tem **position ≠ static**. Observe que a caixa saiu do layout e os parágrafos ocuparam o seu lugar original.

position: fixed

- Elemento posicionado com relação à *viewport* utilizando **top**, **left**, **right** e **bottom**
- A posição do elemento não se altera com a rolagem da página
- O elemento é removido do fluxo normal – não ocupa espaço no layout
- Quando impresso, aparecerá na mesma posição em todas as páginas
- Geralmente utilizado quando se desejar posicionar algo de maneira fixa na tela, como painéis de avisos sobre cookies, privacidade etc.

position: sticky

- Elemento posicionado de acordo com fluxo normal do documento
- Ocupa espaço no layout
- Elemento "gruda" no primeiro ancestral com mecanismo de rolagem
- Normalmente é utilizado em conjunto com `top: 0`

position: sticky

Prof. Daniel A. Furtado

INÍCIOENSINOPROJETOSPUBLICAÇÕES

Programação para Internet

A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Barra de navegação definida com `position: sticky`. Página sem rolagem.

INÍCIOENSINOPROJETOSPUBLICAÇÕES

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Serão aplicadas três avaliações práticas, um projeto de implementação e vários testes de aula. As avaliações práticas devem ser realizadas em horário de aula, sob supervisão do professor. O projeto de implementação deverá ser apresentado pela equipe no final do semestre letivo, conforme cronograma disponibilizado pelo professor.

A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Mesma página sendo exibida após rolagem da tela (nav 'gruda' no topo)

Propriedade `transform`

- Permite mover, rotacionar, torcer e escalonar um elemento
- Exemplos:
 - `transform: translateX(50px);` move o elemento 50px na horizontal
 - `transform: translateY(50px);` move o elemento 50px na vertical
 - `transform: translate(30px,10px);` move 30px na horizontal e 10px na vertical
 - `transform: rotate(45deg);` rotaciona o elemento em 45 graus
 - `transform: scale(2);` dobra o tamanho nas duas direções

Centralização na horizontal com `width` e `margin`

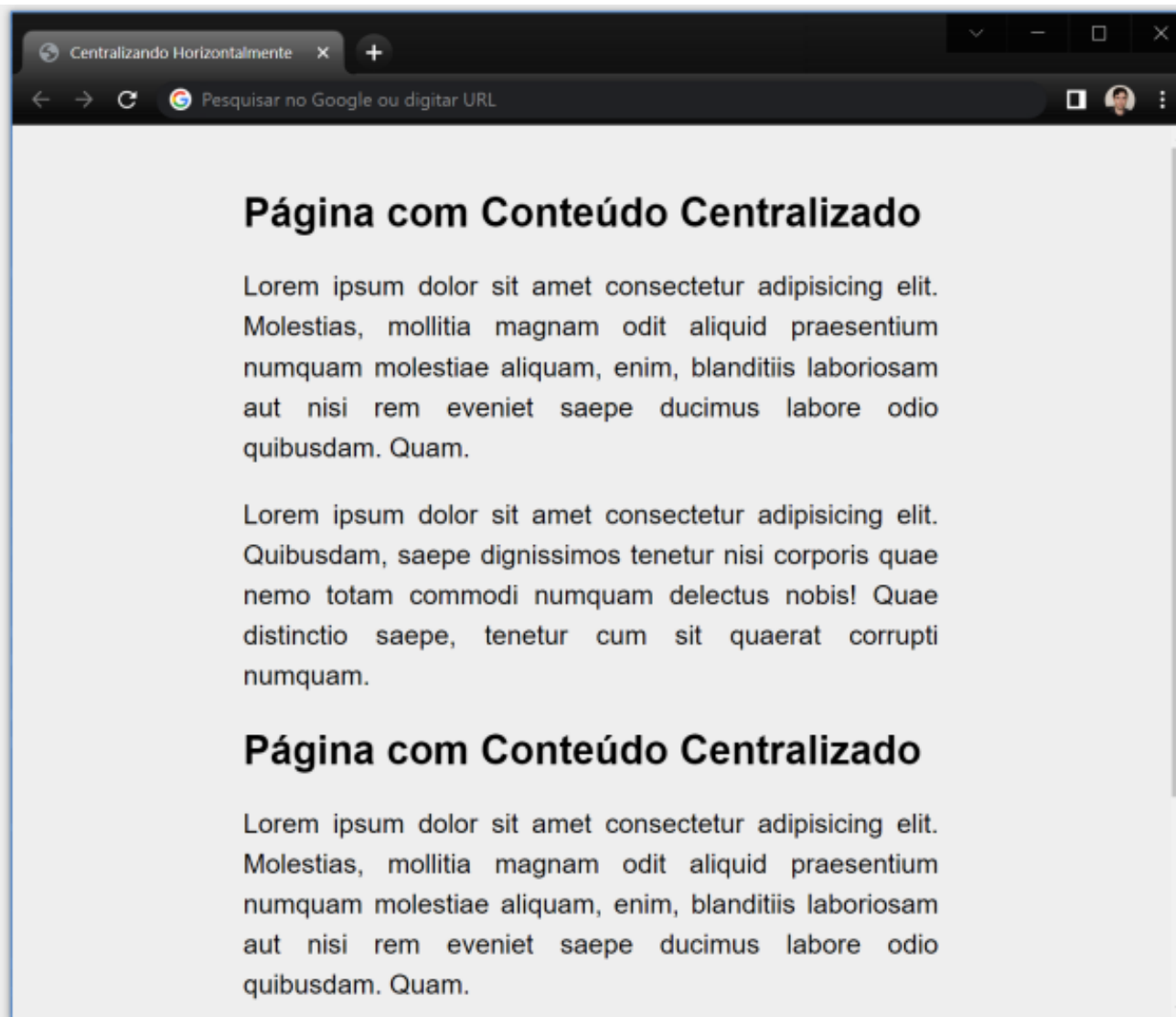
- Uma forma de centralizar horizontalmente um elemento de bloco, sem removê-lo do layout, é utilizando as propriedades `width` e `margin`
- Define-se uma largura com `width` e coloca-se as margens laterais em `auto`
- Com as margens laterais em `auto`, o navegador ajustará os valores igualmente para preencher o espaço, resultando na centralização
- Para centralizar apenas o texto dentro de um elemento, pode-se utilizar `text-align: center;`

