

ciclo formativo

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>ciclo formativo</title>
  </head>
  <body>
    
  </body>
</html>
```



CSS

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>ciclo formativo</title>
  </head>
  <body>
    
  </body>
```



INTRODUÇÃO À CSS

Introdução à CSS

CSS (*Cascading Style Sheets*, ou “Folhas de Estilo e Cascata”) é uma linguagem de marcação de texto responsável por dar estilo à uma página *web*. Enquanto o HTML é responsável por estruturar o esqueleto de uma página, o CSS preenche o corpo dela com cores e formatos.

Em 1995, os cientistas da computação Bert Bos e Håkon Wium perceberam a importância de separar responsabilidades e proporcionar estilo a páginas *web*. Tempos depois, a dupla apresentou a proposta do CSS à W3C, que apoiou a ideia.

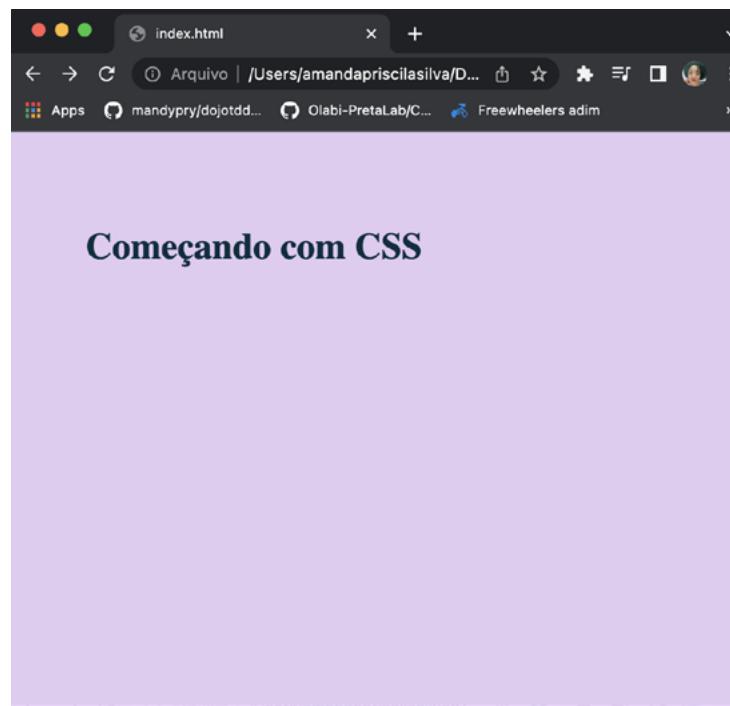
CSS interno, inline e externo

Existem três tipos de CSS para comunicar HTML:

1. **Interno**: confere estilo ao código no cabeçalho do HTML, ou seja, dentro da tag **<head>** com a tag **<style>**;
 - a. No código a seguir, chamaremos nosso primeiro estilo dentro da tag **<head>** (linhas 4 a 12), definiremos a cor de fundo com a tag **body** e outra uma cor para o texto dentro da tag **<h1>**.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <style>
5              body {
6                  background-color: #rgba(162, 85, 213, 0.305);
7              }
8              h1 {
9                  color: #rgb(8, 47, 62);
10                 padding: 60px;
11             }
12         </style>
13     </head>
14
15     <body>
16         <h1>Começando com CSS</h1>
17     </body>
18 </html>
```

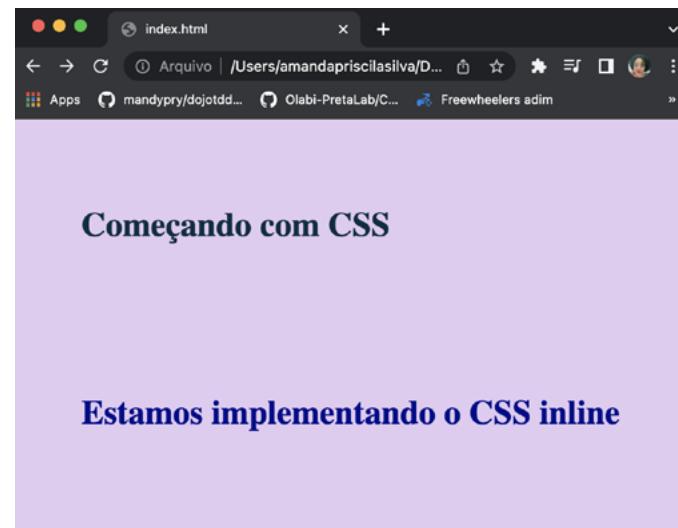
O resultado no nosso navegador será:



- 2. *Inline*:** Possibilita passagem de estilo dentro da própria tag através do atributo *style*;
- Dentro do HTML, vamos inserir outro trecho de código com a tag `<h1>` e passar a cor azul com o atributo *style*, para estilizar nosso segundo trecho de texto.

