Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Innominatum CONSULTORIA

Fundamentos Web

Módulo 1

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

DOM- Document Object Model

Sessão 4

Objetivo geral

Dotar os formandos dos conhecimentos necessários para criar layouts completos e responsivos, utilizando CSS Grid, Media Queries e boas práticas de organização e revisão de código.

Objetivos específicos

Ao final da sessão, os formandos deverão ser capazes de:

- Explicar o que é o **DOM** e como o browser representa o **HTML** internamente;
- Selecionar elementos da página utilizando **querySelector**, **querySelectorAll** e **getElementById**;
- Manipular conteúdo e estilos de elementos DOM, distinguindo **innerHTML** de **textContent** e **style** de **classList**;
- Criar, adicionar e remover elementos HTML dinamicamente através de **JavaScript**.

DOM – Document Object Model

Innominatum CONSULTORIA
IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Navegador (browser) – definição

um navegador Web evoluiu de uma simples ferramenta de exibição de documentos para um ambiente hospedeiro complexo e uma plataforma de aplicativos que desempenha diversas funções fundamentais como:

- Interpretação e Execução de Código;
- Representação e Manipulação de Conteúdo (DOM);
- Gerenciamento de Rede e Protocolos.

O que o navegador faz?

- Recebe um ficheiro HTML (texto):
  - Transforma o HTML numa estrutura em memória;
  - Essa estrutura chama-se DOM.

O que é o DOM?

O DOM (Document Object Model) é a API fundamental para representar e manipular o conteúdo de documentos HTML e XML. Ele transforma um documento estático em uma estrutura interativa, permitindo que scripts JavaScript alterem o conteúdo, a estrutura e a apresentação das páginas Web. Os principais conceitos do DOM são:

- Representação em Árvore;
- Objetos Principais.

HTML ≠ DOM

HTML é:

- Texto;
- Estático.

DOM é:

- Estrutura em memória;
- Dinâmico;
- Pode ser alterado via JS.

O DOM é uma árvore

Cada elemento vira um **nó**:

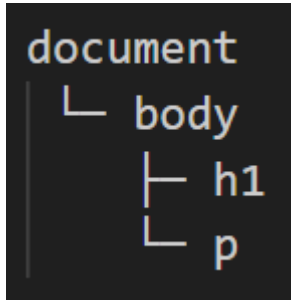
- Os nós organizam-se como:
  - Pai;
  - Filhos;
  - Irmãos.
- Essa estrutura chamamos de **árvore do DOM**.

O DOM é uma árvore

```
<body>
  <h1>Título</h1>
  <p>Texto</p>
</body>
```

O navegador transforma o código HTML em uma estrutura em árvore.

Cada tag vira um nó no DOM.



Tudo no DOM é um nó

Tipos de **nós**:

- Elementos (, <p>, <img>);
- Texto;
- Atributos;
- Comentários.

- Tudo pode ser manipulado via JS e a forma que o JS é pelo objeto **document**, ele representa toda a página.

Tudo no DOM é um nó

Tipos de **nós**:

- Elementos (, <p>, <img>);
- Texto;
- Atributos;
- Comentários.

- Tudo pode ser manipulado via JS e a forma que o JS é pelo objeto **document**, ele representa toda a página.

Estrutura do Grid (Linhas, Colunas e Áreas)

Ativando o grid e definindo as colunas

```css
.layout {
  display: grid;
  /* Definimos o grid nessa classe */
}
```

```css
.layout {
  grid-template-columns: 200px 1fr;
  /* Definindo as colunas e seus valores */
}
```

