

```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_name))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly
```

-- OPERATOR CLASSES ----

# Aula de Scripts

```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_name))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly
```

-- OPERATOR CLASSES ----

# Recapitulação HTML

# Introdução HTML

## O que é HTML?

HTML, ou HyperText Markup Language, é a linguagem de marcação padrão utilizada para criar páginas web. É fundamental entender seu uso para desenvolver um site.

## Importância do HTML

HTML serve como a base da maioria das páginas web. Ele estrutura o conteúdo, permitindo que navegadores interpretem e exibam informações corretamente.

## Como funciona HTML?

HTML utiliza 'tags' para organizar conteúdos. Cada tag tem um propósito específico, como definição de cabeçalhos, parágrafos e elementos de listas.



# Tags Estruturais

## Tag <body>

Todos os conteúdos visíveis para o usuário são organizados dentro da tag <body>. Aqui estão parágrafos, imagens e outros elementos.

## Tag <html>

Essa é a tag raiz que envolve todo o conteúdo HTML de uma página. Sem ela, o navegador não entenderá o que está tudo ali.

## Tag <head>

A tag <head> contém metadados e referências a estilos e scripts. Fundamental para a inclusão de links externos e definição do título da página.



```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Um exemplo básico de HTML seguindo as melhores práticas.">
  <meta name="author" content="Seu Nome">
  <title>Exemplo Básico de HTML</title>
</head>
<body>
  <header>
    <h1>Bem-vindo ao meu site</h1>
  </header>
  <main>
    <section>
      <h2>Seção de Conteúdo</h2>
      <p>Este é um exemplo básico de um documento HTML.</p>
    </section>
  </main>
  <footer>
    <p>&copy; 2023 Seu Nome. Todos os direitos reservados.</p>
  </footer>
</body>
</html>
```

# Tags de Formatação

## Tag <h1> a <h6>

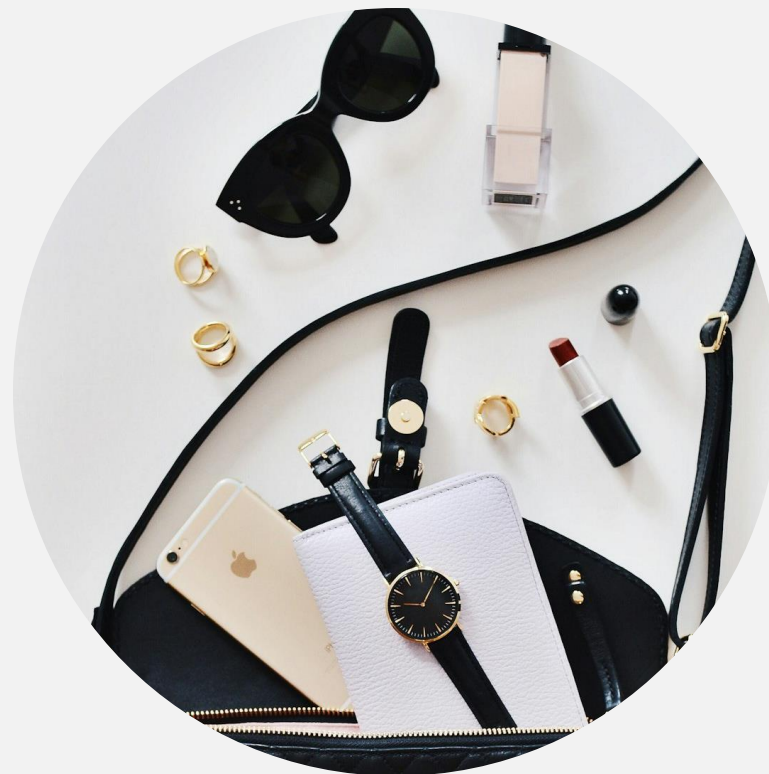
Essas tags definem diferentes níveis de cabeçalhos. <h1> é o mais importante, e <h6> o menos, ambos ajudam SEO e organização.

## Tag <p>

Utilizada para parágrafos de texto, a tag <p> facilita a leitura e estruturação do conteúdo textual na página web.

## Tag <strong> e <em>

As tags <strong> e <em> são utilizadas para destacar texto. A primeira deixa o texto em negrito, enquanto a segunda em itálico.



# Tags de Mídia

## Tag <img>

A tag <img> é usada para incorporar imagens. Importante incluir atributos como 'alt' para acessibilidade e SEO.

## Tag <audio>

Utilize a tag <audio> para incluir arquivos de som. Permite a reprodução de músicas ou áudios sem a necessidade de plugins adicionais.

## Tag <video>

Com a tag <video>, você pode adicionar vídeos diretamente à página. A compatibilidade com diferentes formatos é essencial.





