



# Programação para Internet

---

Módulo 2 – Parte 2

Introdução à Linguagem CSS

Prof. Dr. Daniel A. Furtado - FACOM/UFU

Conteúdo protegido por direito autoral, nos termos da Lei nº 9 610/98

A cópia, reprodução ou apropriação deste material, total ou parcialmente, é proibida pelo autor

# Conteúdo

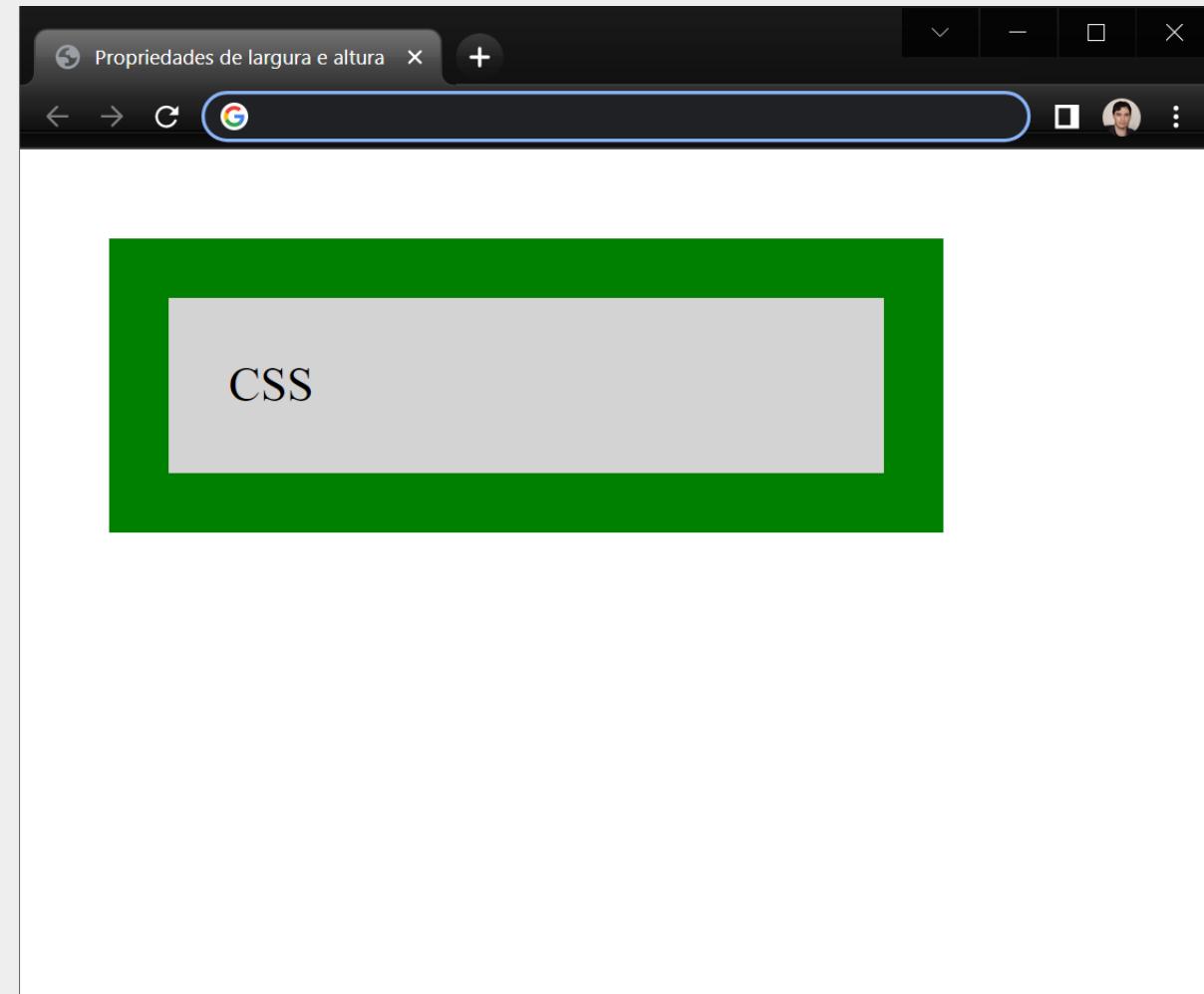
- Exibição e posicionamento de elementos
- Centralização de Elementos
- Pseudo-Classes
- Pseudo-Elementos
- Cascade, Especificidade e Herança
- Introdução ao Design Responsivo
- Introdução à Media Queries
- Dicas e Validação

# Ajustes de Largura e Altura dos Elementos

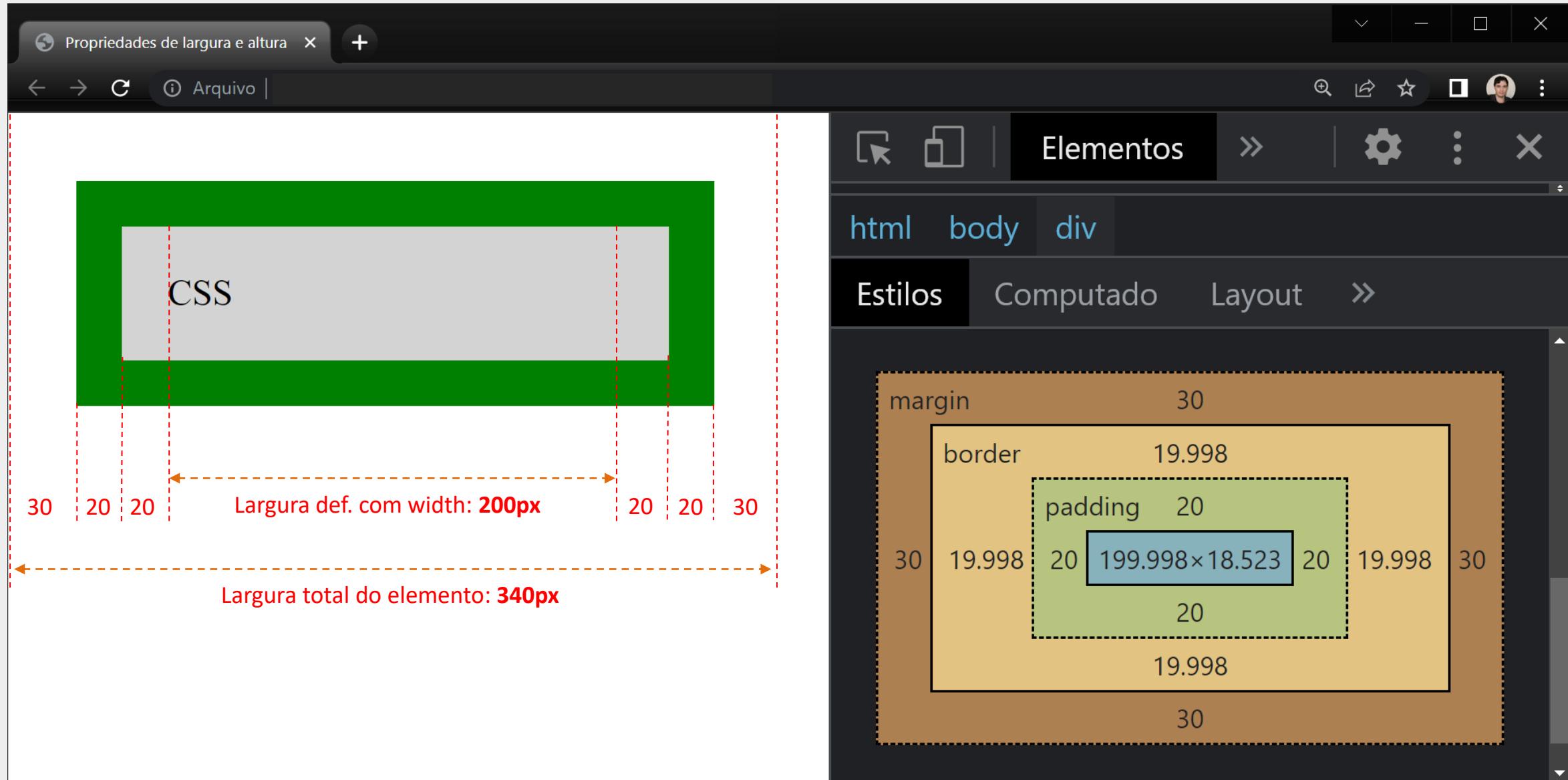
- Propriedades para definir, respectivamente, a largura, altura, mínima largura, mínima altura, máxima largura e máxima altura de um elemento:
  - `width`, `height`,
  - `min-width`, `min-height`,
  - `max-width`, `max-height`
- Observações
  - Na maior parte dos casos, definem tamanhos para a **região de conteúdo** do elemento, sem contabilizar margens, bordas e paddings (porém, esse comportamento pode ser alterado com a propriedade `box-sizing`)
  - Valores em pixels, porcentagem, `em`, `rem`, etc.
  - Outros valores: `auto`, `max-content`, `min-content`, `fit-content`

# Ajuste de largura - Exemplo

```
8  <style>
9      body { margin: 0 }
10     div {
11         background-color: lightgray;
12         width: 200px;
13         margin: 30px;
14         padding: 20px;
15         border: 20px solid green;
16     }
17 </style>
18 </head>
19
20 <body>
21     <div>
22         CSS
23     </div>
24 </body>
```



# Largura Total do Elemento – Cálculo Padrão

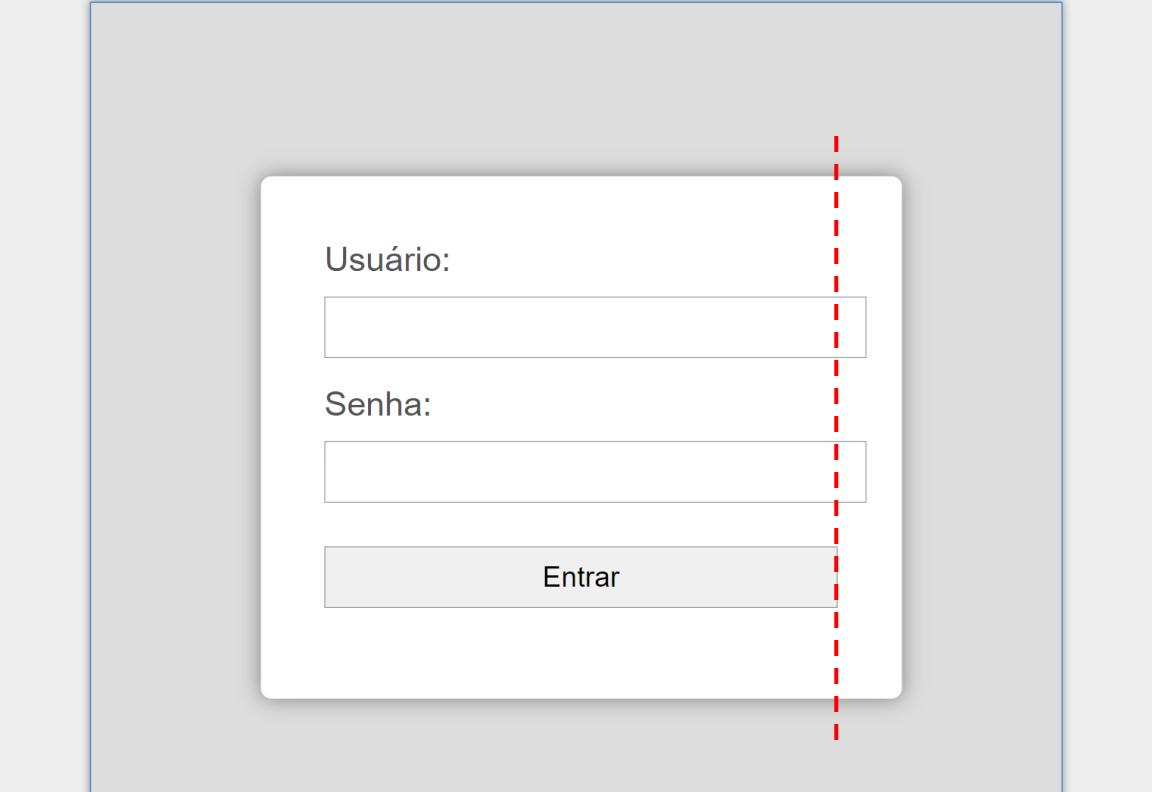


# Propriedade box-sizing

- Altera o modo em que a largura/altura total do elemento é calculada
- **box-sizing: content-box**
  - Valor padrão para a maioria dos elementos
  - Cálculo da largura/altura total realizado conforme mostrado no slide anterior
  - Ex.: a largura total será o valor definido com `width` + bordas, paddings e margens
- **box-sizing: border-box**
  - Cálculo da largura/altura inclui tamanhos de bordas e paddings (**mas não margens**)
  - Ex.: ao definir a largura com `width`, estamos definindo, na verdade, a largura do conteúdo com a borda e o padding. Portanto, a largura do conteúdo em si poderá ser menor para que a borda e o padding entrem no cálculo.