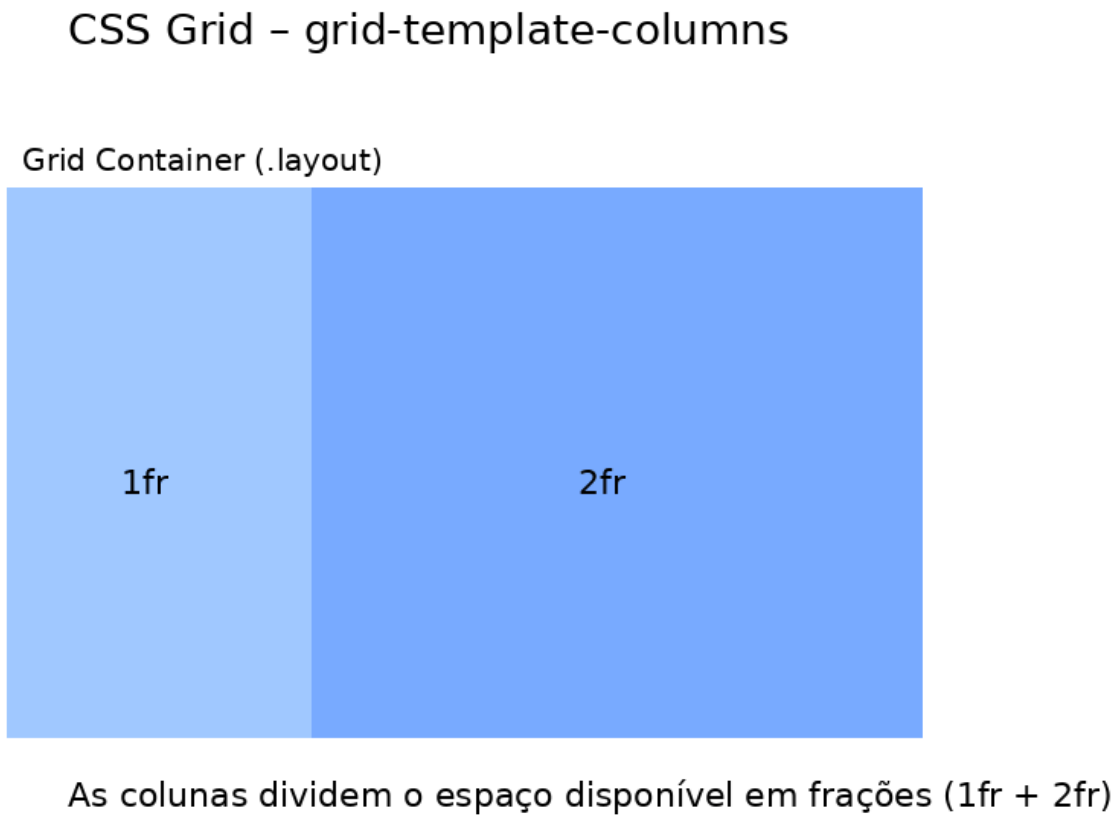
A unidade fr (fração)

```css
.layout {
  grid-template-columns: 1fr 2fr;
  /* fr representa uma fração do espaço disponível */
}
```

Definir linhas no grid

```css
.layout {
  grid-template-rows: auto 1fr auto;
  /* Defini-se 3 linhas, com a primeira e última automáticas
  e a linha do meio que ocupa o espaço restante */
}
```