# Estrutura Semântica

## **Aumenta a acessibilidade**

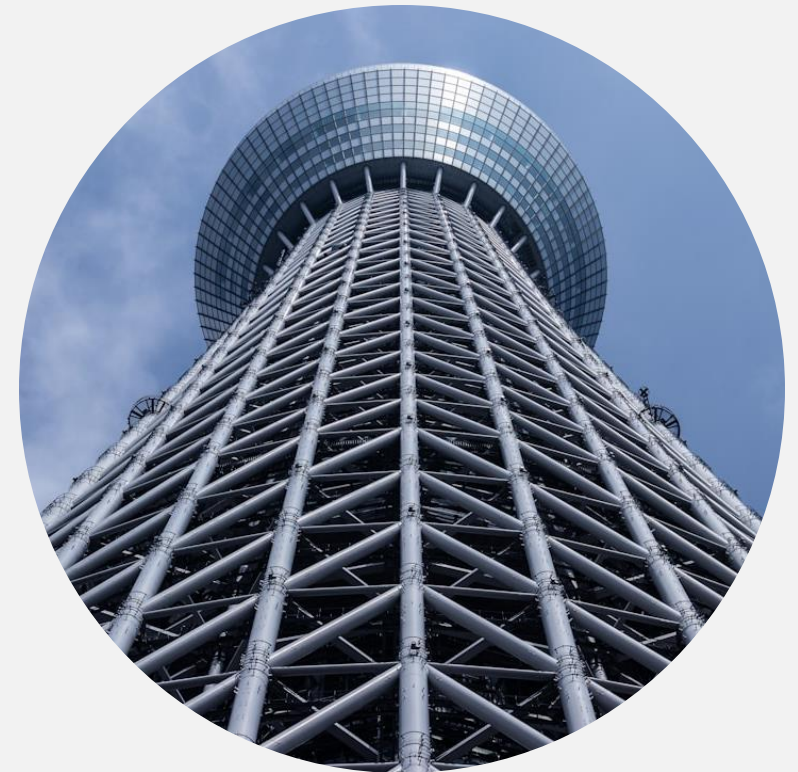
Identificar e usar elementos semânticos permite que tecnologias assistivas interpretem melhor o conteúdo, tornando-o acessível a mais usuários.

## **Melhora o SEO**

Utilizar tags semânticas melhora a indexação por motores de busca, ajudando na classificação do seu site.

## **Clareza do Conteúdo**

Strings de texto e hierarquia bem definidos facilitam a compreensão e navegação pelo conteúdo.





# Uso de Tags Correto

## Tags de Formatação


Evite usar tags como `<b>` ou `<i>` para estilização;

## Tags de Estrutura

Utilize `<header>`, `<footer>` e `<nav>` para organizar o layout do site, permitindo fácil navegação e melhor compreensão.

## Tags para Conteúdo

Tags como `<article>`, `<section>`, `<aside>` ajudam a definir a semântica, facilitando a legibilidade para os usuários.



# Organização do Código

Uma boa organização do código HTML melhora a legibilidade e a manutenção, especialmente em projetos colaborativos. Siga as práticas recomendadas.

Princípio	Descrição	Benefício
Indentação	Use espaços ou tabs consistentes	Melhora a legibilidade do código
Comentários	Adicione comentários claros e concisos	Facilita a manutenção do código
Estrutura lógica	Agrupe seções semelhantes	Facilita a navegação e entendimento