# Propriedade box-sizing - Exemplo

```
main {  
    width: 300px;  
    padding: 30px;  
    background-color: white;  
}  
  
input, button {  
    width: 100%;  
    padding: 0.4rem;  
    outline: none;  
    border: 0.5px solid gray;  
}
```



Os campos aparecem desalinhados do botão porque os campos textuais utilizam, por padrão, `box-sizing: content-box`, enquanto o botão utiliza `box-sizing: border-box`.

# Propriedade box-sizing - Exemplo

```
main {  
    width: 300px;  
    padding: 30px;  
    background-color: white;  
}  
  
input, button {  
    width: 100%;  
    padding: 0.4rem;  
    outline: none;  
    border: 0.5px solid gray;  
    box-sizing: border-box;  
}
```



Usuário:

Senha:

Entrar

Uma solução, neste exemplo, é definir `box-sizing: border-box` para os campos textuais.

# 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

## display: block

- exibição em nível de bloco (como o `<div>`)
- começará em nova linha e ocupará, por padrão, toda a largura

# Propriedade display

## display: inline

- exibição em nível de linha (como o `<span>`)
- `width` e `height` não terão efeito em alguns elementos
- margens e paddings superiores e inferiores de alguns elementos **não são respeitados**

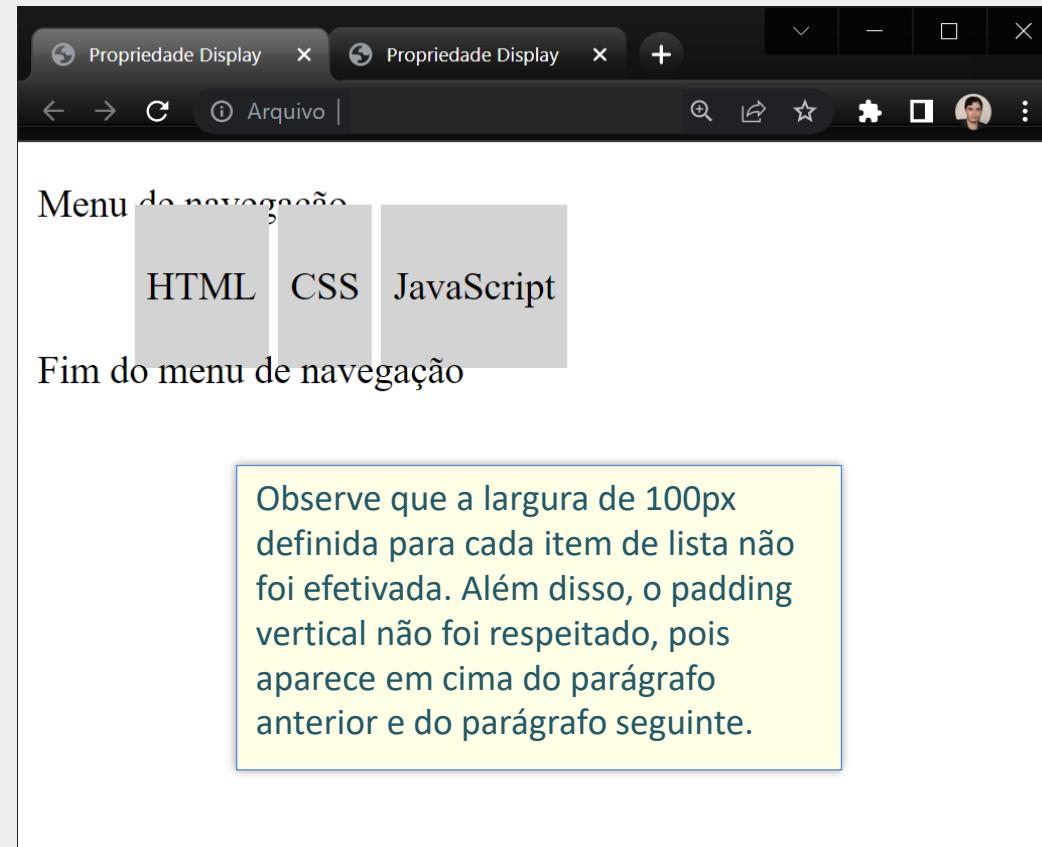
## display: inline-block

- exibição em nível de linha
- possibilidade de usar `width` e `height`
- margens e paddings superiores e inferiores respeitados

# Item de Lista com display: inline

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

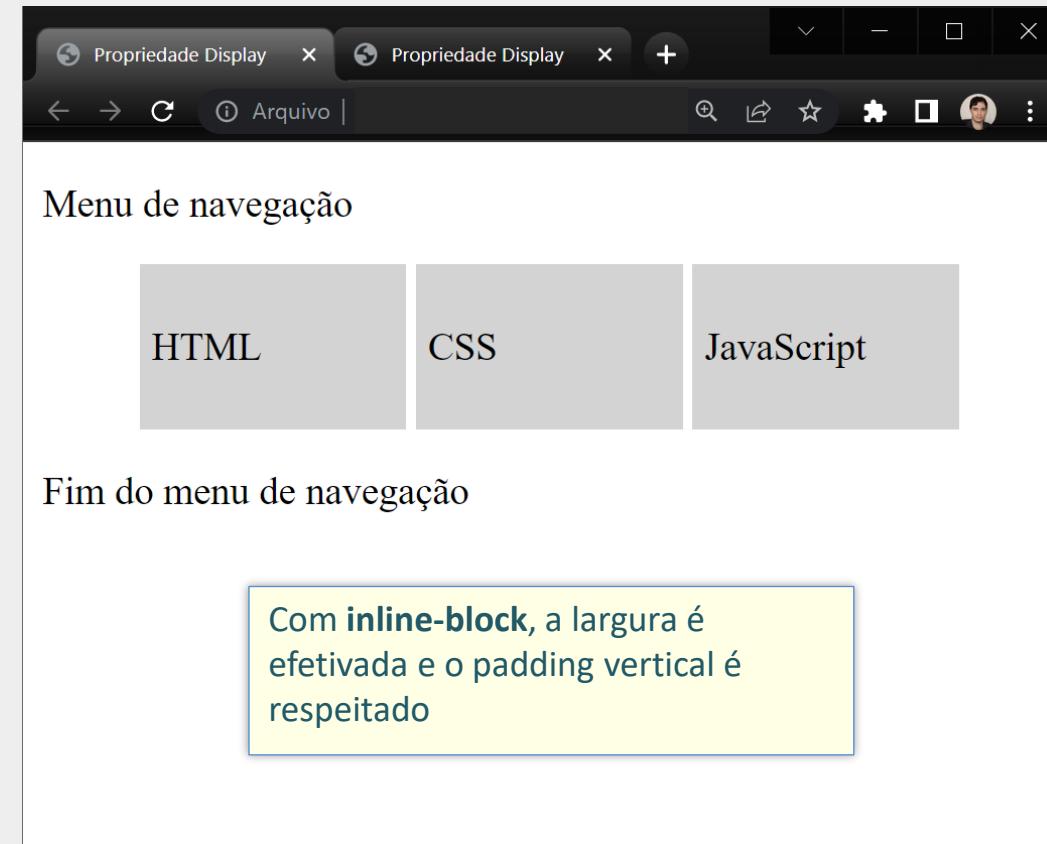
<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>
  </nav>
</body>
```



# Item de Lista com display: inline-block

```
<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
- Valores possíveis
  - **visible**      elemento visível
  - **hidden**     elemento oculto, mas **continua ocupando espaço no layout**
  - **collapse**
    - comumente usado para ocultar linhas/colunas de tabelas, sem ocupar espaço no layout
    - O espaço é removido **sem alterar os tamanhos das demais linhas e colunas** (veja exemplo no próximo slide)

```


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


```

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>

```


| Media da Turma | 6,5 | 8,5 |
|----------------|-----|-----|
|----------------|-----|-----|


```

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>

```


| Media da Turma | 6,5 | 8,5 |
|----------------|-----|-----|
|----------------|-----|-----|


```

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

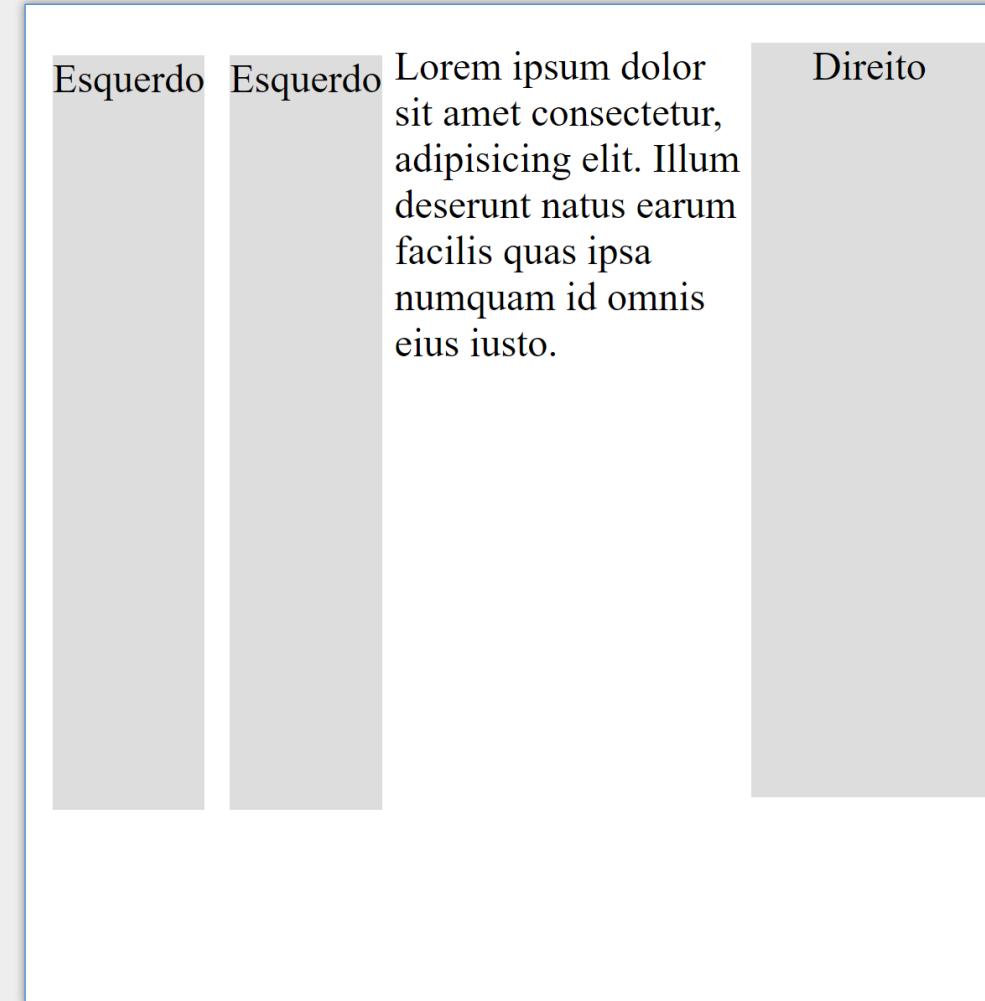
Última linha ocultada com **display: none**. Repare que as larguras das colunas são reajustadas conforme conteúdo (col. **Aluno**)