```
15 |   <body>
16 |     <h1>Começando com CSS</h1>
17 |     <h1 style="color: darkblue;"> Estamos implementando o CSS inline</h1>
18 |   </body>
```

O resultado no navegador será:



3. Externo: melhor e mais recomendado tipo, considera boas práticas para separar responsabilidades de cada código. É preciso criar um arquivo com a extensão **.css** e chamá-lo dentro do arquivo HTML, na tag **<head>**, com a tag **<link />**;

a. Dentro do mesmo repositório (pasta) do nosso arquivo index.html, vamos criar um index.css e passar o código dentro do **<head>** com a tag **<style>**;

```
# style.css U X
pretalab > # style.css > ...
1   body {
2     background-color: □rgba(162, 85, 213, 0.305);
3   }
4   h1 {
5     color: □rgb(8, 47, 62);
6     padding: 60px;
7 }
```

b. Dentro do **HTML**, na tag **<head>**, vamos chamar o arquivo pela tag **<link />**

```
<head>
|   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
```

c. O atributo **rel** diz que o arquivo é responsável pelo estilo da página onde o **type** definiu um texto em CSS. O **href** passa o caminho do arquivo CSS e o resultado no navegador será o mesmo.



Sintaxe básica

Considere a sintaxe abaixo:

```
.classe{  
    propriedade: valor;  
}
```

Sistema de cores

O CSS trabalha com um esquema de cores **nominais**, **hexadecimais** ou **rgb**:

- ➊ Cores **nominais** são passadas com o nome em inglês mesmo no código (por ex., **color: red**);
- ➋ Cores **hexadecimais** são definidas por um código de seis letras ou números precedidos de **#**. Os dois primeiros números representam a intensidade de vermelho; o terceiro e quarto, a intensidade do verde; os dois últimos, a intensidade de azul (cores-base para qualquer outra, como por ex., o código **#FFFFFF**, que representa branco);
- ➌ As cores **rgb** são baseadas no padrão *red*, *green*, *blue*; as tonalidades de cada cor são representadas por um valor entre 0 e 255 (por ex., o código **rgb (255, 255, 255)** representa vermelho).

A propriedade *background color* representa a cor de fundo de um elemento no **CSS**; a propriedade **color**, a cor da fonte de um elemento. Abaixo, o código em HTML, ainda chamando o CSS pela tag **<link>**.

```
<!DOCTYPE html>  
<html>  
    <head>  
        <link rel="stylesheet" type="text/css" href="style.css"/>  
    </head>  
  
    <body>  
        <h1>Esquema de cores em CSS</h1>  
        <h1> Temos um corpo roxo e a fonte rosa claro</h1>  
    </body>  
</html>
```

Abaixo, código CSS no qual definimos o corpo da nossa página (roxo, com texto rosa).