<header>

<nav>

</nav></header>

<div>

<article>

<section></section>

</article>

<aside>

</aside>

</div>

<footer>

</footer>

```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

# Acessibilidade

# Boas Práticas de Acessibilidade

## Atributos ARIA

Implementar atributos ARIA corretamente ajuda tecnologias assistivas a entenderem melhor o conteúdo, melhorando a experiência do usuário.

## Use Elementos Naturais

Utilizar tags HTML apropriadas melhora a acessibilidade, tornando mais fácil para os usuários navegar e entender a página.

## Foco e Navegação

Certifique-se de que todos os elementos interativos sejam acessíveis via teclado, garantindo que todos possam navegar na sua página.



# Introdução ao ARIA

## Definição e Propósito

ARIA, ou Accessible Rich Internet Applications, é um conjunto de atributos que melhora a acessibilidade em aplicações web complexas. Seu propósito é ajudar tecnologias assistivas.

## Importância para Acessibilidade

Esses atributos ajudam usuários com deficiências a interagirem com interfaces dinâmicas, proporcionando uma experiência mais inclusiva na web.

## Conformidade com Padrões

A utilização de ARIA é fundamental para atender normas internacionais como WCAG, garantindo que sites sejam acessíveis a todos.





# Estruturas ARIA

## Landmarks

Landmarks auxiliam na navegação, permitindo que usuários identifiquem rapidamente seções importantes da página.

## Roles

Roles especificam a função de um elemento na interface, como 'button', 'navigation', entre outros.

## States e Properties

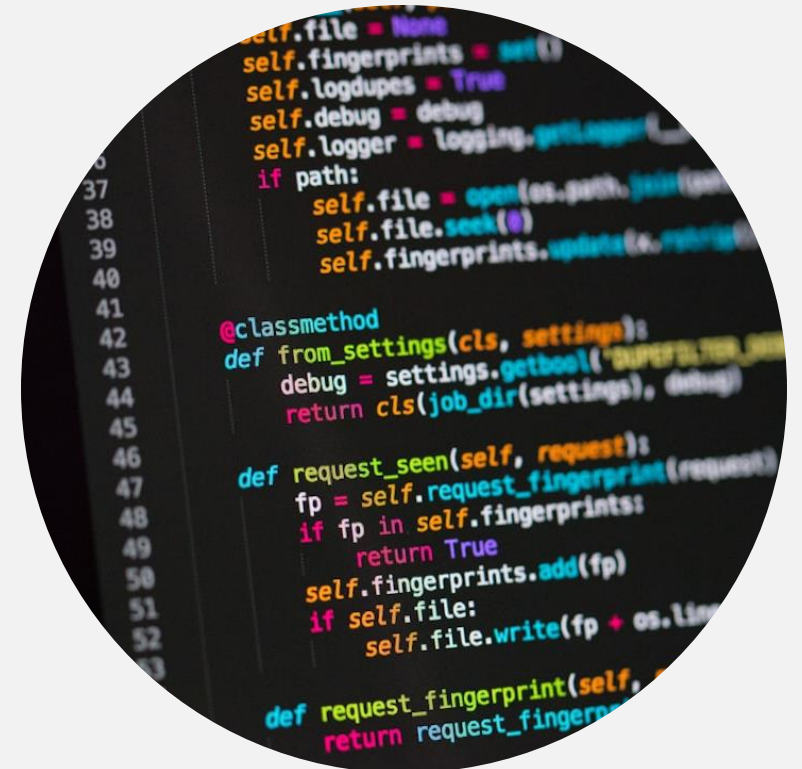
States e properties fornecem informações sobre o estado atual de um elemento, como 'checked' para checkboxes.



# Exemplos de Uso

## Botões

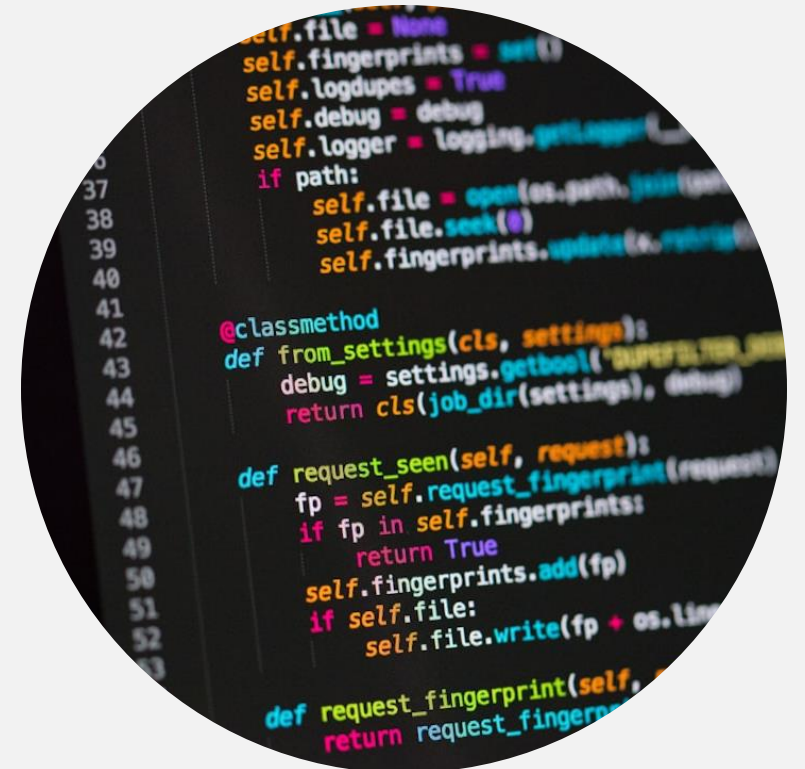
Exemplo: `<button aria-label='Salvar'>Salvar</button>`. O ARIA melhora a acessibilidade ao adicionar descrições.



# Exemplos de Uso

## Navegação