# 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 - Exemplo

```
<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 elem. 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 apenas se necessário

# Propriedade overflow - Exemplo

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

**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

`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

`overflow: auto;`

# Propriedade position

- Define como o elemento é posicionado na página
- Normalmente é utilizada em conjunto com `top`, `left`, `right` e `bottom`
- Valores possíveis
  - `static`
  - `relative`
  - `absolute`
  - `fixed`
  - `sticky`
- O elemento é dito **posicionado** quando `position` tem valor diferente de `static`

# Propriedade position

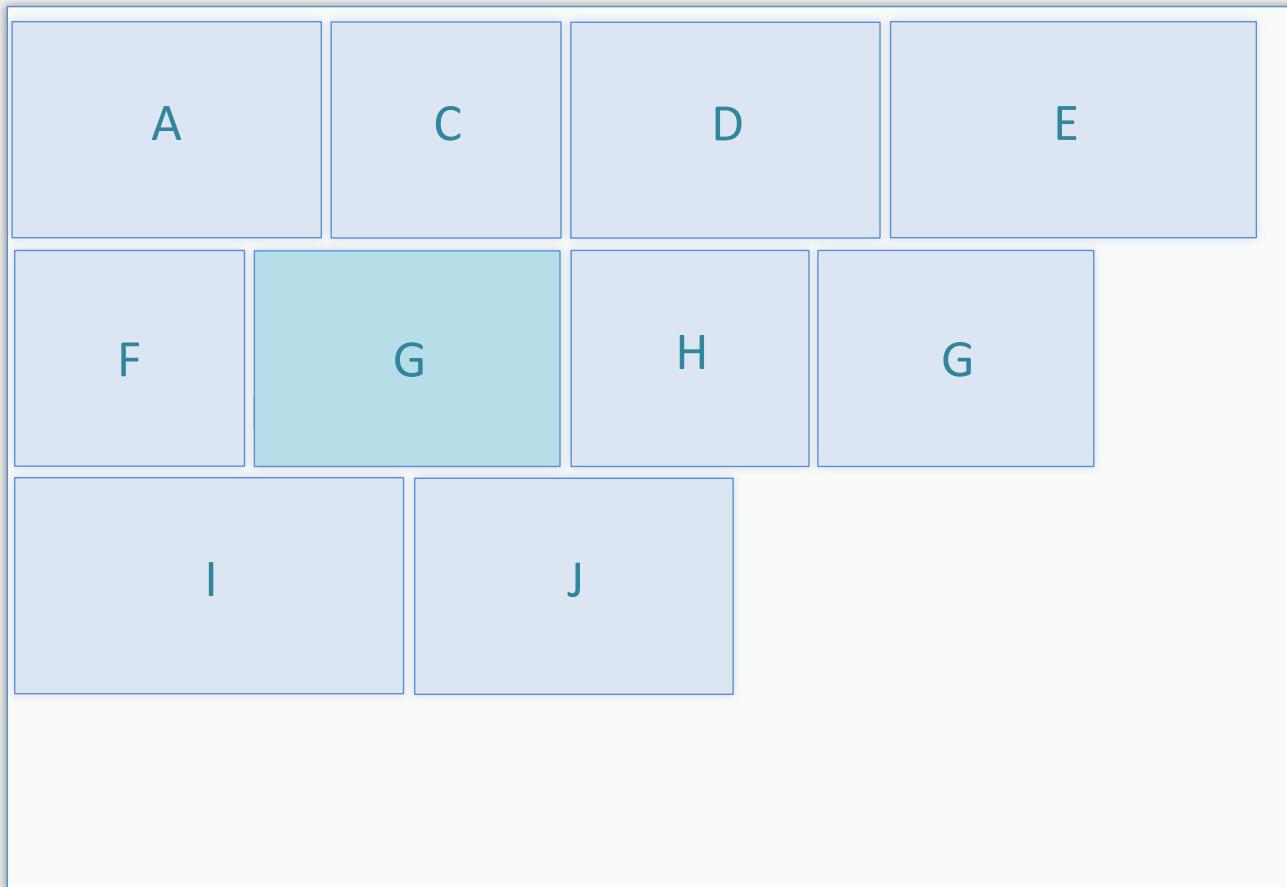
## position: static

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

## position: relative

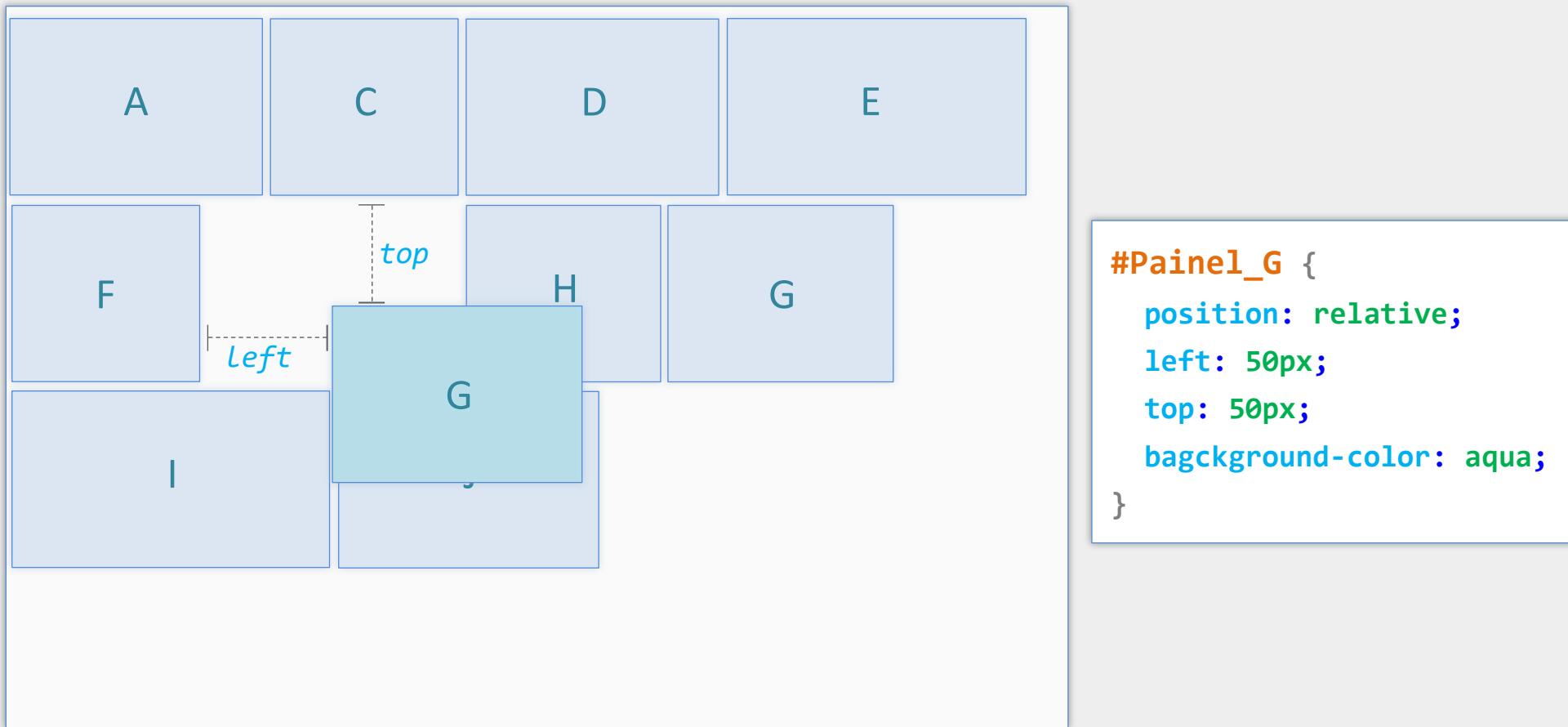
- Elemento primeiramente posicionado de acordo com o fluxo normal
- Em seguida, é deslocado da sua posição com `top`, `left`, `right` e `bottom`
- O deslocamento **não afeta a posição** dos demais elementos
  - Layout geral é o mesmo do posicionamento `static`

# Exemplo de position: static



```
#Painel_G {  
    position: static;  
    background-color: aqua;  
}
```

# Exemplo de position: relative



# Propriedade position - continuação

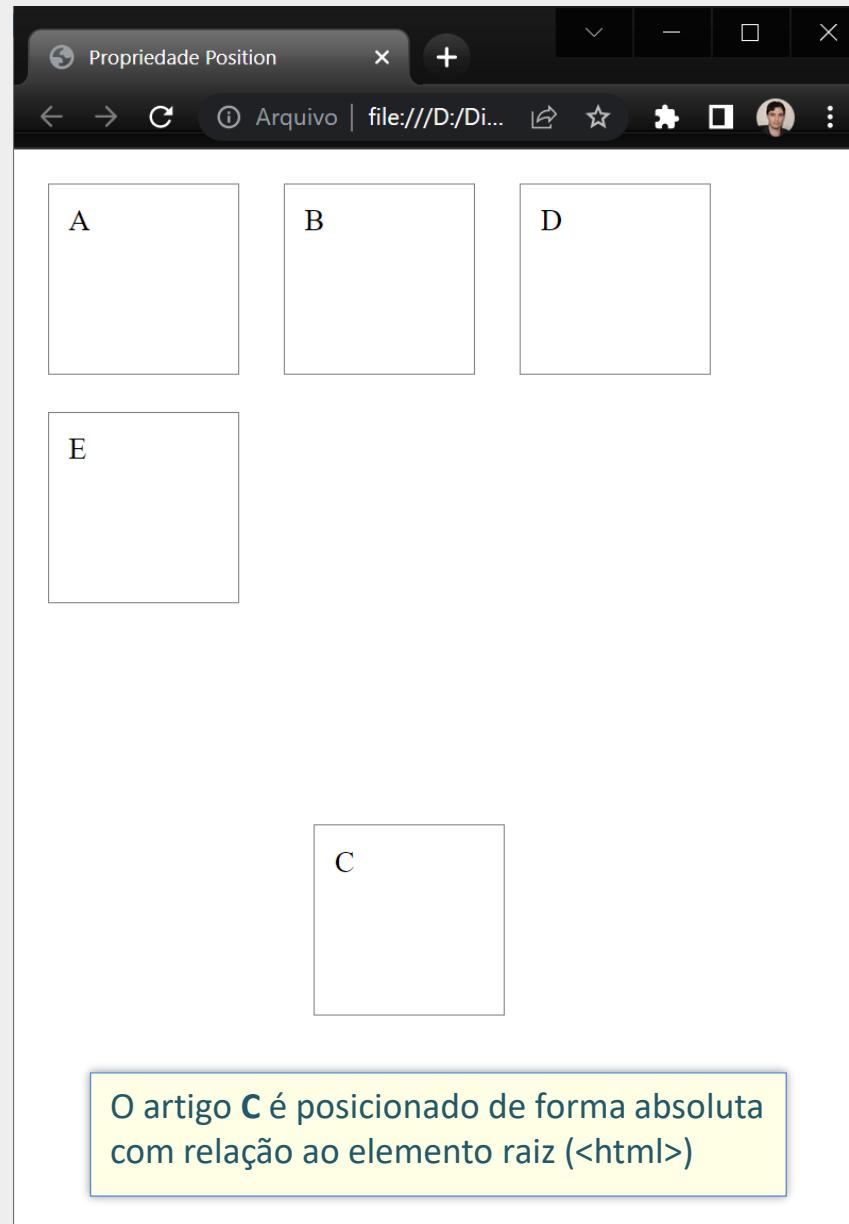
## position: absolute

- Elemento removido do fluxo normal
- Nenhum espaço no layout é criado para o elemento
- Posicionamento com `top`, `left`, `right` e `bottom` relativo ao:
  - Ancestral mais próximo posicionado, se houver
  - Caso contrário, com relação ao container inicial (elemento `<html>`)
- Pode ser utilizado para centralizar um elemento de bloco

# Exemplo de position: absolute