```
body {
    background-color: #6959CD;
}
h1 {
    color: #FFB6C1;
}
```

O resultado na tela:



Medidas absolutas

Usamos **pixel**, **percentagem**, **unidade de comprimento relativo (em)** e **vh** para definir medidas absolutas em atributos de CSS:

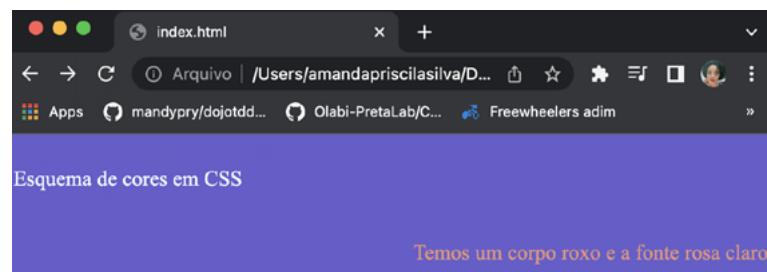
- ➊ **Pixels** (px) variam de acordo com o dispositivo de visualização. Para dispositivos de baixo ponto por polegada, 1px vale como um pixel de dispositivo (ponto) de tela; para impressoras e telas de alta resolução, 1px representa vários *pixels* de dispositivo;
- ➋ **Percentagem (%)** é definida em relação ao comprimento pai. Se a largura do pai for 800px, 50% representam 400px;
- ➌ **Unidades de comprimento relativo (em)** especificam comprimento relativo a outra propriedade de comprimento. As unidades de comprimento relativo escalam melhor entre diferentes meios de renderização;
- ➍ As medidas **vh** e **vw** correspondem a 1/100 da altura e largura da *viewport*, respectivamente. Se a altura do navegador for 900px, 1vh equivale a 9px; se a largura da *viewport* for 750px, 1vw equivale a 7.5px.

Alinhamento

É possível alinhar elementos com CSS na horizontal ou na vertical usando algumas propriedades. Como exemplo, vamos usar o atributo *text-align* em nosso CSS, alinhando o título do texto no lado esquerdo e a frase, no direito.

```
#titulo {
    color: #azure;
    text-align: left;
}

.frase {
    color: #darksalmon;
    text-align: right;
}
```



Fontes

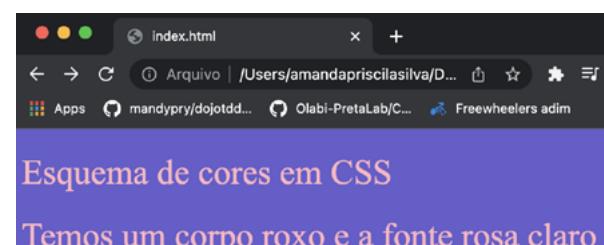
É possível categorizar fontes de diversas formas definindo uma lista, a ser usada na ordem da última fonte da lista. As fontes deverão ser genéricas dentre cinco disponíveis em CSS: ***serif***, ***sans-serif***, ***monospace***, ***cursive*** ou ***fantasy***.

O atributo ***font-family*** altera o tipo de fonte junto ao CSS:

```
h1 {
    color: #FFB6C1;
    font-family: 'Times New Roman';
}
```

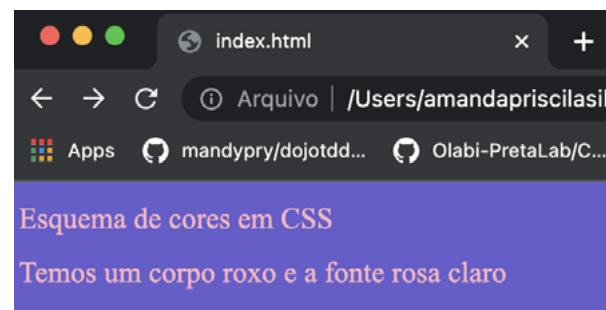
O atributo ***font-weight*** altera a espessura da fonte com o CSS:

```
h1 {
    color: #FFB6C1;
    font-family: 'Times New Roman';
    font-weight: 100;
}
```



O atributo ***font-size*** altera o tamanho da fonte com o CSS:

```
h1 {
    color: #FFB6C1;
    font-family: 'Times New Roman';
    font-weight: 100;
    font-size: large;
}
```

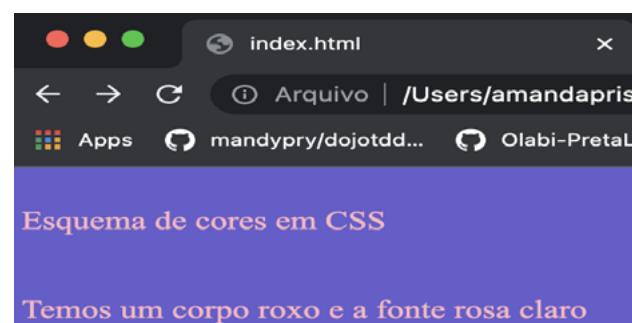


O atributo ***line-height*** define a distância entre as linhas de texto com o CSS:

```

h1 {
    color: #FFB6C1;
    font-family: 'Times New Roman';
    font-weight: 100;
    font-size: large;
    line-height: 50px;
}

```



Identificadores e classes

É possível usar identificadores representados pela palavra “id” em *tags* HTML desde que chamando-os no CSS precedidos por # (cerquilha). Também é possível chamar classes (representadas pela palavra “class”), desde que precedidas por . (ponto). No HTML abaixo, passamos uma ***id*** e uma ***class*** em cada uma das *tags* ***<h1>***:

```

<body>
    <h1 id="titulo">Esquema de cores em CSS</h1>
    <h1 class="frase"> Temos um corpo roxo e a fonte rosa claro</h1>
</body>

```

Criamos um bloco para cada uma no CSS:

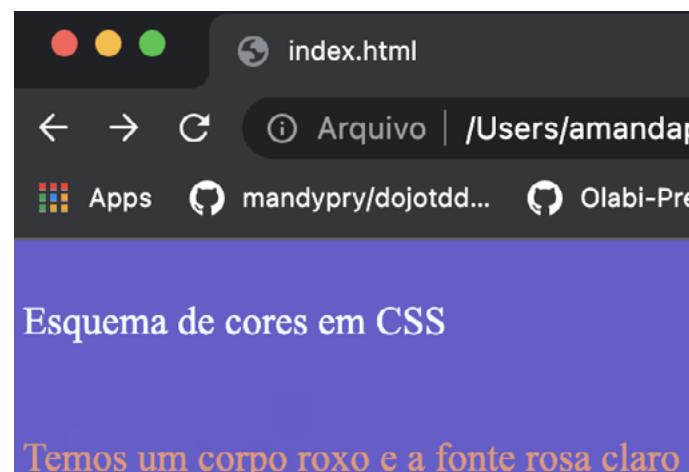
```

#titulo {
    color: azure;
}

.frase {
    color: darksalmon;
}

```

O resultado na tela:



Altura e largura

Usamos as propriedades *width* (largura) e *height* (altura) para definir a dimensão de qualquer elemento. Elas podem representar valores em comprimento (px, mm etc.), percentagem (%) ou mesmo nenhum valor (o padrão). As propriedades *min-width* e *min-height* são usadas para definir largura e altura mínimas do elemento; *max-width* e *max-height*, para definir largura e altura máximas; quando o valor for nenhum, não há largura e altura máximas definidas.

Bordas

A propriedade *border* (borda) é usada para criar uma borda para o elemento. Também é possível alterar cor, largura e estilo da borda.

Algumas propriedades:

- ⌚ ***border***: forma abreviada para as quatro bordas;
- ⌚ ***border-width***: define a espessura da borda;
- ⌚ ***border-style***: define o estilo da borda;
- ⌚ ***border-color***: define a cor da borda;
- ⌚ ***border-top***: borda superior;
- ⌚ ***border-right***: borda direita;
- ⌚ ***border-bottom***: borda inferior;
- ⌚ ***border-left***: borda esquerda.

No HTML abaixo, atribuímos uma classe e um id a duas tags **<h1>**:

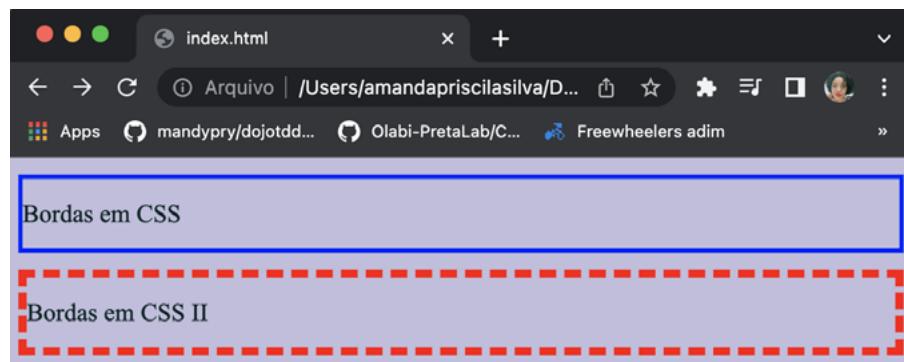
```
<body>
  <h1 id="borda">Bordas em CSS</h1>
  <h1 class="frase">Bordas em CSS II</h1>
</body>
```

No CSS, usamos algumas propriedades de borda:

```
#borda {
  color: □rgb(4, 37, 37);
  border-width: medium;
  border-style: solid;
  border-color: □#00f;
}

.frase {
  color: □rgb(4, 37, 37);
  border-width: 6px;
  border-style: dashed;
  border-color: □#f00;
}
```

O resultado no navegador:

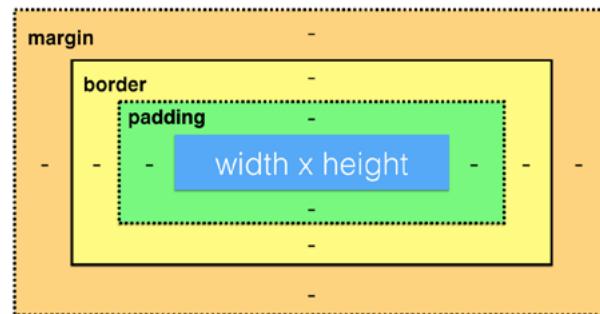


Espaçamento

A propriedade *padding* (espaçamento) define a distância entre o conteúdo de um elemento e suas bordas conforme abaixo:

- ***padding***: define espaçamento nos 4 lados;
- ***padding-top***: define espaçamento superior;
- ***padding-right***: define espaçamento à direita;
- ***padding-bottom***: define espaçamento inferior;
- ***padding-left***: define espaçamento à esquerda.

Na imagem abaixo, a parte azul que define a largura e altura dimensão do elemento e agora vamos focar na parte verde que é o preenchimento da parte azul.



Margem

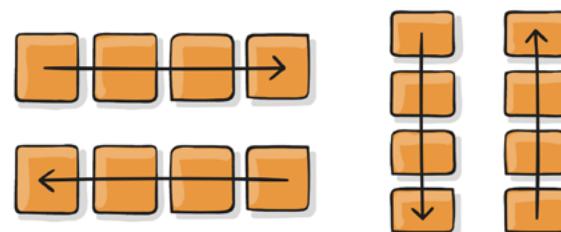
A propriedade *border* representa o espaço em torno de um elemento que o separa dos demais. Podemos definir o valor da margem em comprimento, percentagem, automático e herança. O primeiro passo para escrevermos códigos em HTML é definir a ferramenta a ser utilizada.

Direção flexível e propriedades

(*display, float, clear, clear fix*)

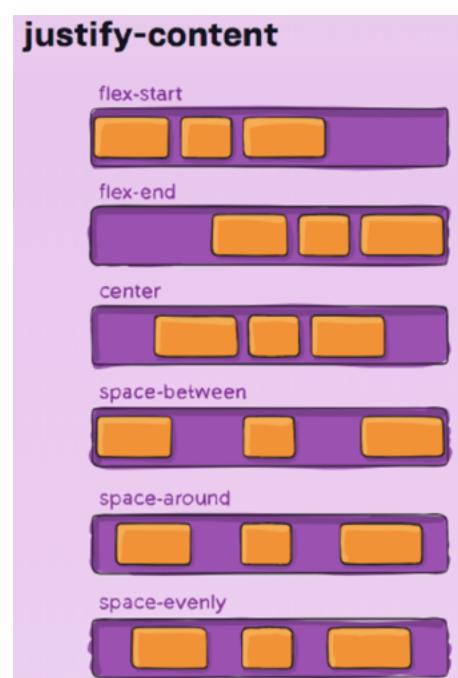
Para definir ícones dentro de um container (caixa em torno de vários itens), temos as seguintes opções:

Flex-direction define como itens serão colocados no container flexível, via eixo principal e direção (normal ou invertida);

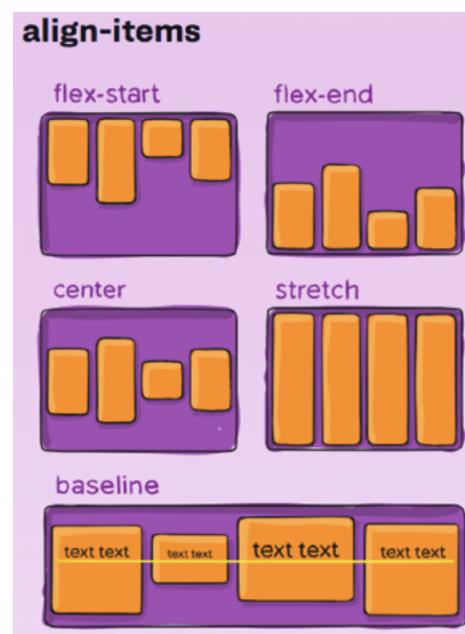


Justify-content define o eixo principal e, como consequência, a direção de seus *flex-items*, o que também permite alterar a ordem dos itens que exigiam a alteração do HTML subjacente.

Propriedades: *flex-start, flex-end, center, space-between, space-around, space-evenly*;



Align-items define o valor de todos os filhos diretos como um grupo dentro de um container. Propriedades: *flex-start*, *flex-end*, *center*, *stretch* e *baseline*;



- ④ **Display** define a organização e o tipo de caixa renderizada em uma página web de forma organizada em *inline* ou bloco, de acordo com o valor fornecido, o que permite contexto flexível para todos os filhos diretos. O *display* trabalha como uma tabela na qual o conteúdo pode ser incluído em linhas e colunas;
- ④ **Float** permite que um item seja colocado ao longo do lado direito ou esquerdo do container; textos e elementos em linha, por conseguinte, se posicionam ao seu redor;
- ④ **Clear**, em conjunto com *float*, é a propriedade usada quando o próximo elemento precisa ficar abaixo e não ao lado, como se fosse *float*;
- ④ **Clear fix** permite igualar o espaçamento entre elementos dentro do container, caso algum seja mais alto que o outro;

Sem Clearfix

 Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Phasellus imperdiet, nulla et
 dictum interdum...



Com Clearfix

 Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Phasellus imperdiet, nulla et
 dictum interdum...

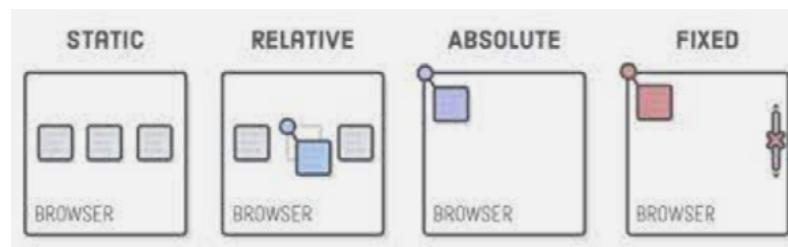


Fonte: [w3school](http://w3school.com)

Posicionamento

Parte crucial na criação de páginas *web*, de maneira que se faz necessário conhecer e entender as propriedades que posicionam elementos dentro da página:

- ➊ A propriedade **position** recebe os valores **static, relative, absolute** e **fixed** para posicionar elementos;
- ➋ A propriedade **static** representa o valor *default* (padrão) de toda página HTML seguindo seu fluxo comum;
- ➌ A propriedade **relative** torna possível para a propriedade *position* aceitar *top, bottom, left* e *right* (topo, inferior, esquerda e direita, respectivamente) - estas responsáveis por alterar o posicionamento do elemento;
- ➍ A propriedade **absolute** posiciona qualquer elemento com *position* diferente de *static* de acordo com o elemento pai;
- ➎ A propriedade **fixed** é semelhante à *absolute*; a diferença é que nela, a referência passa a ser a janela do navegador, ou seja, a área que aparece para o usuário independente de barra de rolagem.



Referências:

W3School: [w3school CSS](#)

DevMedia: [devmedia](#)

Developer mozilla: [Developer mozilla CSS](#)

ciclo formativo

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>ciclo formativo</title>
  </head>
  <body>
    
    <h1>CSS</h1>
    <h2>Ciclo Formativo Básico</h2>
    <h3>em Tecnologia PretaLab</h3>
  </body>
</html>
```



```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>ciclo formativo</title>
  </head>
  <body>
    
    <h1>CSS</h1>
    <h2>Ciclo Formativo Básico</h2>
    <h3>em Tecnologia PretaLab</h3>
  </body>
</html>
```



UM PROGRAMA DE



COM APOIO DE