As duas dimensões do grid: colunas e linhas

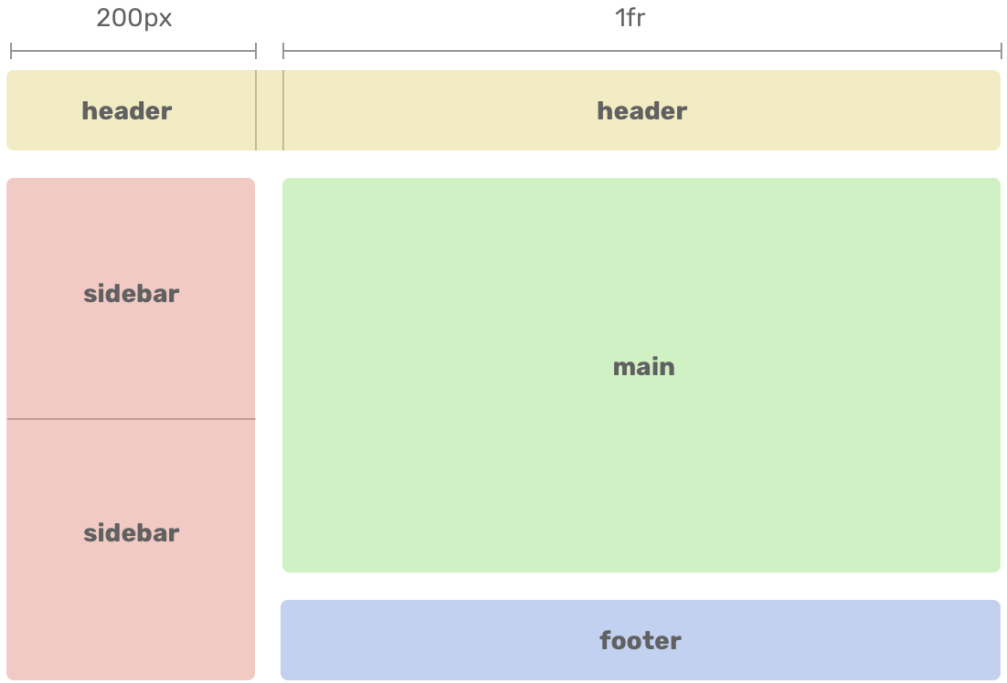


Grid area

```
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "menu content"
    "footer footer";
}
```

- Conjunto de células que formam uma região retangular nomeada;
- Áreas tornam o layout mais legível;
- Facilita manutenção e leitura do código.

```
grid-template-columns: 200px 1fr;
grid-template-areas: "header header"
                     "sidebar main"
                     "sidebar footer";
```



Responsividade

O que é responsividade?

- Responsividade é a capacidade de um site de:
  - Adaptar-se a diferentes tamanhos de ecrã;
  - Manter legibilidade e usabilidade.
- O layout **muda**, não apenas encolhe;
- Responsividade não é uma escolha.

Por que a responsividade é importante?

- Utilizadores acedem via:
  - Telemóvel;
  - Tablet;
  - Portátil;
  - Monitores e TVs de alta resolução.
- Um único layout não serve para todos.

Breakpoints

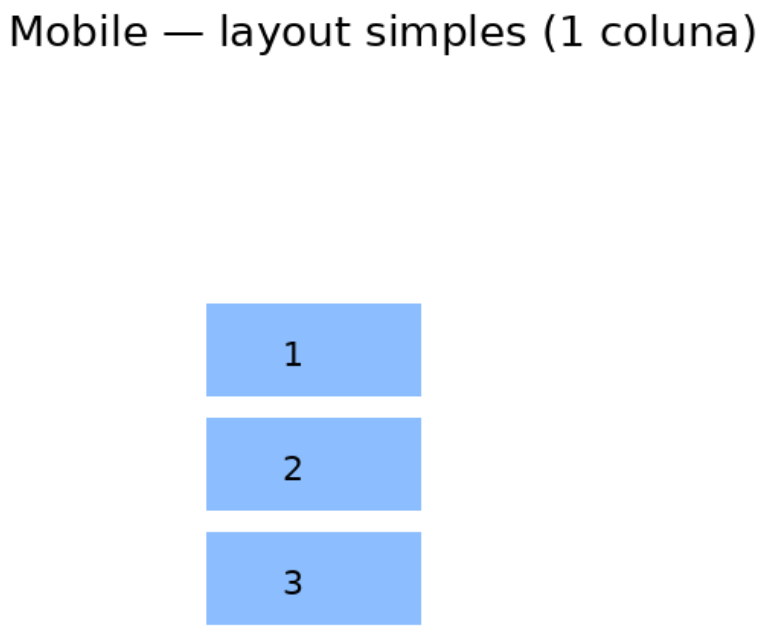
- Breakpoints são **pontos onde o layout muda**;
- Normalmente associados a larguras do ecrã;
  - Mobile;
  - Tablet;
  - Desktop.

Breakpoints Comuns

Definição de Media queries

- Media queries permitem aplicar CSS **condicionalmente**:
  - Aplicação de estilos com base nas características do dispositivo;
  - Aplicação de estilos com base no ambiente, como a largura da tela (viewport), altura, orientação (retrato/paisagem) ou tipo de mídia (tela, impressão)
  - Possibilitam o design responsivo, ao adaptar o layout e a melhor apresentação possível do conteúdo.