```
<style>
  article {
    margin: 10px; padding: 10px;
    border: 1px solid gray;
    width: 80px; height: 80px;
    display: inline-block;
  }

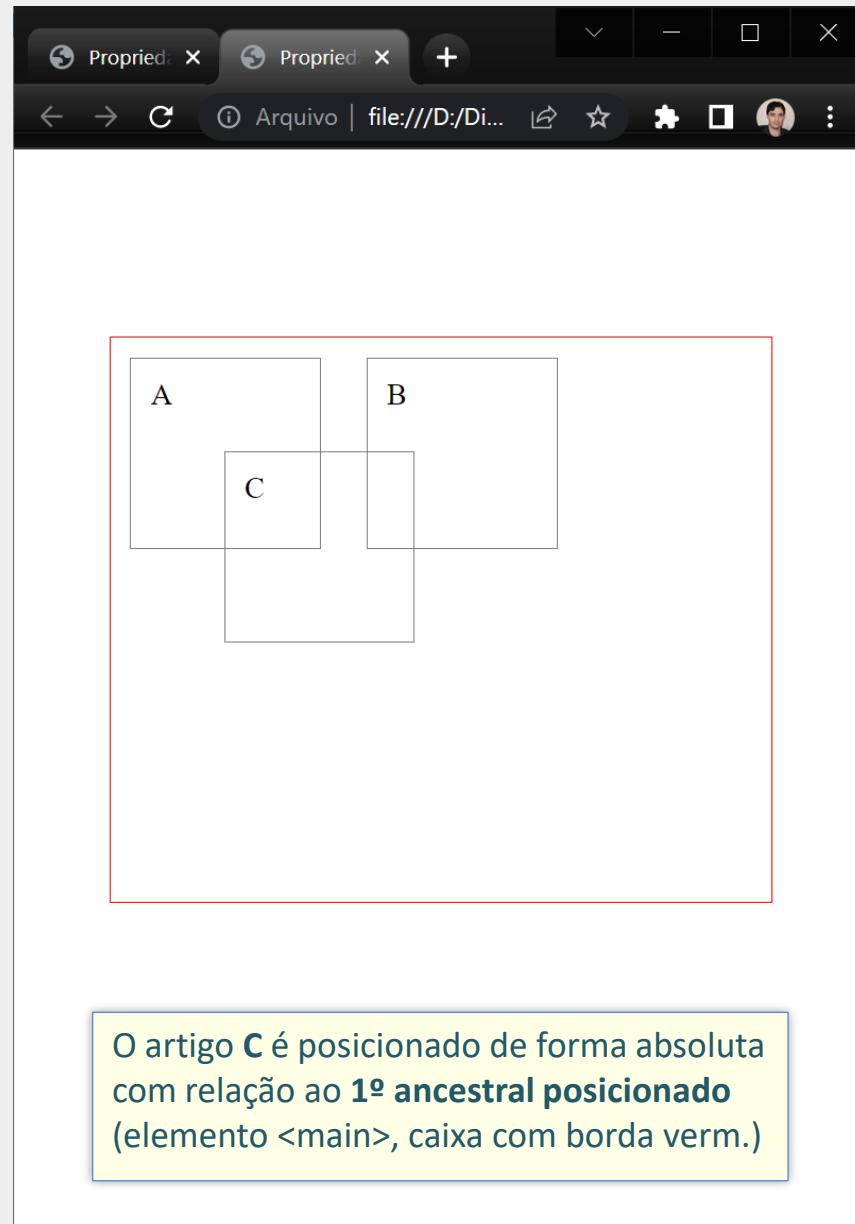
  .posAbs {
    position: absolute;
    left: 150px;
    top: 350px;
  }
</style>
</head>
<body>
  <main>
    <article>A</article>
    <article>B</article>
    <article class="posAbs">C</article>
    <article>D</article>
    <article>E</article>
  </main>
</body>
```



O artigo C é posicionado de forma absoluta com relação ao elemento raiz (<html>)

# Exemplo de position: absolute

```
<style>
  main {
    width: 80%; height: 300px;
    margin: 100px auto;
    border: 1px solid red;
    position: relative;
  }
  article {
    margin: 10px; padding: 10px;
    border: 1px solid gray;
    width: 80px; height: 80px;
    display: inline-block;
  }
  .posAbs {
    position: absolute;
    left: 50px;
    top: 50px;
  }
</style>
</head>
<body>
  <main>
    <article>A</article>
    <article>B</article>
    <article class="posAbs">C</article>
  </main>
</body>
```



# Propriedade position - continuação

## position: fixed

- Elemento posicionado com relação à **viewport** (janela do navegador)
- Posição **não se altera com a rolagem** da página
- Posicionamento com as propriedades **top, left, right e bottom**
- Elemento **removido do fluxo normal** - nenhum espaço é criado no layout
- Quando impresso, aparecerá na mesma posição em todas as páginas

## 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**

Exemplos disponíveis em <https://youtu.be/caJ7Q65aiLE?t=2200>

# Exemplo de position: sticky

The screenshot shows a website layout with a fixed navigation bar at the top. The navigation bar contains the following items: 'INÍCIO' (Home), 'ENSINO' (Teaching), 'PROJETOS' (Projects), and 'PUBLICAÇÕES' (Publications). The main content area features a title 'Prof. Daniel A. Furtado' and a sub-section titled 'Programação para Internet'. Below this, there are two large text blocks. The first text block discusses the discipline's objectives and its focus on Web application development. The second text block details specific goals related to Web programming paradigms and dynamic websites. At the bottom of the page, there are two descriptive captions: one for the initial state and one for the state after scrolling.

**Prof. Daniel A. Furtado**

INÍCIO ENSINO PROJETOS PUBLICAÇÕES

**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.

INÍCIO ENSINO PROJETOS PUBLICAÇÕES

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.

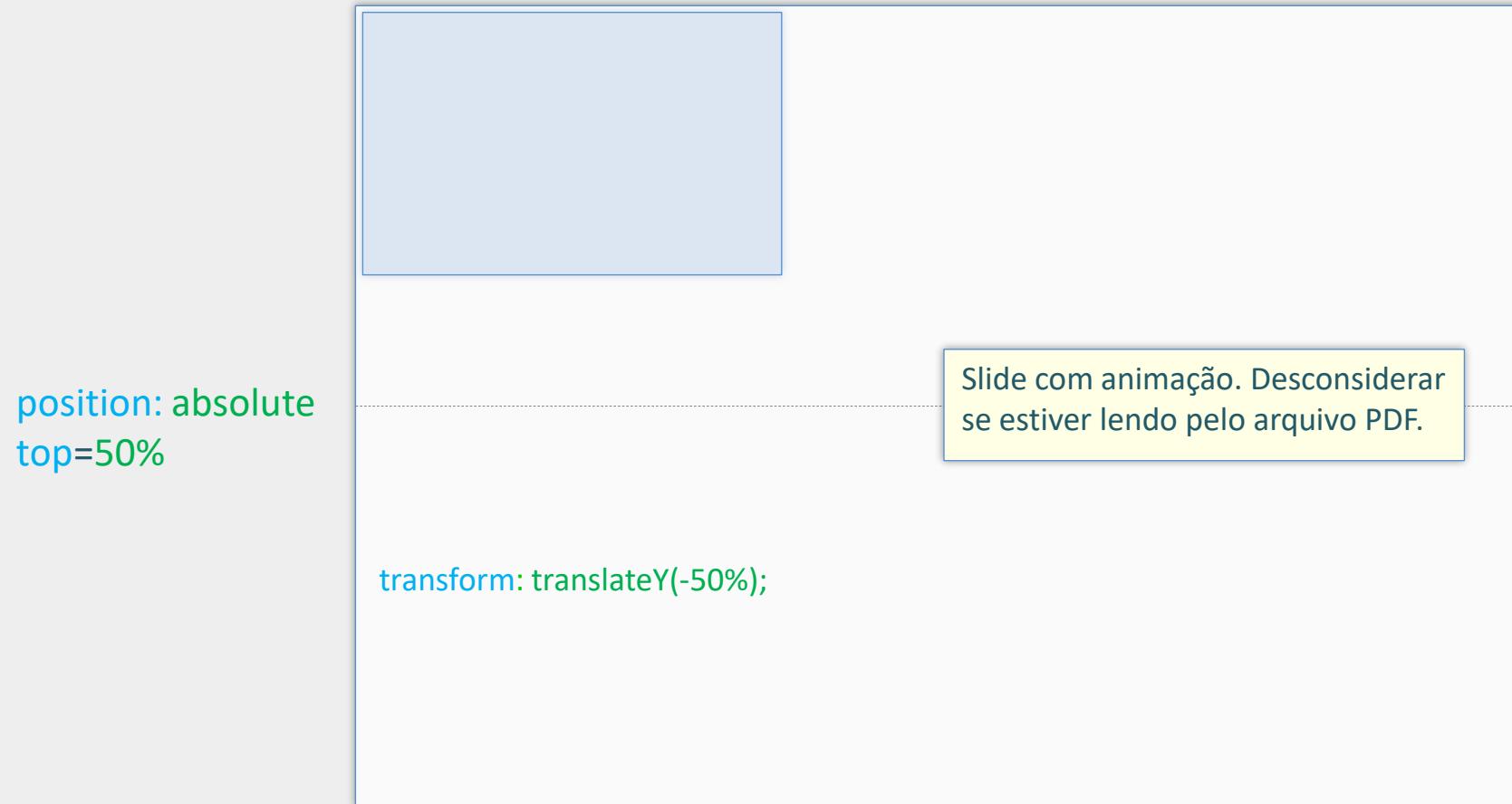
Mesma página sendo exibida após rolagem da tela (nav 'gruda' no topo)

# Propriedade transform

- Permite mover, rotacionar, torcer e escalar um elemento
- Exemplos:
  - `transform: translateX(50px);` move o elemento 50 pixels em x (hor.)
  - `transform: translateY(50px);` move o elemento 50 pixels em y (vert.)
  - `transform: translate(30px,10px);` move o elemento 30 pixels em x e 10 em y
  - `transform: rotate(45deg);` rotaciona o elemento em 45 graus
  - `transform: scale(2);` dobra o tamanho nas duas direções

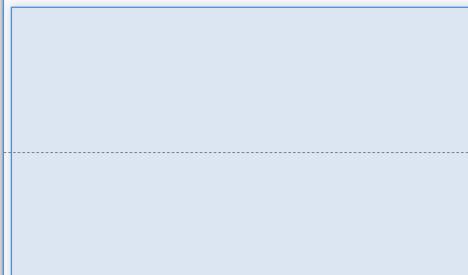
Há muitas outras formas de uso

# Centralizando Bloco na Vertical com Pos. Absoluto



# Centralizando Bloco na Vertical com Pos. Absoluto

position: absolute  
top=50%



transform: translateY(-50%);

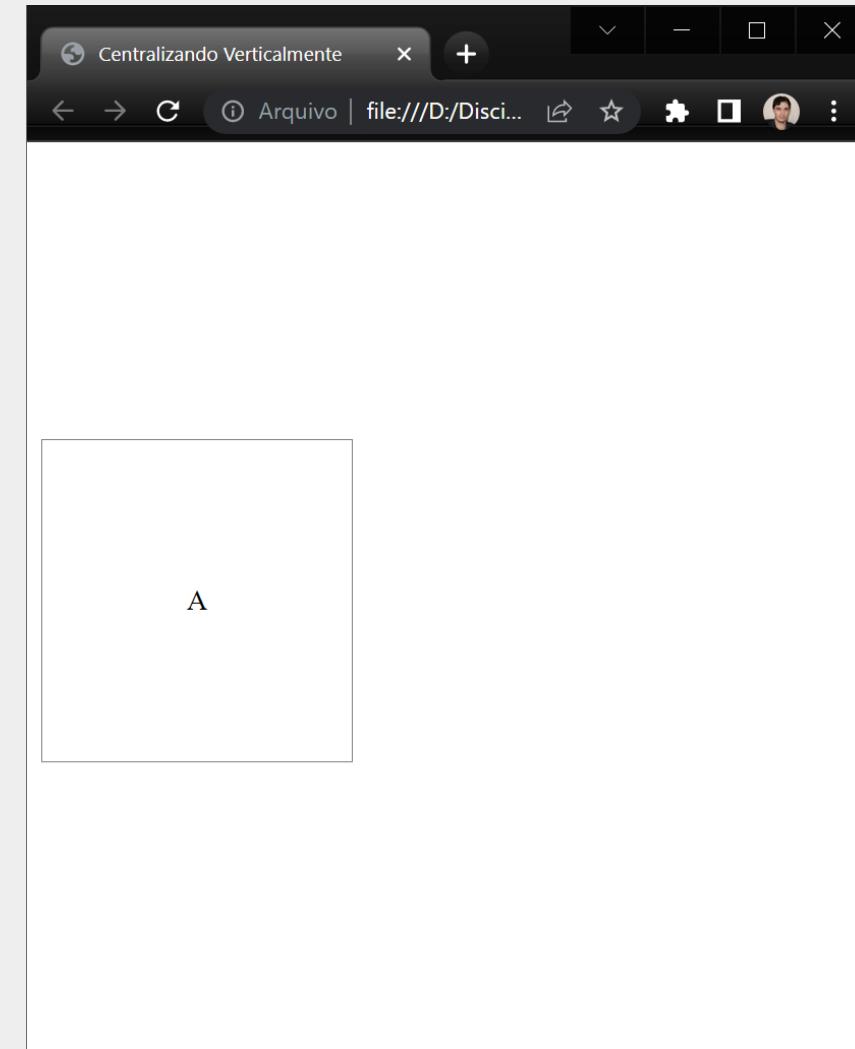
# Centralizando Bloco na Vertical com Pos. Absoluto

```
<style>
  main {
    padding: 80px;
    border: 1px solid gray;

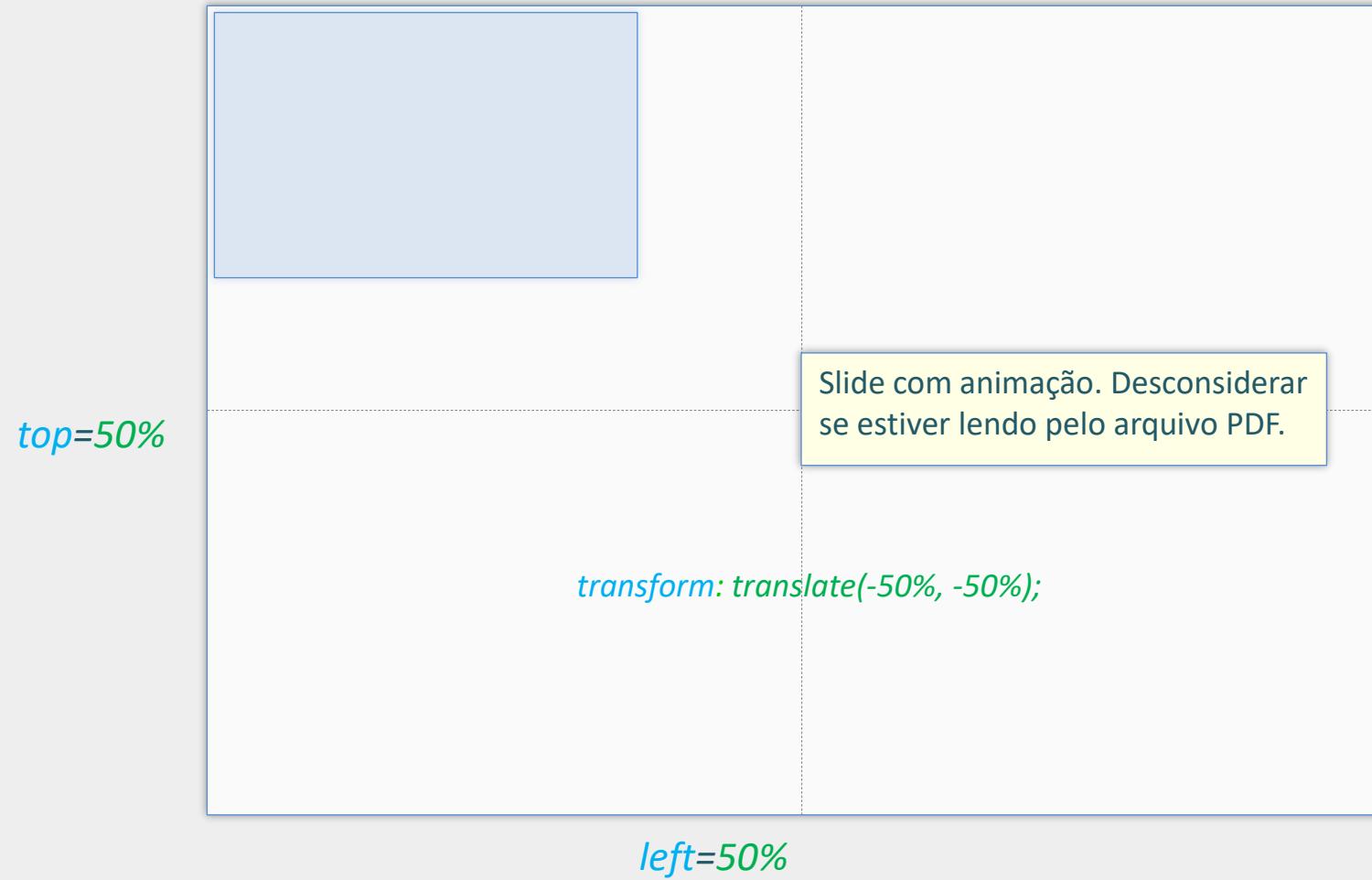
    position: absolute;
    top: 50%;
    transform: translateY(-50%);

  }
</style>
</head>

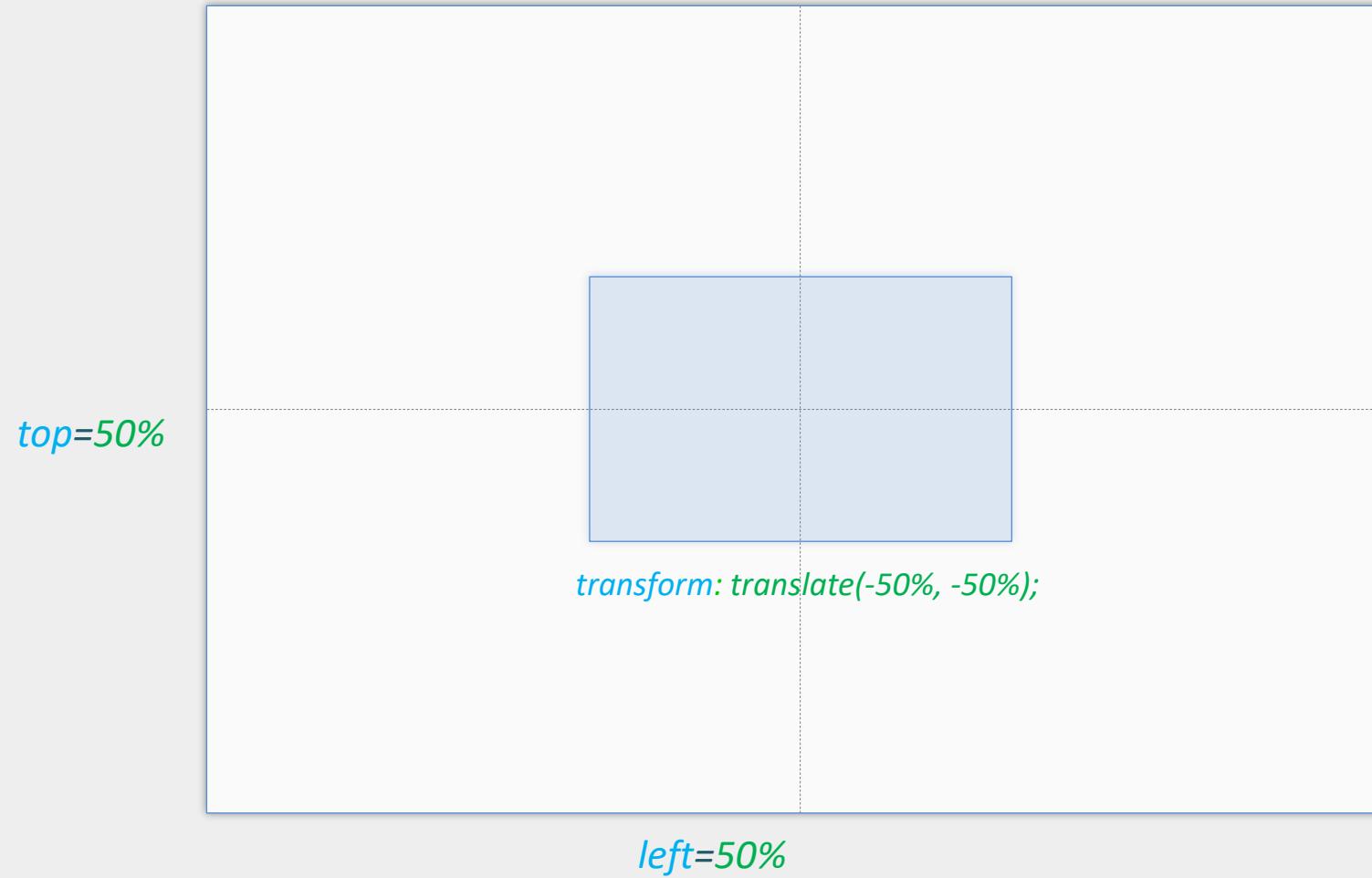
<body>
  <main>A</main>
</body>
```



# Centralizando na Vertical e Horizontal com Pos. Absoluto



# Centralizando na Vertical e Horizontal com Pos. Absoluto

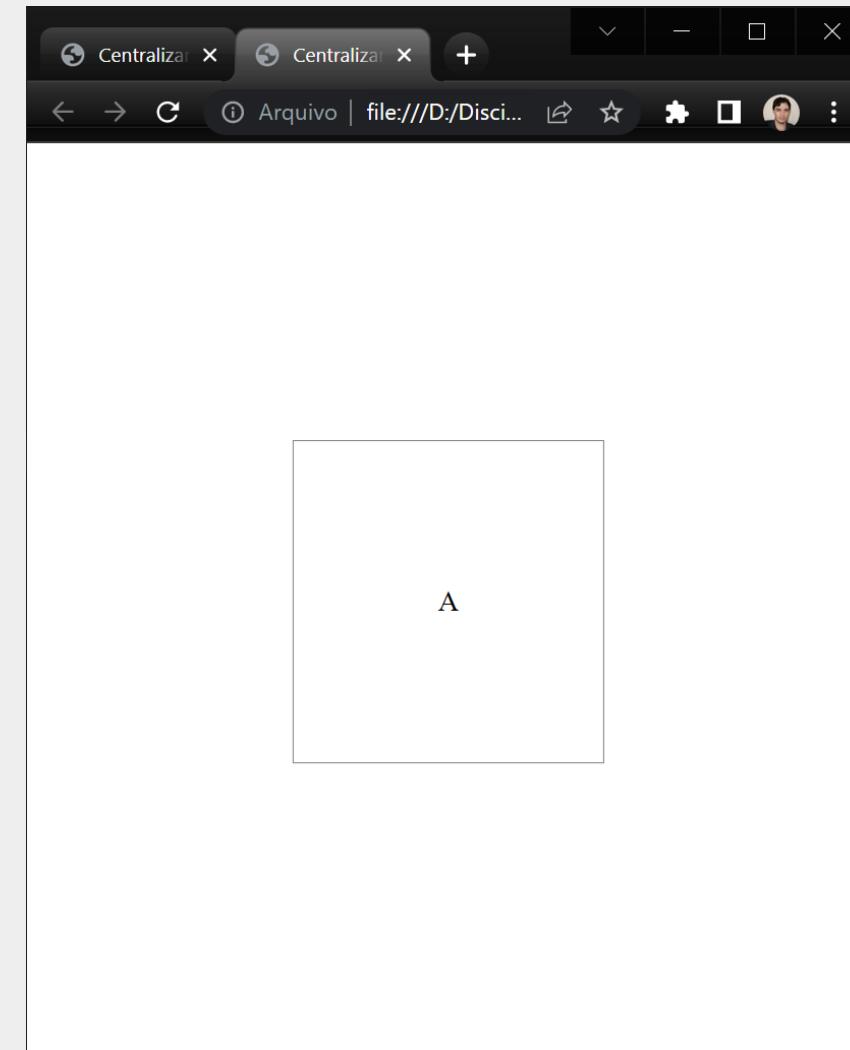


# Centralizando na Vertical e Horizontal com Pos. Absoluto