```
<nav aria-label="Navegação principal">  
<ul>  
<li><a href="#home">Início</a></li>  
<li><a href="#about">Sobre</a></li>  
<li><a href="#services">Serviços</a></li>  
<li><a href="#contact">Contato</a></li>  
</ul>  
</nav>
```



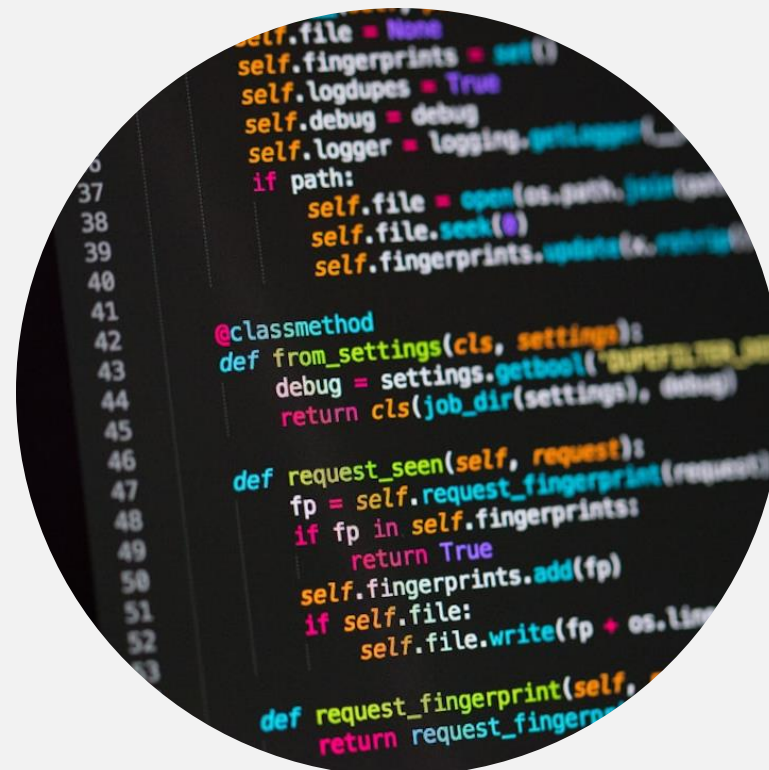
# Exemplos de Uso

## Botão de Alternância (Toggle)

```
<button aria-pressed="false" onclick="this.setAttribute('aria-pressed', this.getAttribute('aria-pressed') === 'true' ? 'false' : 'true')">
```

Mostrar/Ocultar Conteúdo

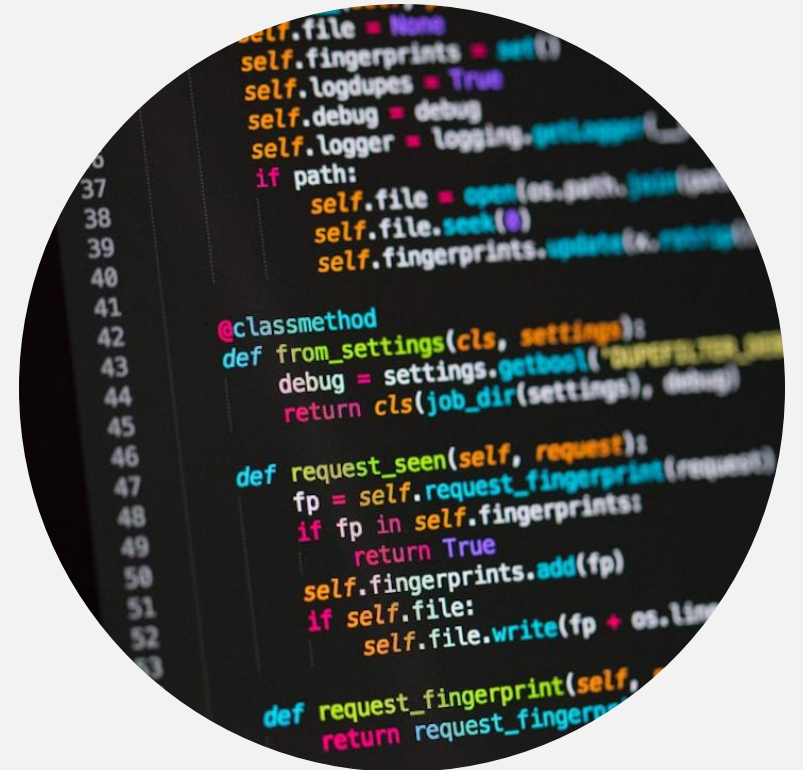
```
</button>
```



# Exemplos de Uso

## Conteúdo Dinâmico

```
<div aria-live="polite">  
<p id="dynamicContent">Este conteúdo será atualizado dinamicamente.</p>  
</div>  
<script>  
// Simula uma atualização de conteúdo  
setTimeout(function() {  
  document.getElementById('dynamicContent').textContent = 'O conteúdo foi atualizado!';  
}, 3000);  
</script>
```



# Exemplos de Uso

## Barra Lateral (Aside)

```
<aside aria-labelledby="sidebarHeading">
```

```
<h2 id="sidebarHeading">Informações Adicionais</h2>
```

```
<p>Aqui estão algumas informações adicionais...</p>
```

```
</aside>
```





# Exemplos de Uso

## Campos de Formulário

```
<form>
```

```
<label for="email">E-mail:</label>
```

```
<input type="email" id="email" name="email" aria-required="true">
```

```
</form>
```



# Desafios e Limitações

## Desconhecimento dos Desenvolvedores

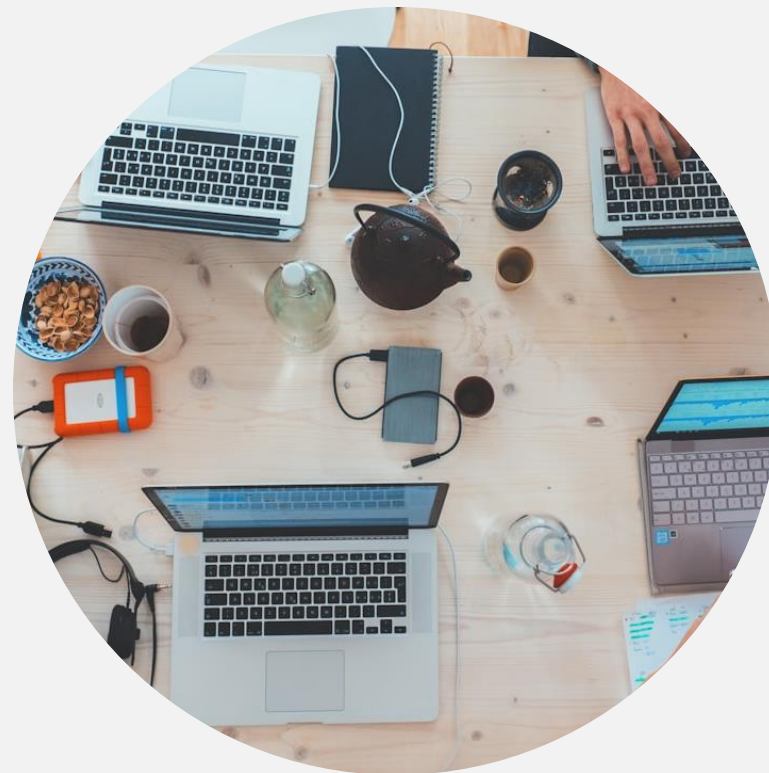
Muitos desenvolvedores não utilizam ARIA adequadamente, o que diminui sua eficácia e potencial.

## Overuse de ARIA

Usar ARIA desnecessariamente pode confundir tecnologias assistivas, causando erros de interpretação.

## Performance e Responsividade

O uso excessivo de ARIA pode impactar a performance e causar lentidão em aplicações web.



```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add
```

```
mirror_ob.select= 1
```

```
mirror_ob.select=1
```

```
context.scene.objects.active
```

```
("Selected" + str(modifier_ob
```

```
mirror_ob.select = 0
```

```
= bpy.context.selected_object
```

```
data.objects[one.name].select
```

```
print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

# Mãos à Obra

```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_name))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

# HTML Preparação para CSS

# HTML com CSS

## Estrutura HTML Básica

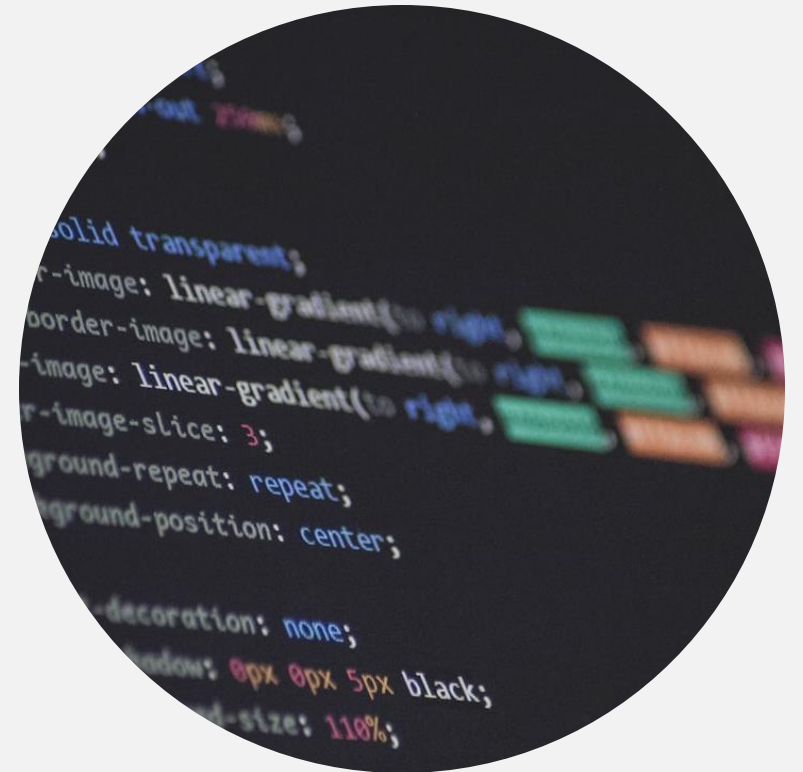
A estrutura básica de uma página HTML consiste em tags como `<html>`, `<head>`, e `<body>`. Essas tags são essenciais para definir o conteúdo.

## Como CSS Funciona

CSS (Cascading Style Sheets) é usado para estilizar HTML. Ele permite personalizar a aparência de elementos dentro de uma página.

## Integração do CSS

O CSS pode ser integrado ao HTML de três maneiras: inline, internal e external. Cada método tem uso e aplicabilidade específica.



# Seletores Essenciais

## Seletores por Classe

Usam o ponto (.) antes do nome da classe para estilizar elementos. Exemplo: `.menu { font-size: 14px; }`.

## Seletores por Nome

Selecionam elementos HTML pelo seu nome. Exemplo: `h1 { color: red; }`.

## Seletores por ID

Utilizam o símbolo # para selecionar um elemento único. Exemplo: `#header { background: blue; }`.





# Exemplo de Uso

## Exemplo de CSS Inline

O CSS inline resulta em estilização imediata. Exemplo: `<h1 style='color: red;'>Título</h1>`.

## Exemplo de CSS Interno

CSS interno é usado dentro da tag `<style>`. Exemplo: `<style>h1 {color: blue;}</style>`.

## Exemplo de CSS Externo

Utilizar um arquivo .css externo é eficiente. Exemplo: `<link rel='stylesheet' href='styles.css'>`.



# Importância dos IDs

## Unicidade do ID

Os IDs devem ser únicos em uma página, permitindo uma estilização específica para um único elemento.

## Facilidade de Seleção

Fácil de selecionar e estilizar um elemento. Por exemplo, `#footer { background-color: grey; }`.

## Interatividade com JavaScript

IDs são comumente utilizados para manipulação de elementos com JavaScript, como em eventos de clique.



```
mirror_mod.use_z = True  
operation == "MIRROR_Z"  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add
```

```
mirror_ob.select= 1
```

```
mirror_ob.select=1
```

```
context.scene.objects.active
```

```
("Selected" + str(modifier_ob
```

```
mirror_ob.select = 0
```

```
= bpy.context.selected_object
```

```
data.objects[one.name].select
```

```
print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

# CSS

# "Style sheets" ou "Folhas de Estilo"

- **CSS** é um padrão recomendado pelo **W3C** para todos os browsers (a World Wide Web Consortium é a entidade que cuida do desenvolvimento e padronização das tecnologias ligadas à Web).
- "Cascading Style sheets" ou "Folhas de Estilo Encadeadas" é uma tecnologia (linguagem) criada para definir estilos (cores, tipologia, posicionamento, etc)

# Recomendações para o desenvolvedor

- Com **CSS** economiza-se tempo de criação e manutenção (isola os códigos de formatação aplicado a várias páginas HTML em um único arquivo ".css");
- Reduz código de descritores HTML da página (tags);
- Os navegadores (browsers) carregam mais rápido;

# Recomendações para o desenvolvedor

- Maior eficiência no gerenciamento do layout;
- **CSS** é simples pois descreve apenas estilos;
- Com recursos mais avançados de **CSS** pode-se conseguir um design sofisticado sem utilização de imagens e tabelas.



# Como Usar?

- CSS externo (arquivo ".css")
- CSS interno (dentro da tag <head>)
- Estilos inline (dentro do elemento HTML)

# Inline

Estilos são aplicados diretamente no elemento HTML usando o atributo **style**.

```
<!DOCTYPE html>  
  
<html lang="pt-BR">  
  
<head>  
  
<meta charset="UTF-8">  
  
<title>Exemplo de CSS</title>  
  
</head>  
  
<body>  
  
<h1 style="color: blue; text-align: center;">Título com CSS Inline</h1>  
  
<p style="font-size: 20px;">Parágrafo com CSS Inline</p>  
  
</body>  
  
</html>
```

# Interno

Estilos são aplicados dentro do documento HTML, entre as tags **<style>** no **<head>**.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8">
<title>Exemplo de CSS</title>
<style>
h1 {
color: green;
text-align: center;
}
p {
font-size: 18px;
}
</style>
</head>
<body>
<h1>Título com CSS Interno</h1>
<p>Parágrafo com CSS Interno</p>
</body>
</html>
```

# Externo

Estilos são aplicados em um arquivo separado (styles.css), referenciado no documento HTML.

```
<!DOCTYPE html>  
  
<html lang="pt-BR">  
  
<head>  
  
<meta charset="UTF-8">  
  
<title>Exemplo de CSS</title>  
  
<link rel="stylesheet" href="styles.css">  
  
</head>  
  
<body>  
  
<h1>Título com CSS Externo</h1>  
  
<p>Parágrafo com CSS Externo</p>  
  
</body>  
  
</html>
```

# Sintaxe CSS

- A sintaxe CSS compõe-se de três partes: um seletor, uma propriedade e um valor.

```
selector {  
    property: value;  
}
```

- O **seletor** é o elemento/tag que definimos (maioria das vezes trata-se de uma tag HTML), a **propriedade** é o atributo que desejamos inserir para este elemento, e cada propriedade tem um **valor** atribuído. A propriedade e o valor são separados por dois pontos ":", delimitados por chaves "{}" e finalizados por ";"

# Sintaxe CSS

- Ex:

```
body {  
    color: #000000;  
}
```

- Se o valor tem múltiplas palavras ele deve ser colocado entre aspas “”.

- Ex:

```
p {  
    font-family: “Comic Sans MS”;  
}
```

# Sintaxe CSS

- Algumas aplicações de **CSS** exigem um conjunto maior de declarações, neste caso é aconselhável definir os seletores um em cada linha.

- Ex:

```
p {  
    text-align: center;  
    color: black;  
    font-family: arial;  
}
```



# Herança

- Herança significa que você pode especificar estilos variados (*classes*) para cada elemento na página, e cada *classe* herdará algumas características de estilo de seu elemento básico ou de seu elemento pai na estrutura do documento HTML.

# Atributo class

- Com o atributo "class" podemos definir diferentes estilos para um mesmo elemento.
- Por exemplo, se quiséssemos ter 2 tipos de parágrafos "<p>" em nosso documento HTML: um alinhado à direita e outro centralizado.

# Atributo class

- Ex:

```
p.direita {text-align: right;}  
p.centro {text-align: center;}
```

- Aplicando os estilos definidos na página HTML:

```
<p class="direita">Este parágrafo está alinhado à direita</p>  
<p class="centro">Este outro parágrafo está centralizado</p>
```

# Atributo ID

- Com o atributo ID podemos definir um estilo único para ser utilizado em muitos elementos.
- Ex:

**#direita {text-align: right;}**

- Aplicando o estilo na página HTML:

**<p id="esquerda">Este parágrafo está alinhado à direita</p>**

**<h3 id="esquerda">Este header também ficará alinhado à direita</h3>**

Obs: o atributo ID deve ter valor único no documento.

# Comentário no CSS

- Podemos inserir comentários entre as definições de estilos para melhor documentar a finalidade de cada uma delas.
- Este procedimento é extremamente saudável visto que provavelmente não lembraremos dos significados e aplicação dos estilos em um futuro próximo.
- O comentário irá ser ignorado pelo Browser. Ele começa com `"/*"`, e termina com `"*/"` :

# Comentário no CSS

```
/* este é um comentário */
```

```
p  
{  
    text-align: left;  
    /* outro comentário */  
    color: blue;  
    font-family: verdana, arial, “sans-serif”;  
}
```

# Regras aplicadas em fontes

- **font-family** - utilizado para indicar qual a fonte que uma *tag* ou documento HTML irá mostrar. De forma geral:

p { font-family: Verdana, Arial, Helvetica }

- Neste exemplo, tudo o que estiver entre as *tag* <p> e </p> no documento HTML será visualizado na fonte *Verdana*.
- A razão de definir mais de uma fonte deve-se ao fato de que nem todos os computadores possuem as mesmas fontes instaladas. No caso, se o computador não possuir a *Verdana* instalada, automaticamente passará para a *Arial*.



# Regras aplicadas em fontes

- **font-size** - utilizado para definir o tamanho das fontes. Existem 3 formas para definição de tamanho:
  - em pontos (pt), pixels (px), cm (cm) e outras unidades de medida;
  - através de palavras-chave;
  - em percentagem (%)

# Regras aplicadas em fontes

- **Ponto** - pode ser usada para definir o tamanho de um elemento em uma página HTML.
- Ex: `p { font-size: 20pt; }` para pontos

# Regras aplicadas em fontes

- **Pixel** é uma medida familiar aos *web designers*. Pode dar uma boa noção do tamanho que a fonte aparecerá na tela.
- **p { font-size: 20px; }** para pixels, por exemplo.

# Regras aplicadas em fontes

- As unidades de medida que podem ser utilizadas são:
  - pt: pontos;
  - px: pixels;
  - in: inches;
  - cm: centímetros;
  - mm: milímetros, e outras.

# Regras aplicadas em fontes

- **font-weight** é o comando **CSS** que controla a propriedade bold (negrito) de uma *tag*:

```
h1 { font-weight: bold; }
```

- No exemplo acima, todas as palavras da página que estiverem compreendidas entre as *tags* `<h1>` e `</h1>` serão mostradas em negrito.

# Regras aplicadas em fontes

- **font-style** é o comando **CSS** que controla a propriedade italic de uma *tag*.

```
h1 { font-style: italic; }
```

- No exemplo acima o browser irá procurar uma versão itálico da fonte para apresentar o texto compreendido entre as *tags* `<h1>` e `</h1>`.

# Regras aplicadas em fontes

- **text-transform** serve para controlar os atributos maiúsculas e minúsculas de um texto:

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

- No exemplo, tudo o que estiver compreendido entre as *tags* <p> e </p> será mostrado em maiúsculas.



# Regras aplicadas em fontes

- **text-decoration** - serve para controlar o sublinhado das palavras.

```
p { text-decoration: underline; }
```

- No exemplo acima todas as palavras em negrito da página aparecerão também sublinhadas.

# Regras aplicadas em parágrafos

- **line-height** - é o comando que controla o espaçamento entre as linhas de um texto. O espaço é definido pela distância entre a base das linhas de um texto.

**p { line-height: 14pt; }**

- Obs: as unidades de medida são as mesmas explicadas para o comando **font-size**: pt, px, in, cm, mm, pc, ex, em.

# Regras aplicadas em parágrafos

- **text-align** é a comando utilizado para o alinhamento de parágrafos em **CSS**.
- Ela só funciona em elementos que definam parágrafos, como <p>, <h1> a <h6>, <blockquote> e <uL>. Os valores que podem ser aplicados são: left, right, center e justify (justificado).

```
p { text-align: justify; }
```

# Regras aplicadas em parágrafos

- **text-indent** é o comando que permite configurar o valor da indentação de um parágrafo. Também só funciona em elementos que definam parágrafos, como os descritos para o text-align.

```
p { text-indent:40pt; }
```

# Padding

A propriedade padding permite que você controle o espaçamento entre o conteúdo e as bordas dos elementos HTML.

Não são válidos valores negativos para espaçamento.

Em declaração única a ordem dos espaçamentos é: superior, direito, inferior e esquerdo.

# Padding

Há quatro modos de se declarar abreviadamente os espaçamentos:

padding: valor1.....**os 4 espaçamentos terão valor1;**

- **padding: 10px:**
  - top, right, bottom e left padding têm 10px;

padding: valor1 valor2.....**espaçamento superior e inferior terão valor1**

**- espaçamento direito e esquerdo terão valor2**

- **padding: 10px 25px:**
  - top e bottom padding têm 10px;
  - right e left padding têm 25px;

# Padding

padding: valor1 valor2 valor3.....**espaçamento superior terá valor1 - espaçamento direito e esquerdo terão valor2 - espaçamento inferior terá valor3**

padding: 10px 25px 30px:

top padding tem 10px;

right e left padding têm 25px;

bottom padding tem 30px;

padding: valor1 valor2 valor3 valor4....**os espaçamentos superiores, direito, inferior e esquerdo nesta ordem.**

padding: 10px 25px 30px 15px;

top padding tem 10px;

right padding tem 25px;

bottom padding tem 30px;

left padding tem 15px;

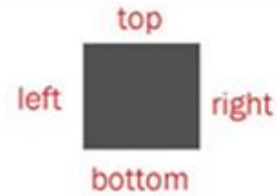


# Margins

## Margin

```
h1 {margin: 5px 10px 5px 10px;}
```

top right bottom left



```
h1 {margin: 5px 10px 5px ;}
```

top right & left bottom

```
h1 {margin: 5px 10px ;}
```

top & bottom right & left

# Border

Responsável por adicionar bordas em qualquer elemento html. Principais propriedades do atributo border:

- border-style:-----estilo da borda
- border-width:-----espessura da borda
- border-color:-----cor da borda
- border-top-width:-----espessura da borda superior
- border-top-style:-----estilo da borda superior
- border-top-color:-----cor da borda superior
- border-right-width:-----espessura da borda direita
- border-right-style:-----estilo da borda direita
- border-right-color:-----cor da borda direita
- border-bottom-width:-----espessura da borda inferior
- border-bottom-style:-----estilo da borda inferior

# Border

- border-top-color:-----cor da borda superior
- border-right-width:-----espessura da borda direita
- border-right-style:-----estilo da borda direita
- border-right-color:-----cor da borda direita
- border-bottom-width:-----espessura da borda inferior
- border-bottom-style:-----estilo da borda inferior
- border-bottom-color:-----cor da borda inferior
- border-left-width:-----espessura da borda esquerda
- border-left-style:-----estilo da borda esquerda
- border-left-color:-----cor da borda esquerda

# Border

- border-top:----- maneira abreviada para todas as propriedades da borda superior
- border-right:---- maneira abreviada para todas as propriedades da borda direita
- border-bottom:--- maneira abreviada para todas as propriedades da borda inferior
- border-left:----- maneira abreviada para todas as propriedades da borda esquerda
- border:----- maneira abreviada para todas as quatro bordas

# Border

Algumas propriedades só aceitam valores específicos, ou seja, temos que respeitar os valores delas ao passar um parâmetro à ela, esses valores estão listados abaixo:

- **color:**
  1. código hexadecimal: #FFFFFF
  2. código rgb: rgb(255,235,0)
  3. nome da cor: red, blue, green...etc
- **style:**
  1. none: nenhuma borda
  2. hidden: equivalente a none
  3. dotted: borda pontilhada
  4. dashed: borda tracejada
  5. solid: borda contínua
  6. double: borda dupla

# Border

7. groove: borda entalhada
8. ridge: borda em ressalto
9. inset: borda em baixo relevo
10. outset: borda em alto relevo

- **width:**

1. thin: borda fina
2. medium: borda média
3. thick: borda grossa
4. length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)