Centralização na horizontal com **width** e **margin**

```
<style>
  body {
    text-align: justify;
    width: 60%;
    margin: 40px auto;
  }
</style>
</head>

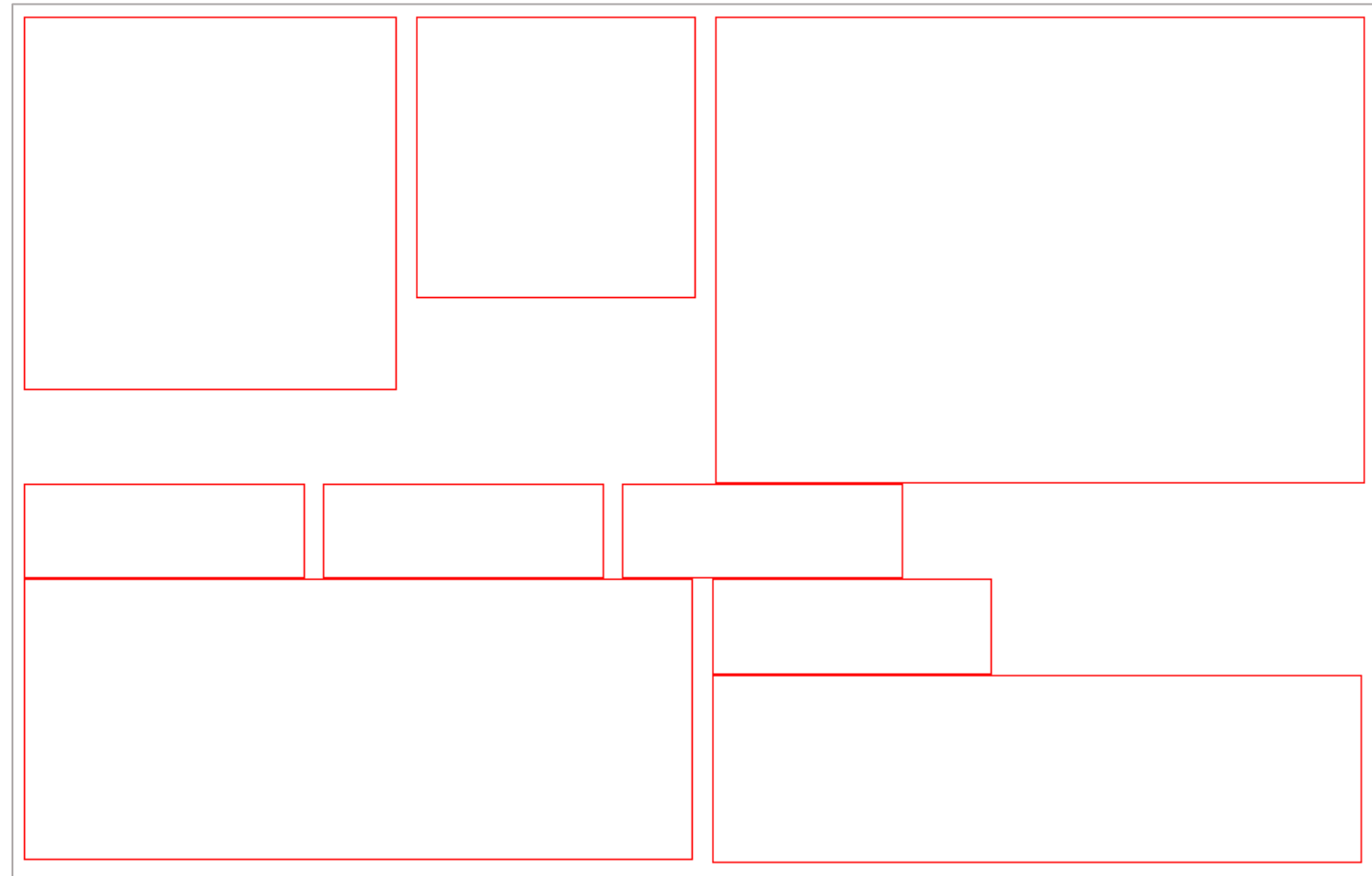
<body>
  <h2>Página com Conteúdo Centralizado</h2>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit.
    molestiae aliquam, enim, blanditiis laboriosam
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    nemo totam commodi numquam delectus nobis! Quae
    distinctio saepe, tenetur cum sit quaerat corrupti
    numquam.

  <h2>Página com Conteúdo Centralizado</h2>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit.
    molestiae aliquam, enim, blanditiis laboriosam
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    nemo totam commodi numquam delectus nobis! Quae
    distinctio saepe, tenetur cum sit quaerat corrupti
    numquam.
</body>
```



Exercício

- Implemente uma página que contém 9 <div>
 - Com tamanhos variados
 - Com borda
 - Utilize `float: left` para posicionamento
- Crie uma regra CSS do tipo ID para cada uma
- Crie uma regra CSS do tipo classe compartilhada entre todas



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>