Mobile, tablet e desktop



Layout desktop x mobile

```css
/* Desktop */
.layout {
  grid-template-areas:
    "header header"
    "menu   main"
    "footer footer";
}
```

```css
/* Mobile */
@media (max-width: 768px) {
  .layout {
    grid-template-areas:
      "header"
      "main"
      "menu"
      "footer";
  }
}
```

Limite inferior

```css
/* Estilos para telas a partir de 768px (tablets e maiores) */
@media (min-width: 768px) {
  body {
    font-size: 18px; /* Aumenta o texto */
    padding: 20px; /* Mais espaço */
  }
  .coluna {
    width: 50%; /* Transforma layout em duas colunas */
    float: left;
  }
}
```

Innominatum CONSULTORIA

Variáveis CSS

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

O que são variáveis CSS

Variáveis CSS (também conhecidas com propriedades personalizadas), permitem **armazenar valores reutilizáveis** (cores, fontes, tamanhos) no CSS, seu objetivo é manter o código mais organizado, facilitar a manutenção e criar temas dinâmicos. Os nomes das variáveis são sensíveis a maiúsculo e minúsculo e as variáveis ainda podem serem manipuladas por JavaScript.

Criar variáveis com :root

```css
:root {
  --primary-color: #0057ff;
  --secondary-color: #f5f5f5;
  --spacing: 1rem;
}
/* :root representa o elemento raiz HTML */
/* Variáveis ficam disponíveis globalmente */
```

Usar variáveis com var()

```css
body {
  background-color: var(--secondary-color);
  color: var(--primary-color);
}
```

Variáveis com base para temas

```css
:root {
  --primary-color: #0057ff;
}

.dark-theme {
  --primary-color: #00171e;
}
```

Pseudo Classes (Estados e Interação)

O que são pseudo-classes?

Pseudo-classes em CSS são palavras-chave que selecionam elementos com base em um **estado especial** ou **informação não presente no HTML**, permitindo estilizar itens como links visitados (:visited), botões sob o mouse (:hover), campos de formulário marcados (:checked) ou elementos em uma posição específica (como :first-child), sem precisar adicionar classes ou IDs extras ao HTML, tornando o código mais limpo e dinâmico.

Para que servem pseudo-classes?

- Dar feedback visual ao aplicar estilo dinâmico e condicional;
- Melhorar a experiência do utilizador ao criar interatividade através dos seus elementos.

Pseudo-classes – Exemplo – hover

```html
<div class="card">
  <img src="image.jpg" alt="Person 1">
  <div class="card-info">
    <h3>Person 1</h3>
    <p>Lorem ipsum dolor sit amet.</p>
  </div>
</div>
```

```css
.card:hover .card-info {
  transform: translateY(0);
}
```