```
<style>
  main {
    padding: 80px;
    border: 1px solid gray;

    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
</style>
</head>

<body>
  <main>A</main>
</body>
```



# Centralizando na Vertical e Horizontal com Pos. Absoluto

```
main {  
    width: 50%;  
    background-color: #eee;  
    border: 0.5px solid gray;  
    margin: 0;  
    padding: 2% 4%;  
  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
}
```



## A FACOM

A Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia foi criada em 2000, a partir do extinto Departamento de Informática (DEINF), criado em 1988, no âmbito do CETEC.

No início dos anos 2000 foi criado na Faculdade o Programa de Pós-Graduação em Ciência da Computação.

# Centralizando na Horizontal com width e margin

- Define-se uma largura com `width`
- Coloca-se as margens laterais no automático (`auto`)
- Exemplo:

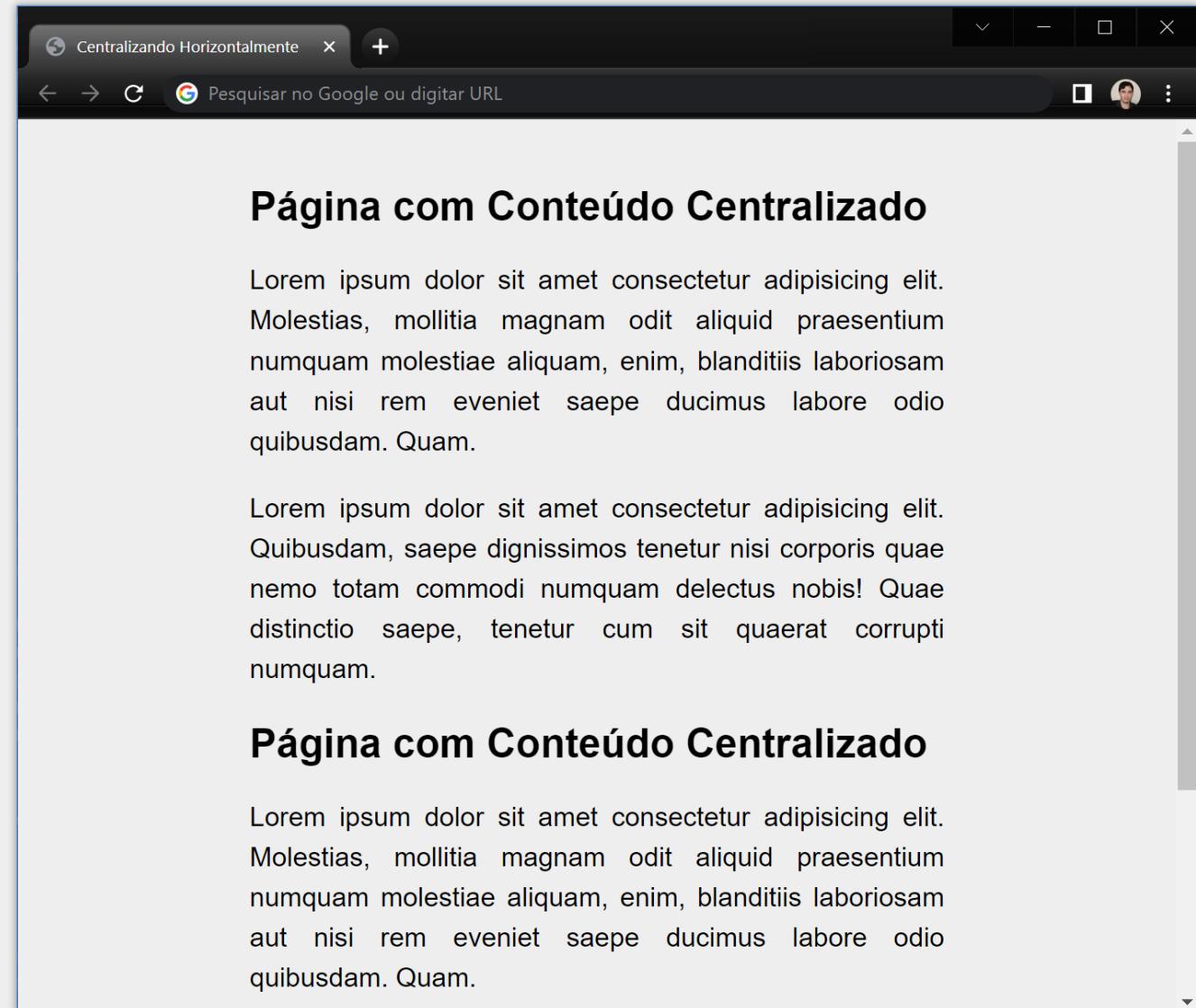
```
body {  
    width: 60%;  
    margin: 0 auto;  
}
```

**OBS:** para centralizar apenas o texto dentro de um elemento, utilize:  
`text-align: center;`

# Centralizando na Horizontal com width e margin

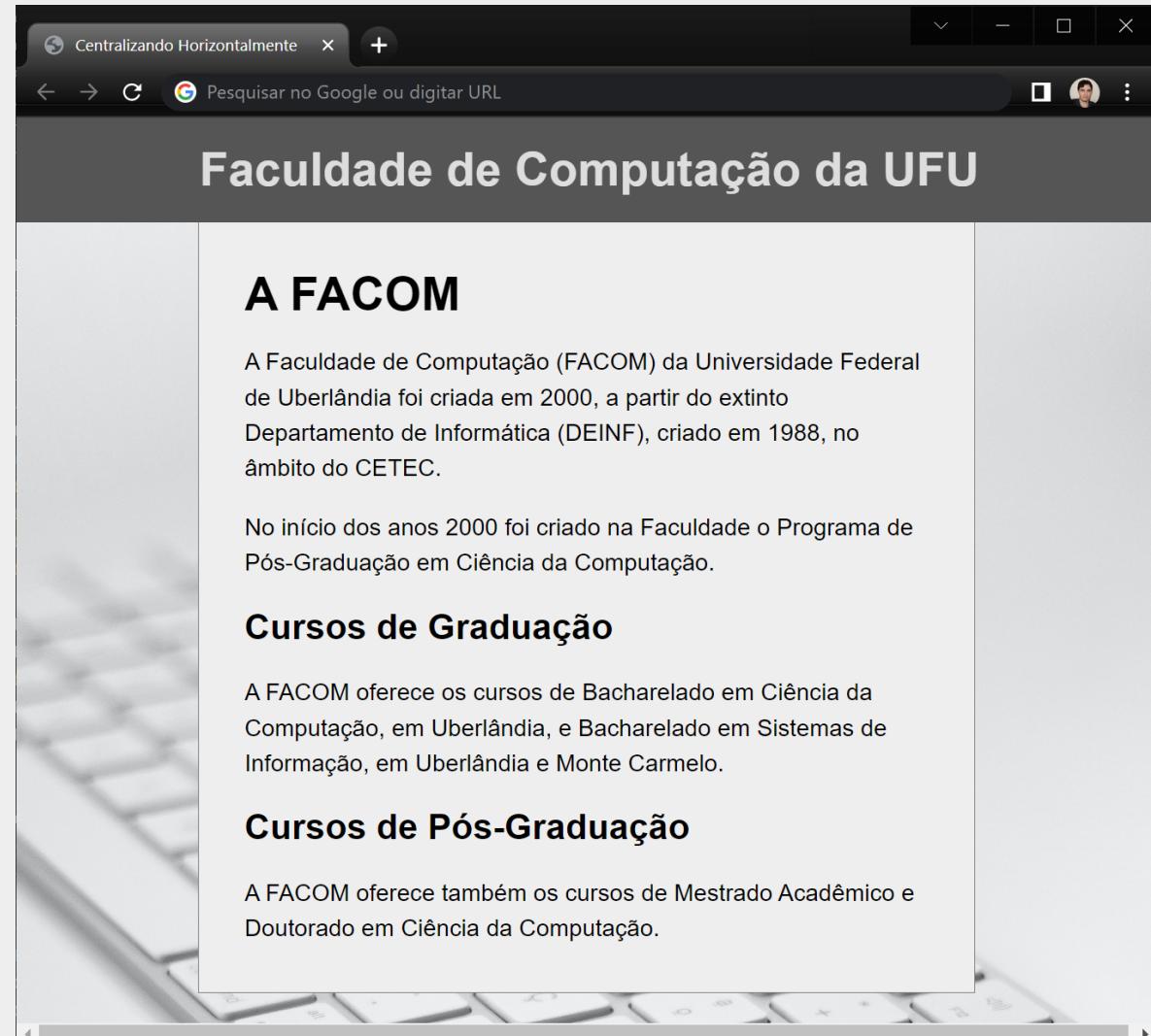
```
<style>
  body {
    text-align: justify;
    width: 60%;
    margin: 40px auto;
  }
</style>
</head>

<body>
  <h2>Página com Conteúdo Centralizado</h2>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisci...
    molestiae aliquam, enim, blanditiis laborios...
  <p>Lorem ipsum dolor sit amet consectetur adipisci...
    nemo totam commodi numquam delectus nobis! ...
  <h2>Página com Conteúdo Centralizado</h2>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisci...
    molestiae aliquam, enim, blanditiis laborios...
  <p>Lorem ipsum dolor sit amet consectetur adipisci...
    nemo totam commodi numquam delectus nobis! ...
  </body>
```



# Centralizando na Horizontal com width e margin

```
header {  
    background-color: #555;  
    width: 100%;  
}  
  
main {  
    background-color: #eee;  
    width: 60%;  
    margin: 0 auto;  
}  
</style>  
</head>  
<body>  
    <header><h1>Faculdade de Computação da UFU</h1>  
    </header>  
    <main>  
        <h1>A FACOM</h1>  
        <p>A Faculdade de Computação (FACOM) da  
            Universidade Federal de Uberlândia foi criad  
            a partir do extinto Departamento de Informát
```



# Pseudo-Classes

- Indica um **estado em particular** do elemento selecionado
- Palavra-chave adicionada a um seletor, precedida por dois-pontos

```
a:visited {  
    color: gray;  
}
```

Os links **que já foram visitados** aparecerão na cor cinza

```
input:invalid {  
    border-color: red;  
}
```

Os campos do tipo input aparecerão com borda vermelha **quando o conteúdo é inválido**  
(Ex.: um campo de e-mail faltando o @)

# Exemplos de Pseudo-Classes

Pseudo-Classe	Descrição	Exemplos
<b>:link</b>	Estilo inicial do link	<code>a:link {color: blue;}</code>
<b>:hover</b>	Usuário interage com elem. sem ativar Ex.: ponteiro do mouse sobre o elem.	<code>a:hover, button:hover {...}</code>
<b>:active</b>	Estado ativo do elemento Ex.: botão do mouse pres. sobre o elem.	<code>a:active, p:active {...}</code>

# Exemplos de Pseudo-Classes

Pseudo-Classe	Descrição	Exemplos
<b>:valid</b>	Campo de form. com conteúdo válido	<code>input:valid {...}</code>
<b>:invalid</b>	Campo com conteúdo inválido	<code>input:invalid {...}</code>
<b>:checked</b>	checkbox, radio ou option selecionado	<code>radio:checked {...}</code>
<b>:focus</b>	Elemento recebendo foco	<code>input:focus {...}</code>

# Exemplos de Pseudo-Classes

Pseudo-Classe	Descrição	Exemplos
<b>:first-child</b>	Primeiro filho do elemento pai	<code>li:first-child {color: blue;}</code>
<b>:last-child</b>	Último filho do elemento pai	<code>li:last-child {color: blue;}</code>
<b>:nth-child</b>	N-ésimo filho do elemento pai	<code>li:nth-child(2) {color: blue;}</code> Altera a cor dos <b>segundos</b> <li>'s

# Pseudo-Elementos

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

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

A primeira linha de cada parágrafo será apresentada com letras maiúsculas

```
p::selected {  
    color: green;  
    background-color: black;  
}
```

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

# Outros Pseudo-Elementos

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

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

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

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

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

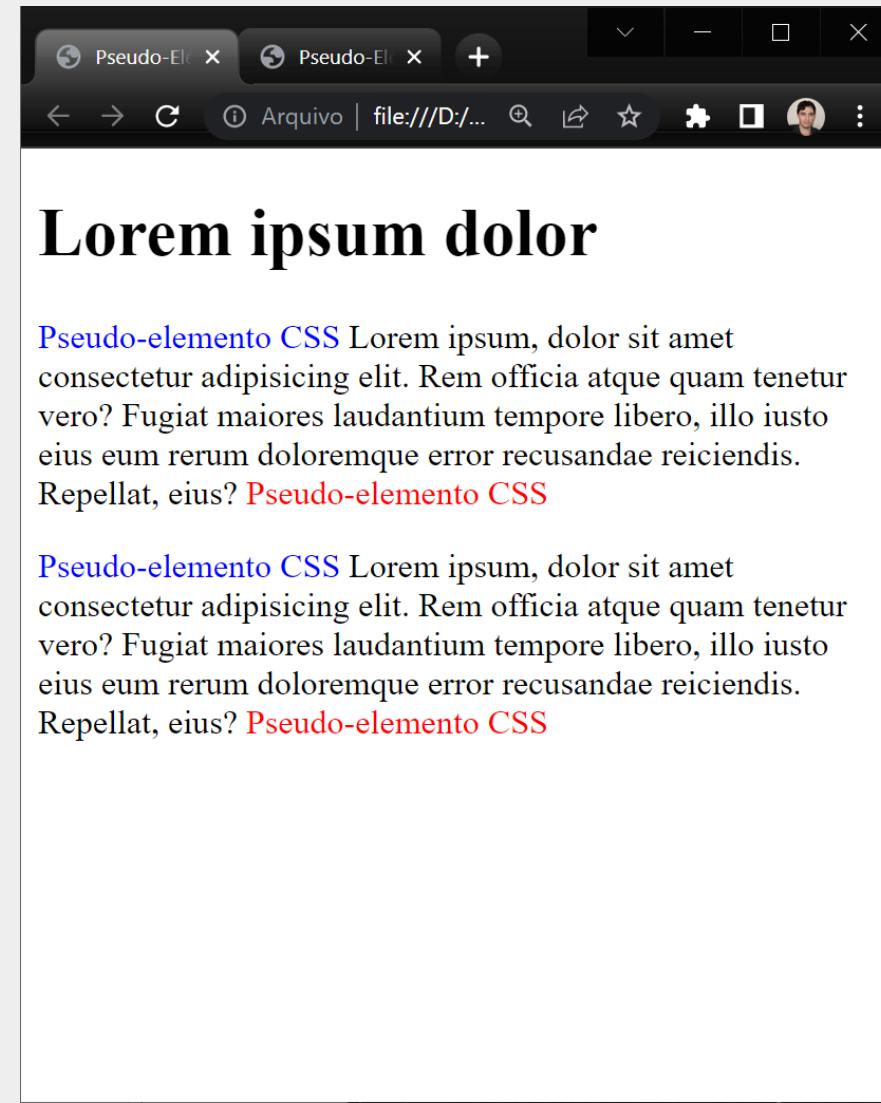
Altera a cor dos textos de **placeholder** dos campos **input**

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

# Pseudo-Elementos - Exemplo

```
<style>
  p::after {
    content: ' Pseudo-elemento CSS ';
    color: red;
  }

  p::before {
    content: ' Pseudo-elemento CSS ';
    color: blue;
  }
</style>
</head>
<body>
  <h1>Lorem ipsum dolor</h1>
  <p>Lorem ipsum, dolor sit amet consectetur
     laudantium tempore libero, illo iusto.
  <p>Lorem ipsum, dolor sit amet consectetur
     laudantium tempore libero, illo iusto.
```

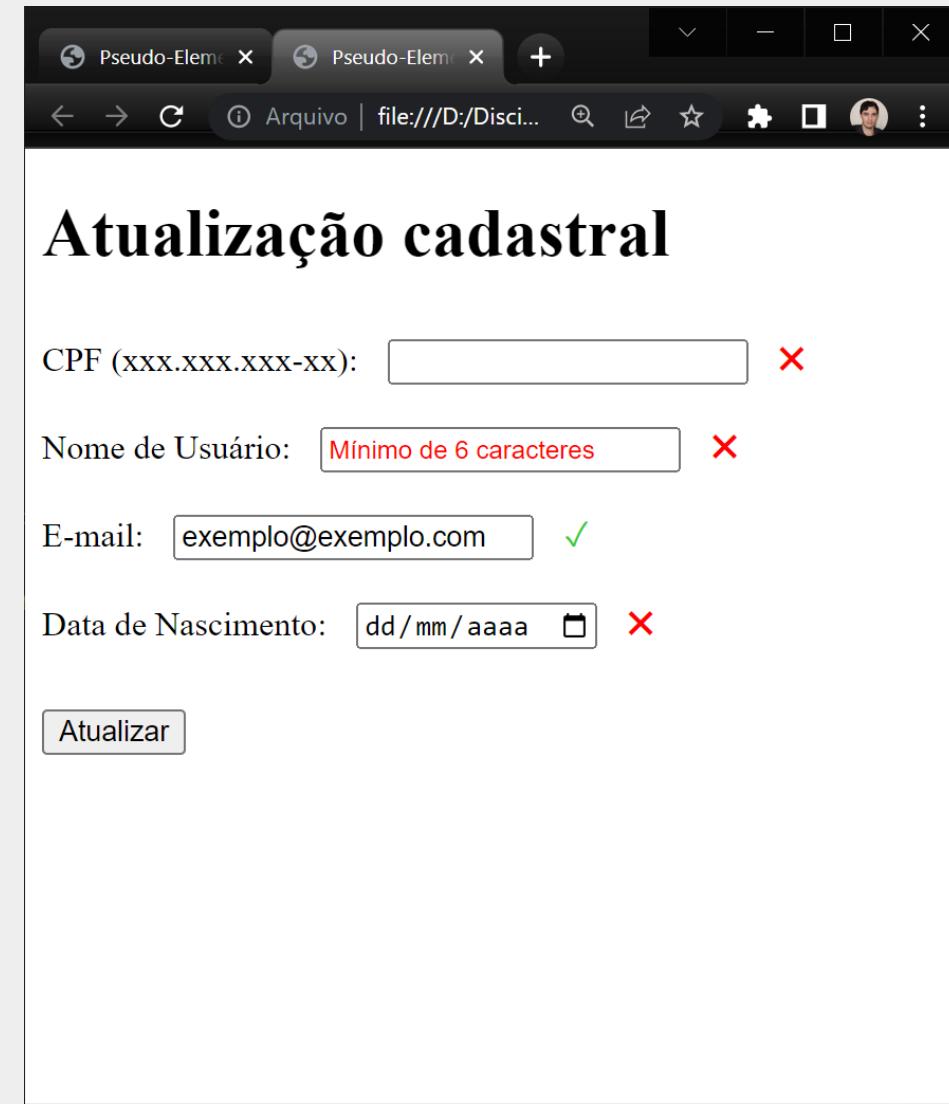


# Pseudo-Elementos - Exemplo

```
input:invalid + span::after {
    content: 'X';
    color: red;
}

input:valid + span::after {
    content: '✓';
    color: limegreen;
}
</style>
</head>

<body>
<form action="action.php" method="post">
    <h1>Atualização cadastral</h1>
    <div>
        <label for="cpf">CPF (xxx.xxx.xxx-xx):</label>
        <input type="text" id="cpf" name="cpf"
               pattern="\d{3}\.\d{3}\.\d{3}-\d{2}">
        <span></span>
    </div>
```



# Observações sobre Pseudo-Elementos

- A notação com 4 pontos (:::) foi introduzida na CSS3
- ::: é a forma atualmente recomendada para pseudo-elementos
- Diferencia pseudo-elementos de pseudo-classes
- Entretanto, na CSS2 utilizava-se apenas dois pontos (:)
  - Os navegadores ainda suportam essa forma

# Cascade, Especificidade e Herança

- Em caso de regras conflitantes, qual regra será aplicada?
- Problemas comuns:
  - Regras CSS sem efeito
  - Resultado muito diferente do esperado
- Nessas situações é fundamental conhecer como o navegador decide pelas regras
  - Ordem no código
  - Especificidade
  - Herança

# Ordem das Regras

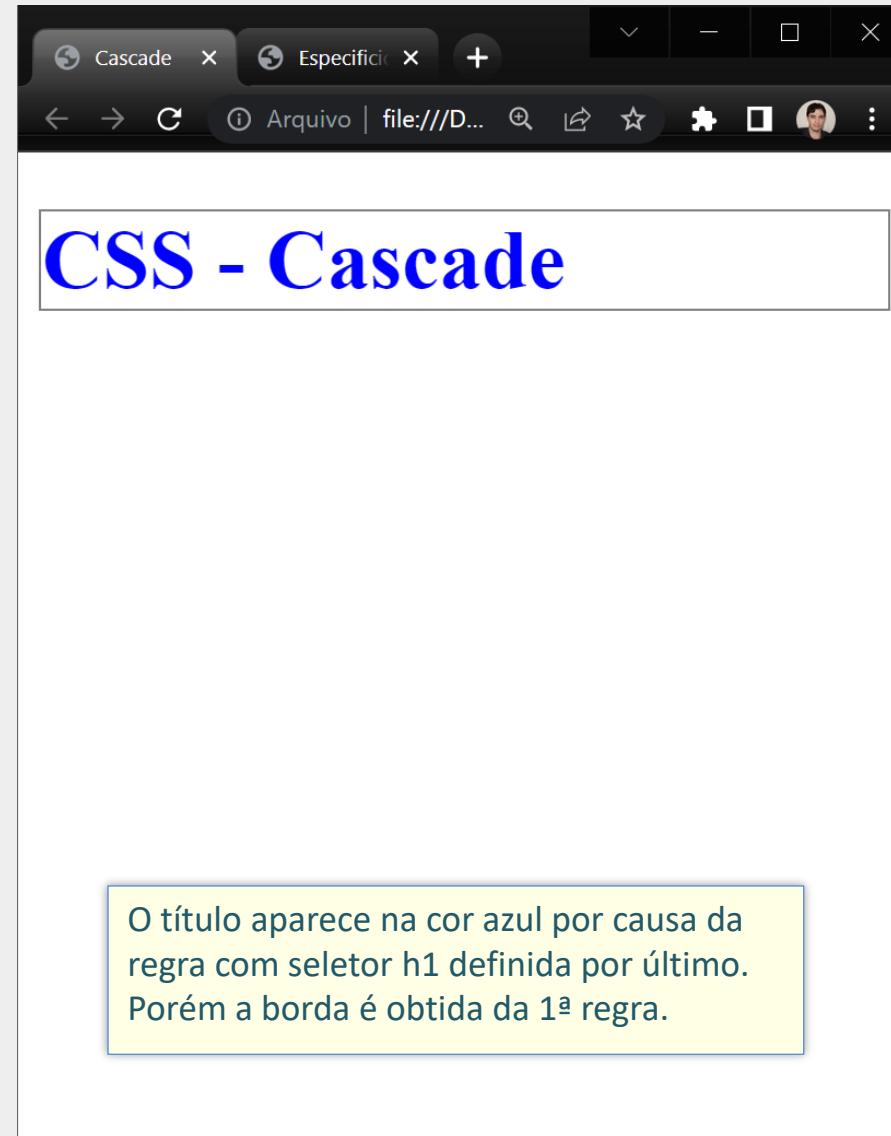
Caso hajam declarações CSS repetidas e elas estejam em regras do mesmo tipo de seletor, prevalecerá a **declaração definida por último**

# Ordem das Regras - Cascade

```
<style>
  h1 {
    color: red;
    border: 1px solid gray;
  }

  h1 {
    color: blue;
  }
</style>
</head>

<body>
  <h1>CSS - Cascade</h1>
</body>
```



# Especificidade

Seletor de Elemento

Seletor de Classe

Seletor de ID

Código Inline



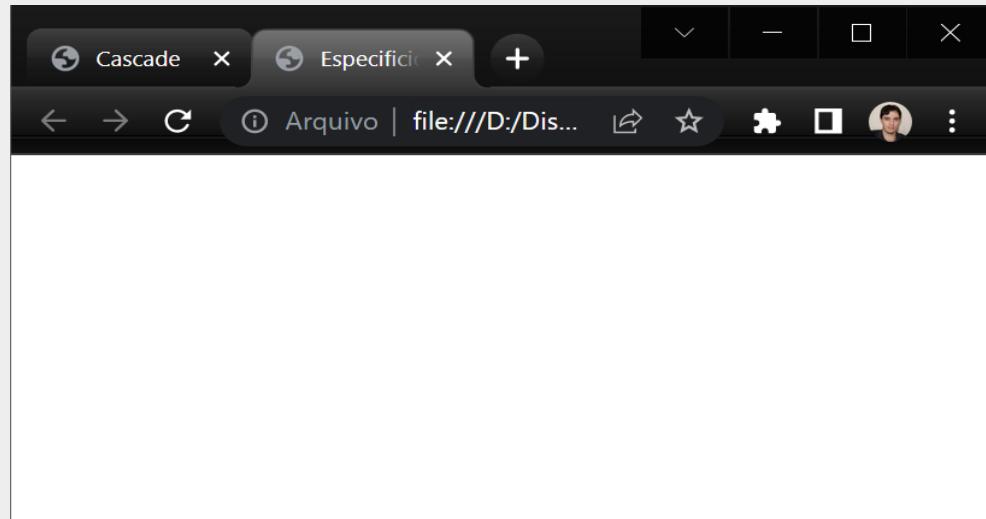
Maior especificidade  
Sobreposição de declarações anteriores

# Especificidade das Regras

```
<style>
  h1 {
    color: red;
    text-transform: lowercase;
    margin-top: 200px;
  }

  #tituloVerde {
    color: green;
    border: 1px solid black;
  }

  .tituloAzul {
    color: blue;
    text-transform: uppercase;
  }
</style>
</head>
<body>
  <h1 id="tituloVerde" class="tituloAzul">
    CSS - Especificidade
  </h1>
</body>
```



**CSS - ESPECIFICIDADE**

Uma cor diferente é definida em cada regra, porém a cor verde, do seletor de ID, é a que prevalece porque tal seletor é mais específico que os outros.

`text-transform` aparece na 1<sup>a</sup> e na 3<sup>a</sup> regra, porém o valor que prevalece é o da 3<sup>a</sup> regra porque ela é mais específica.

Observe que a margem superior vem da regra menos específica pois tal propriedade não é redefinida nas demais.

# Herança

- Algumas propriedades herdam os valores do elemento pai
  - Ex.: `font-family`, `color` e `text-align`
  - Ex.: um `<p>` dentro de um `<div>` herda a fonte definida para o `<div>`
- Outras propriedades não herdam
  - Ex.: `width`, `margin` e `padding`
  - Ex.: um `<p>` dentro de um `<div>` não herda as margens definidas para o `<div>`
- Definir uma propriedade para o valor `inherit` ativa a herança

# Design Responsivo

- Baseado na ideia de que o website deve ser bem exibido em todos os dispositivos (desktop, tablet ou smartphone);
- Evita a necessidade de ter diferentes versões do website para diferentes tamanhos de tela;
- Normalmente envolve a utilização da "meta tag viewport", media queries, unidades relativas, etc.;

# Meta Tag viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

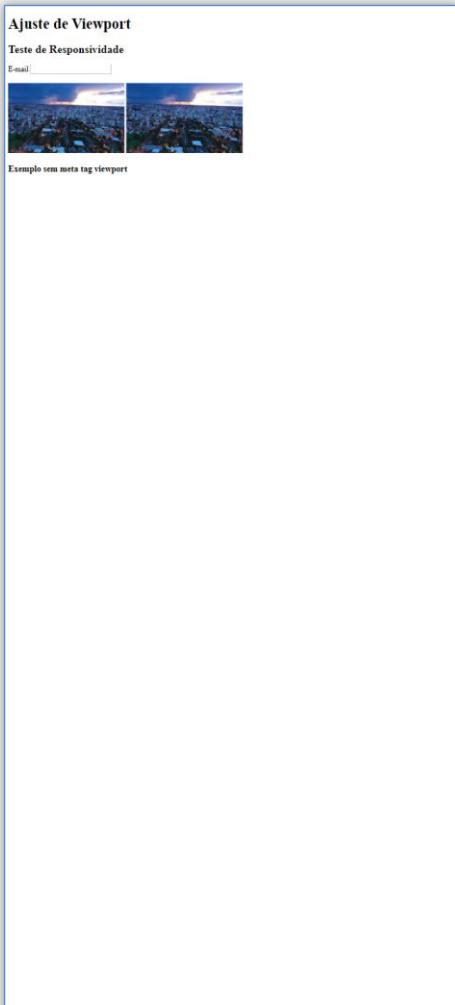
- Viabiliza a responsividade conforme dispositivo/tela
- Página melhor ajustada à tela de smartphones e tablets
- Ativa o fator de escala no dimensionamento em dispositivos móveis
  - O pixel ratio do dispositivo passa a ser considerado

# Utilizando a Meta Tag viewport

```
<head>
    <meta charset="UTF-8">
    <title>Teste de Viewport</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
    <h1>Ajuste de Viewport</h1>
    <h2>Teste de Responsividade</h2>
    <label for="email">E-mail</label>
    <input type="email" id="email">
    <br><br>
    
    
    <h3>Exemplo com meta tag viewport e</h3>
</body>
```

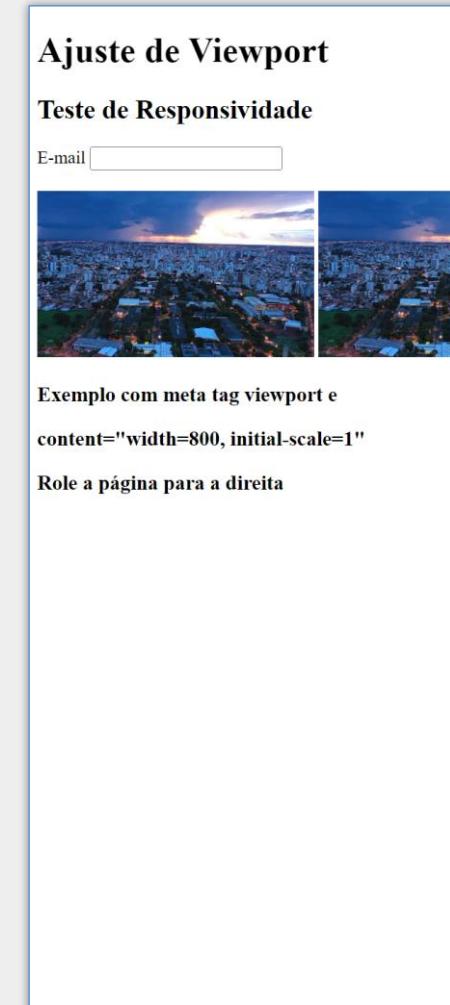
# Utilizando a Meta Tag viewport



Página **sem** a *meta tag viewport*  
(acessada pelo smartphone)



Página **com** a *meta tag viewport*  
(acessada pelo smartphone)



Página **com** a *meta tag viewport*  
e **width** com valor **800**



Página **com** a *meta tag viewport*  
e **width=800, initial-scale=2**

# Introdução à Media Queries

- Permite ao desenvolvedor testar condições sobre o navegador ou dispositivo do usuário para a aplicação ou não das regras CSS;
- Por exemplo, é possível aplicar estilos CSS apenas quando a tela do dispositivo tenha uma largura mínima ou máxima; ou esteja em determinada posição, como na vertical ou na horizontal;

# Introdução à Media Queries

**all** - para todos os dispositivos

**screen** - para dispositivos com tela

**print** - para impressão (ex. modo Ctrl-P)

**MediaType** é opcional.

Se omitido será  
considerado **all**

```
@media MediaType AND MediaCondition {
```

```
/* Código CSS */
```

```
}
```

```
(min-width: 400px)
```

```
(max-width: 900px)
```

```
(min-width: 400px) and (max-width: 900px)
```

```
(400px <= width <= 900px)
```

```
(orientation: portrait)
```

```
(orientation: landscape)
```

Exemplos de expressões (**media condition**). **min-width**,  
**max-width** e **orientation** são exemplos de **media features**

# Introdução à Media Queries

```
body {  
    width: 60%;  
    padding: 2% 0;  
    margin: 0 auto;  
}  
  
@media (max-width: 600px) {  
    body {  
        width: 95%;  
    }  
}
```

O corpo da página ocupará 95% da largura quando a largura da viewport for menor ou igual a 600px.

max-width é uma *media feature*. Há outras opções, como orientation

# Introdução à Media Queries

```
<style>
body {
    width: 60%;
    padding: 2% 0;
    margin: 0 auto;
    line-height: 2.0;
}

@media print and (orientation: portrait) {
    body {
        line-height: 1.1;
        font-size: 10pt;
    }
}
</style>
```

O espaçamento entre linhas e o tamanho da fonte serão reduzidos quando o documento estiver em modo de impressão na orientação **retrato** (Ctrl-P)

# Media Query com Lista de Expressões

- É possível combinar uma lista de media queries separando as expressões com vírgula
- O código CSS será aplicado quando pelo menos uma das expressões for verdadeira

```
@media screen and (orientation: landscape),  
screen and (min-width: 900px) {  
    body {  
        background-color: gray;  
    }  
}
```

O cor de fundo será cinza caso a mídia seja do tipo "tela" com orientação "horizontal" ou caso seja do tipo tela com largura mínima de 900px.

**OBS:** não existe o operador **OR** em media queries, mas a vírgula tem papel similar.

# 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
  - [jigsaw.w3.org/css-validator/](http://jigsaw.w3.org/css-validator/)

# CSS e Cache do Navegador

- Eventualmente o navegador pode armazenar estilos CSS em memória
- Neste caso, mudanças nas regras 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

# Referências

- [w3.org/Style/CSS/](http://w3.org/Style/CSS/)
- [developer.mozilla.org/en-US/docs/Web/CSS](http://developer.mozilla.org/en-US/docs/Web/CSS)
- [w3.org/Style/CSS/learning](http://w3.org/Style/CSS/learning)
- [w3.org/TR/mediaqueries-3/](http://w3.org/TR/mediaqueries-3/)
- HTML and CSS: Design and Build Websites, Jon Duckett.