Pseudo-classes – Exemplo – hover

Person 1

Person 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Person 3

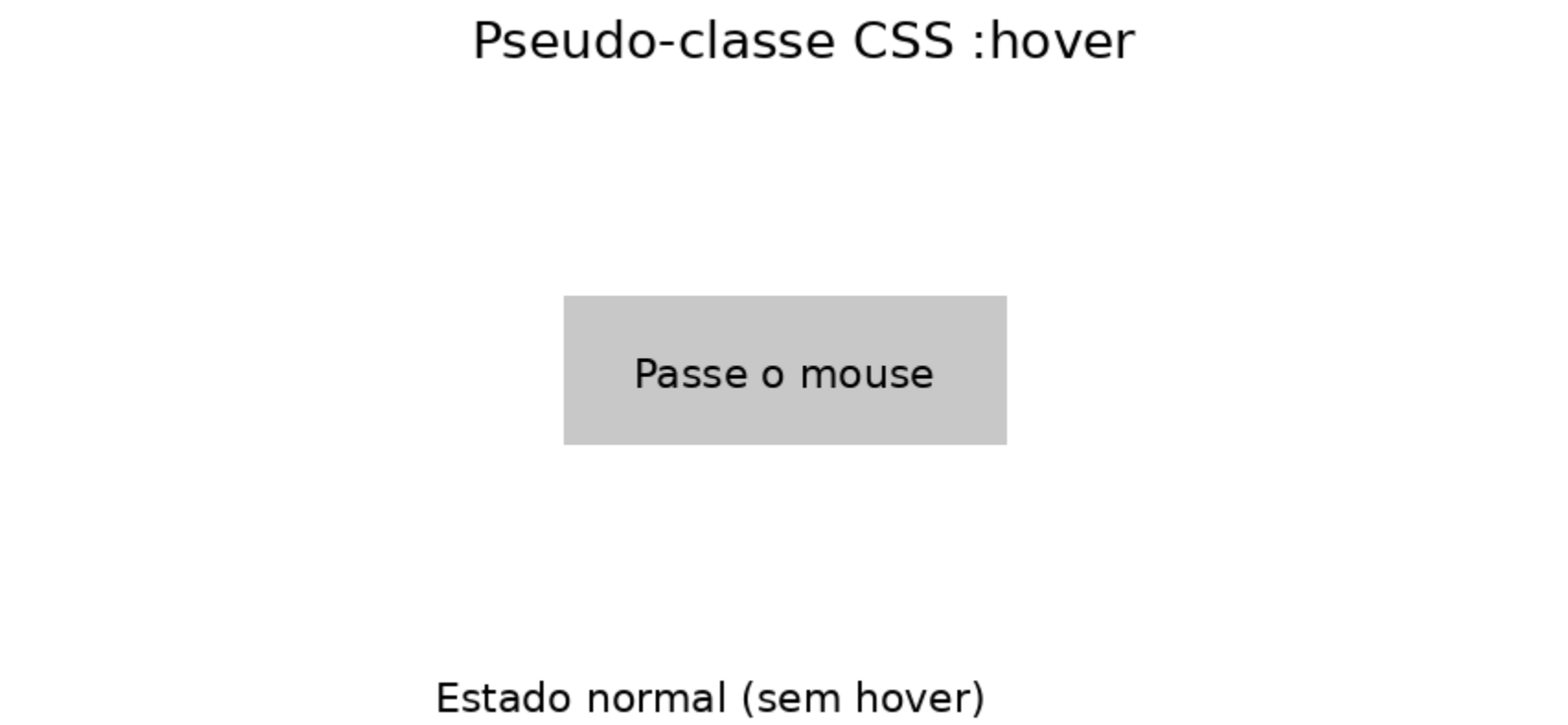
Person 4

Pseudo-classes – Exemplo – hover

```html
<button class="btn-hover">
  Passe o mouse
</button>
```

```css
.btn-hover:hover {
  background-color: #78befe;
}
```

Pseudo-classes – Exemplo – hover



Pseudo-classes – Exemplo – focus

```html
<div class="container-input">
  <label for="nome">Digite seu nome:</label>
  <input type="text" id="nome"
  class="input-field" placeholder="Seu Nome">
</div>
```

```css
/* Estilo quando o input recebe foco */
.input-field:focus {
    border-color: blue;
    box-shadow:
    0 0 0 3px rgba(0, 0, 255, 0.2);
    outline: none;
}
```

Pseudo-classes – Exemplo – focus

Pseudo-classe :focus

Estado normal (sem foco)

Pseudo-classes – Exemplo – active

```html
<button class="btn">
  Clicar
</button>
```

```css
.btn:active {
  transform: scale(0.95);
  box-shadow: inset 0 4px 8px rgba(0,0,0,0.3);
}
```

Pseudo-classes – Exemplo – active

O que são pseudo-elementos?

Pseudo-elementos CSS são palavras-chave que permitem estilizar partes específicas de um elemento HTML ou criar conteúdo "virtual" que não existe no DOM, como a primeira letra/linha ou conteúdo antes/depois de um elemento, usando a sintaxe **seletor::pseudo-elemento (::)** para diferenciá-los das pseudo-classes (:) e adicionar estilos como { <</font-size<</ } ou conteúdo com { <</content<</ }. Eles são úteis para formatação de texto e adição de elementos decorativos sem adicionar mais HTML.

Diferença entre pseudo-classe e pseudo-elementos

- Pseudo-classe -> Define um estado especial de um elemento existente [ex: quando o mouse passa por cima, deste elemento] (:hover);
- Pseudo-elemento -> Estiliza uma parte específica de um elemento ou cria uma estrutura visual que não existe no HTML original (::before);
- Um reage a interação, o outro cria estrutura visual.

Casos típicos de uso

- Ícones decorativos;
- Linhas e separadores;
- Destaques visuais;
- Overlays;
- Badges simples.
- Tudo que não é conteúdo visual

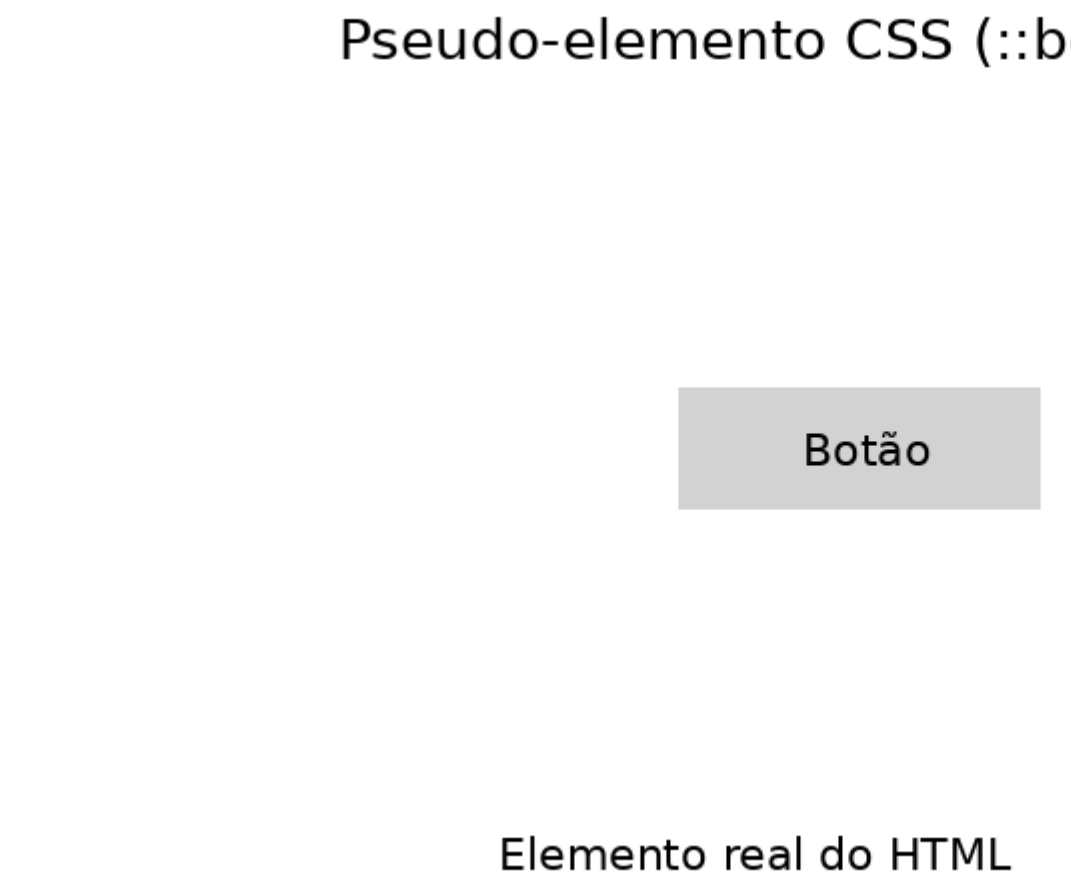
Pseudo-elementos – Exemplo – after

```html
<button class="meu-botao">
    Botão
</button>
```

```css
/* Pseudo-elemento ::before */
.meu-botao::before {
  content: "::before"; /* Texto que aparecerá */
  background-color: blue;
}
```

```css
/* Pseudo-elemento ::after */
.meu-botao::after {
  content: "::after"; /* Texto que aparecerá */
  background-color: #ffcc80;
}
```

Pseudo-elementos – Exemplo – after



Síntese

- CSS Grid (Layout 2D);
- Responsividade;
- Variáveis CSS;
- Pseudo-classes;
- Pseudo-elementos.

Conclusão

Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL