



HTML

Introdução

O que é HTML?

Definição

O HTML é uma linguagem de marcação, ela fornece o esqueleto de toda página web. HTML (Linguagem de Marcação de Hipertexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web.



Estrutura

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Estrutura

<!DOCTYPE html>

Informa ao navegador que esse documento é do tipo HTML e sua versão. Quando está escrito apenas html, indica que é a mais recente...

<html>

Representa a raiz do documento, serve com um container que engloba todos os outros elementos HTML.

<body>

É onde fica todo o conteúdo de texto, imagem e vídeos, em que o usuário vê e interage, nele os conteúdos são estruturados pelas demais tags do HTML.

Estrutura

<script>

Esse elemento contém instruções de script ou aponta para um arquivo de script externo por meio do atributo src.

<head>

Compreende as informações do documento que serão interpretadas pelo navegador (metadados). Como por exemplo, título do documento, links para folhas de estilo etc.

<meta>

Define metadados, ou seja, informações sobre dados de um documento HTML. As <meta> tags vão dentro do elemento <head> e são usadas para especificar o conjunto de caracteres, o autor do documento, as configurações da janela de visualização etc.

Estrutura

<link>

É uma tag vazia, que contém apenas atributos e faz a relação do documento HTML com recursos externos, é comumente usado para vincular uma folha de estilo externa, também é usada para definir o favicon da página (ícone da aba do navegador), como outros recursos.

<style>

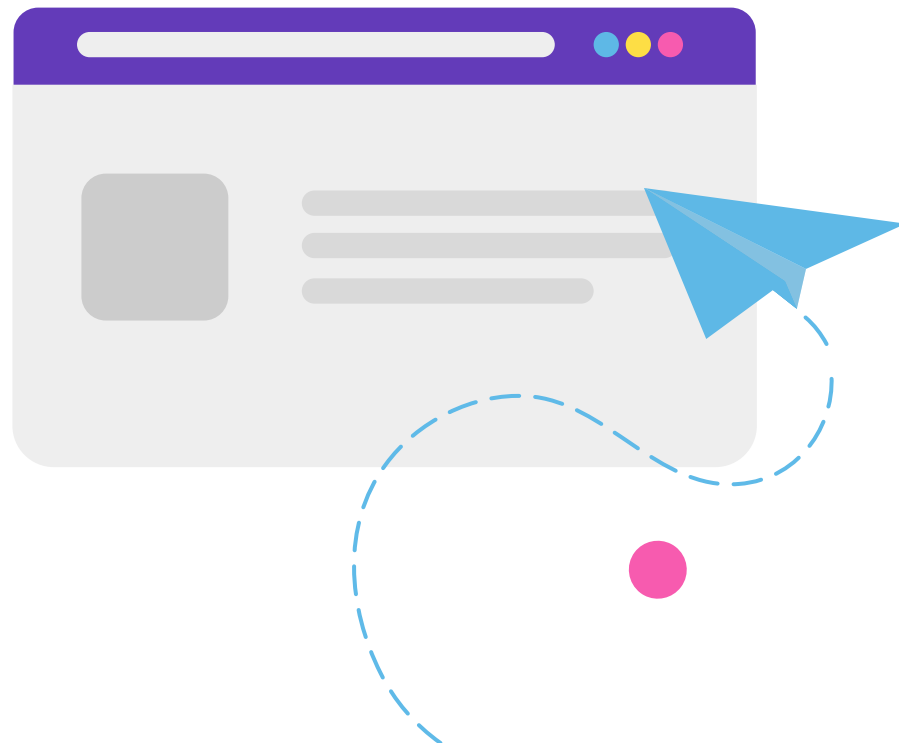
Essa tag é usada para declarar estilos (CSS) para um documento.

Parágrafos

Definição

O elemento HTML **<p>** (ou Elemento HTML Parágrafo) representa um parágrafo do texto.

<p> Isso é um parágrafo **</p>**



Cabeçalhos

Definição

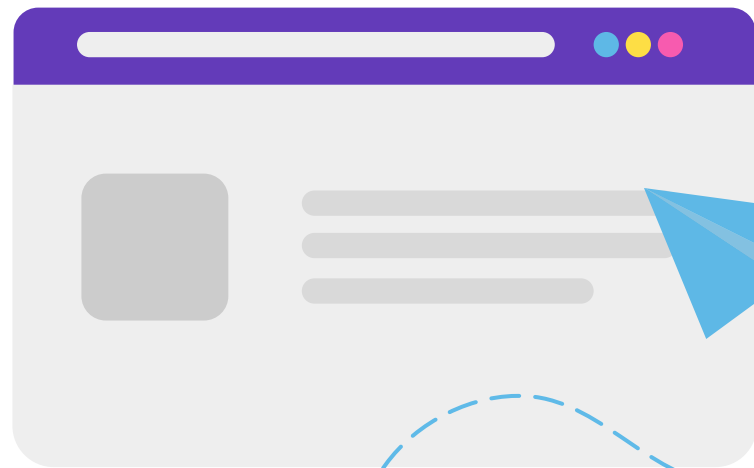
Em HTML o título é exatamente o que parece: um título real ou subtítulo, quando você coloca no texto a tag **<h1>**, o texto crescerá e a dimensão das letras será de acordo com o número de cabeçalhos. Os títulos podem ter dimensões de 1 a 6, onde 1 é a menor dimensão e 6 a maior.

<h1> Cabeçalhos **</h1>**
<h2> são **</h2>**
<h3> ótimos **</h3>**
<h4> para **</h4>**
<h5> títulos e **</h5>**
<h6> subtítulos **</h6>**

Formatação de textos

Definição

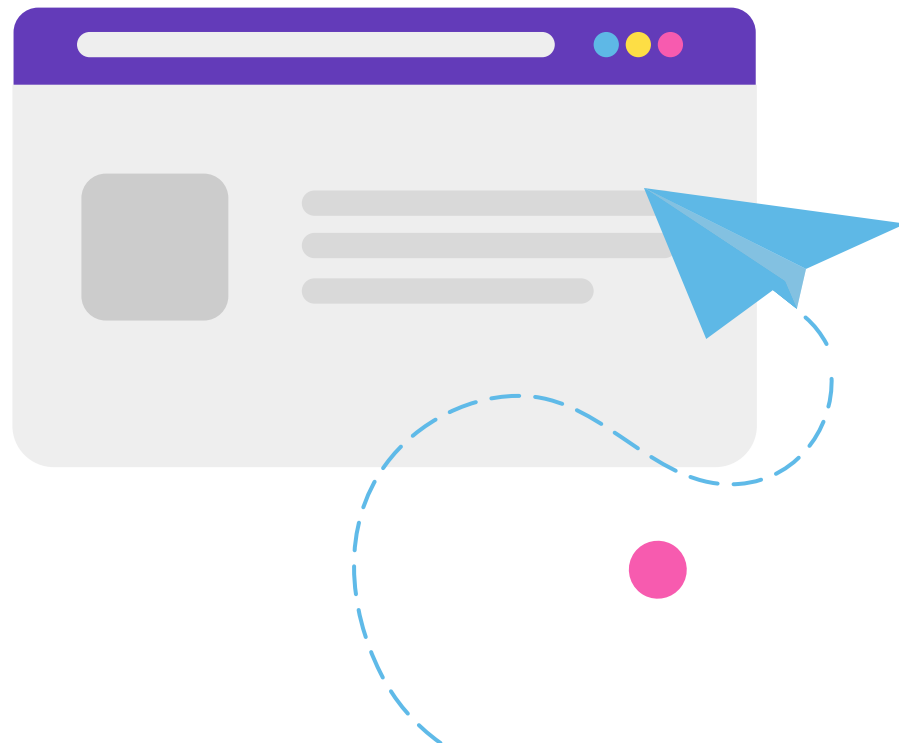
A medida que você vai colocando mais textos na sua página, precisará de elementos de formatação para ela. Essas tags de formatação podem fazer os elementos negritos, itálicos, sublinhados, mas não só isso.



Formatação de textos

Tags

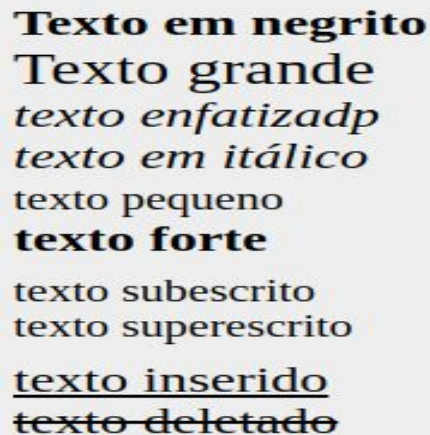
- «**b**» Define texto em negrito
- «**big**» Define texto grande
- «**em**» Define texto enfatizado
- «**i**» Define texto em itálico
- «**small**» Define texto pequeno
- «**strong**» Define texto forte
- «**sub**» Define texto subescrito
- «**sup**» Define texto superescrito
- «**ins**» Define texto inserido
- «**del**» Define texto cancelado



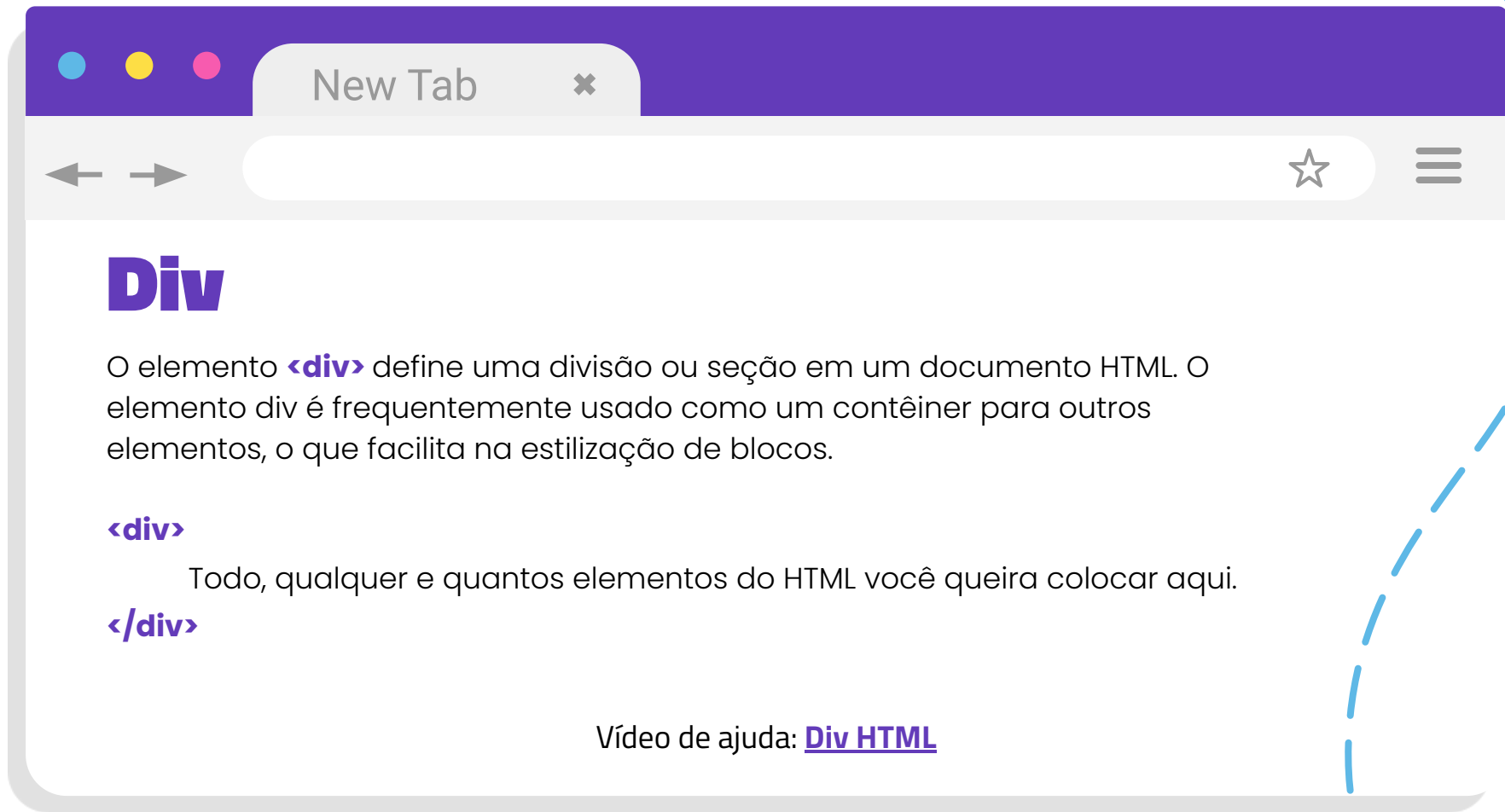
Formatação de textos

Exemplos

**** texto em negrito ****
<big> texto grande **<big/>**
**** texto enfatizado ****
<i> texto em itálico **<i/>**
<small> texto pequeno **<small/>**
**** texto forte ****
<sub> texto subescrito **<sub/>**
<sup> texto superescrito **<sup/>**
<ins> texto inserido **<ins/>**
**** texto cancelado ****



Texto em negrito
Texto grande
texto enfatizado
texto em itálico
texto pequeno
texto forte
texto subescrito
texto superescrito
texto inserido
~~texto deletado~~



Div

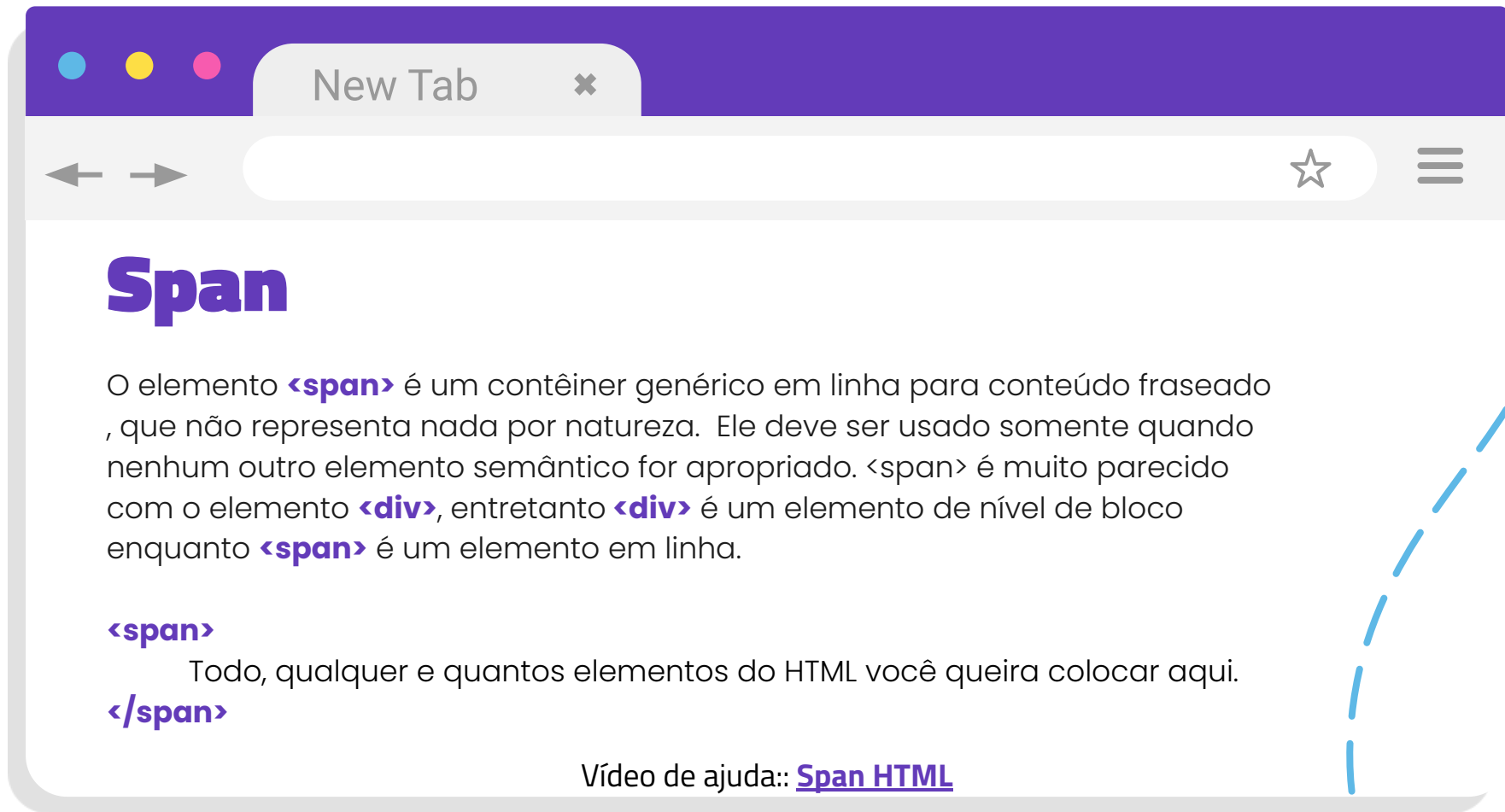
O elemento `<div>` define uma divisão ou seção em um documento HTML. O elemento div é frequentemente usado como um contêiner para outros elementos, o que facilita na estilização de blocos.

`<div>`

Todo, qualquer e quantos elementos do HTML você queira colocar aqui.

`</div>`

Vídeo de ajuda: [Div HTML](#)



Span

O elemento **** é um contêiner genérico em linha para conteúdo fraseado , que não representa nada por natureza. Ele deve ser usado somente quando nenhum outro elemento semântico for apropriado. **** é muito parecido com o elemento **<div>**, entretanto **<div>** é um elemento de nível de bloco enquanto **** é um elemento em linha.

Todo, qualquer e quantos elementos do HTML você queira colocar aqui.

Vídeo de ajuda:: [Span HTML](#)

Listas

As listas são muito importantes quando queremos listar alguns itens no site e também para a criação de menu de navegação.

Todo elemento de uma lista deve estar entre a tag **** (List Item).

Não Ordenadas

Ordenadas

<dl>

Por definição

Vídeo de ajuda: [Listas HTML](#)

Lista Não Ordenada

As Listas não ordenadas são iniciadas com a tag `` elas não possuem uma ordem previamente definida.

``

`Nome`

`Telefone`

`Casa`

`Estado`

``

Como fica na página:

- Nome
- Telefone
- Casa
- Estado

Ordenada

Uma lista ordenada começa com a tag ``.
Por padrão, as listas são ordenadas por números.

``

`Nome`

`Telefone`

`Casa`

`Estado`

``

Como fica na página:

1. Nome
2. Telefone
3. Casa
4. Estado

Ordenada

Podemos ordenar também por a,b,c,d, etc.
Caso queira que seja ordenada por letras maiúsculas, basta em colocar **<ol type="a">**.

type indica o tipo de numeração:

- **'a'** indica letras minúsculas,
- **'A'** indica letras maiúsculas,
- **'i'** indica algarismos romanos minúsculos,
- **'I'** indica algarismos romanos maiúsculos,
- e **'1'** indica números (padrão).

Por definição

Já as Listas por definição são um pouco mais diferentes, elas são representadas pela tag `<dl>`, seguido de `<dt>` e `<dd>`.

`<dl>`

`<dt>`HTML`</dt>`

`<dd>`HTML Básico`</dd>`

`<dd>`HTML Avançado`</dd>`

`<dt>`PHP`</dt>`

`<dd>`PHP Básico`</dd>`

`<dd>`PHP Intermediário`</dd>`

`<dd>`PHP Avançado`</dd>`

`</dl>`

Como fica na página:

HTML

HTML Básico

HTML Avançado

PHP

PHP Básico

PHP Intermediário

PHP Avançado

Imagens



como usar

O elemento `` representa a inserção de imagem no documento.

```

```

TAG

``

Tag para inserir a imagem

Atributos

ALT

`alt="Minha Figura"`

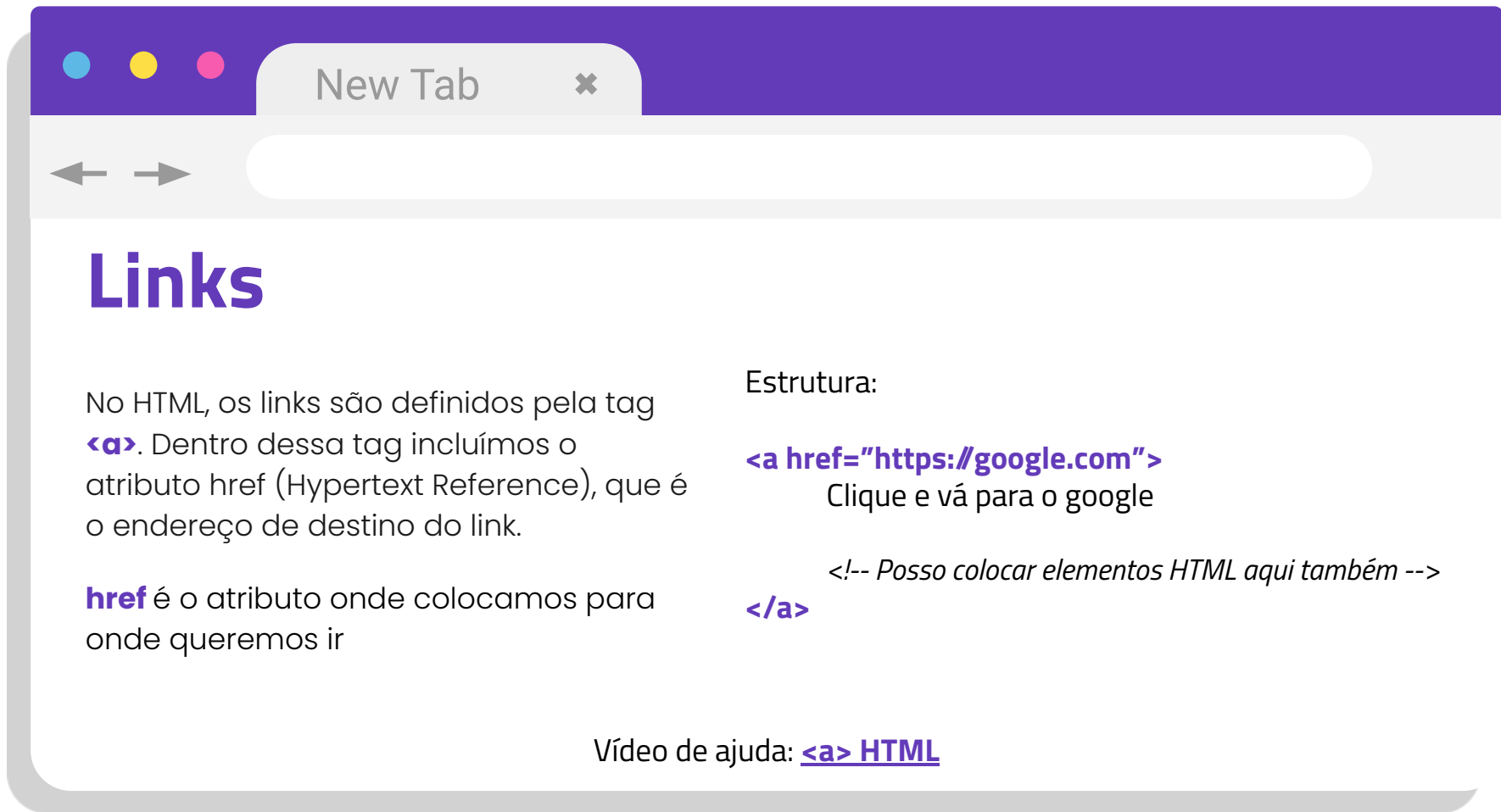
Texto caso a imagem falhe

SRC

`src="minhaImagem.jpg"`

Define o caminho da imagem

Vídeo de ajuda: [Imagem HTML](#)



Links

No HTML, os links são definidos pela tag **<a>**. Dentro dessa tag incluímos o atributo href (Hypertext Reference), que é o endereço de destino do link.

href é o atributo onde colocamos para onde queremos ir

Estrutura:

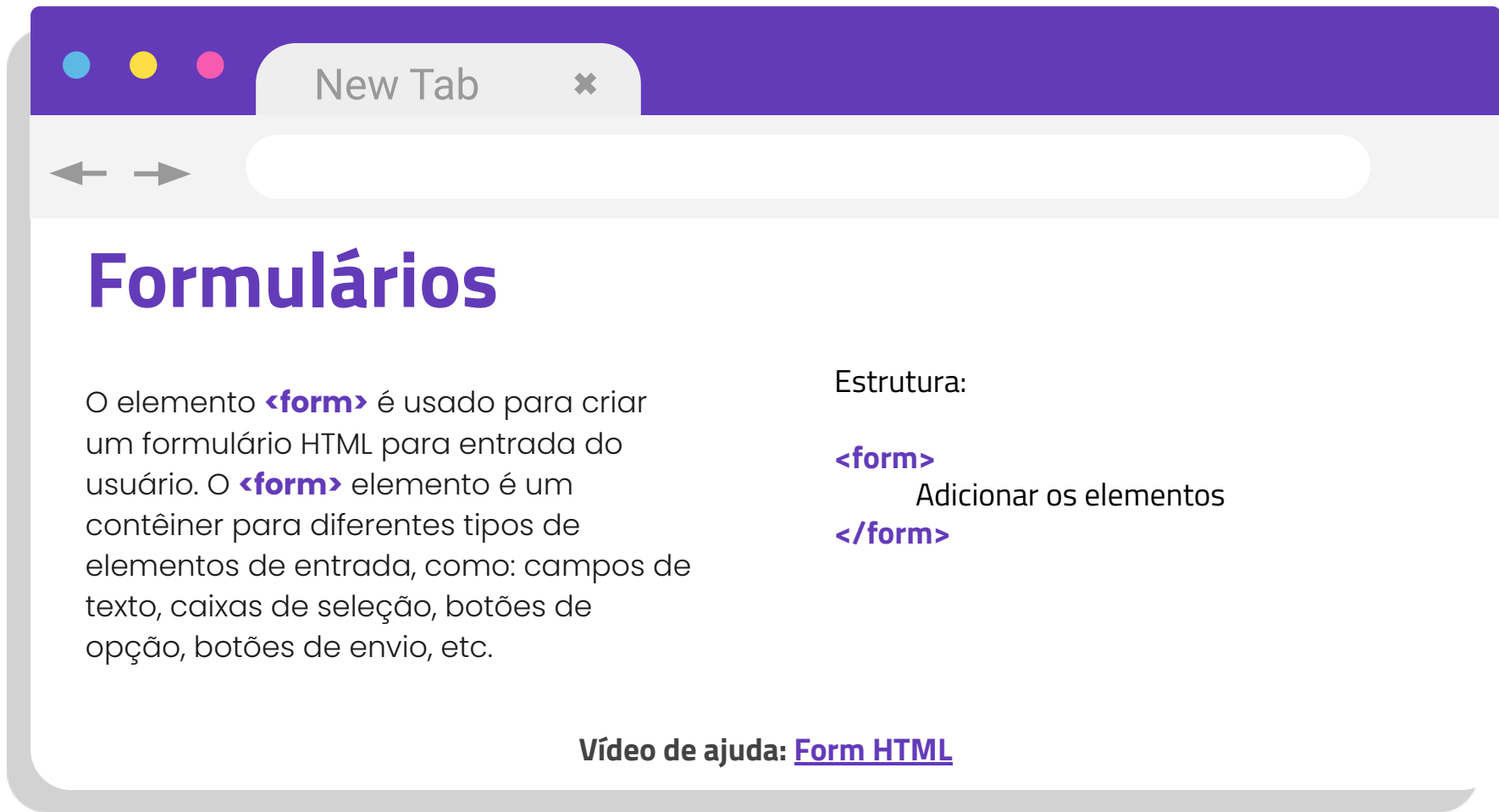
```
<a href="https://google.com">
```

Clique e vá para o google

<!-- Posso colocar elementos HTML aqui também -->

```
</a>
```

Vídeo de ajuda: [<a> HTML](#)



Formulários

O elemento **<form>** é usado para criar um formulário HTML para entrada do usuário. O **<form>** elemento é um contêiner para diferentes tipos de elementos de entrada, como: campos de texto, caixas de seleção, botões de opção, botões de envio, etc.

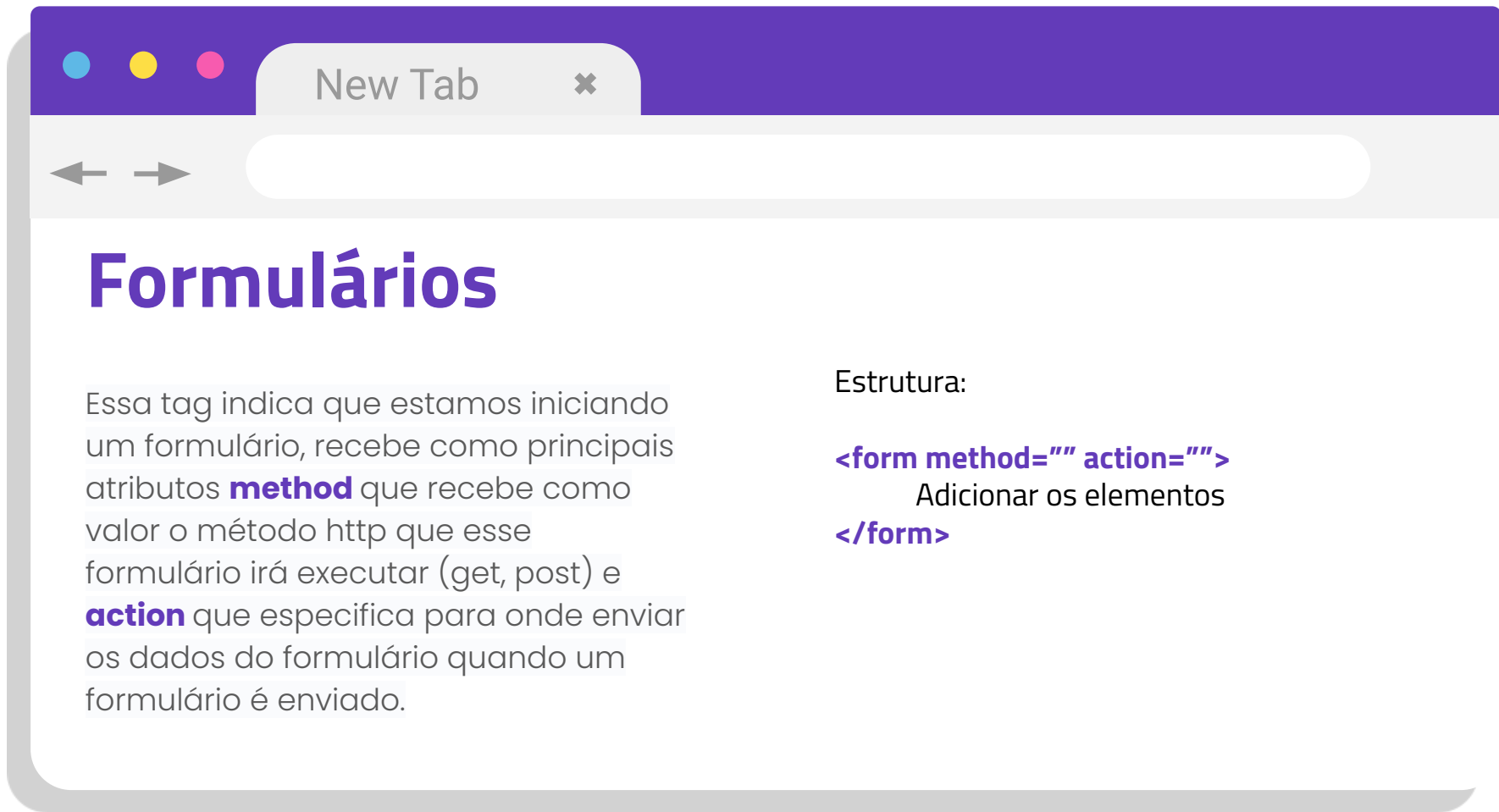
Estrutura:

<form>

Adicionar os elementos

</form>

Vídeo de ajuda: [Form HTML](#)

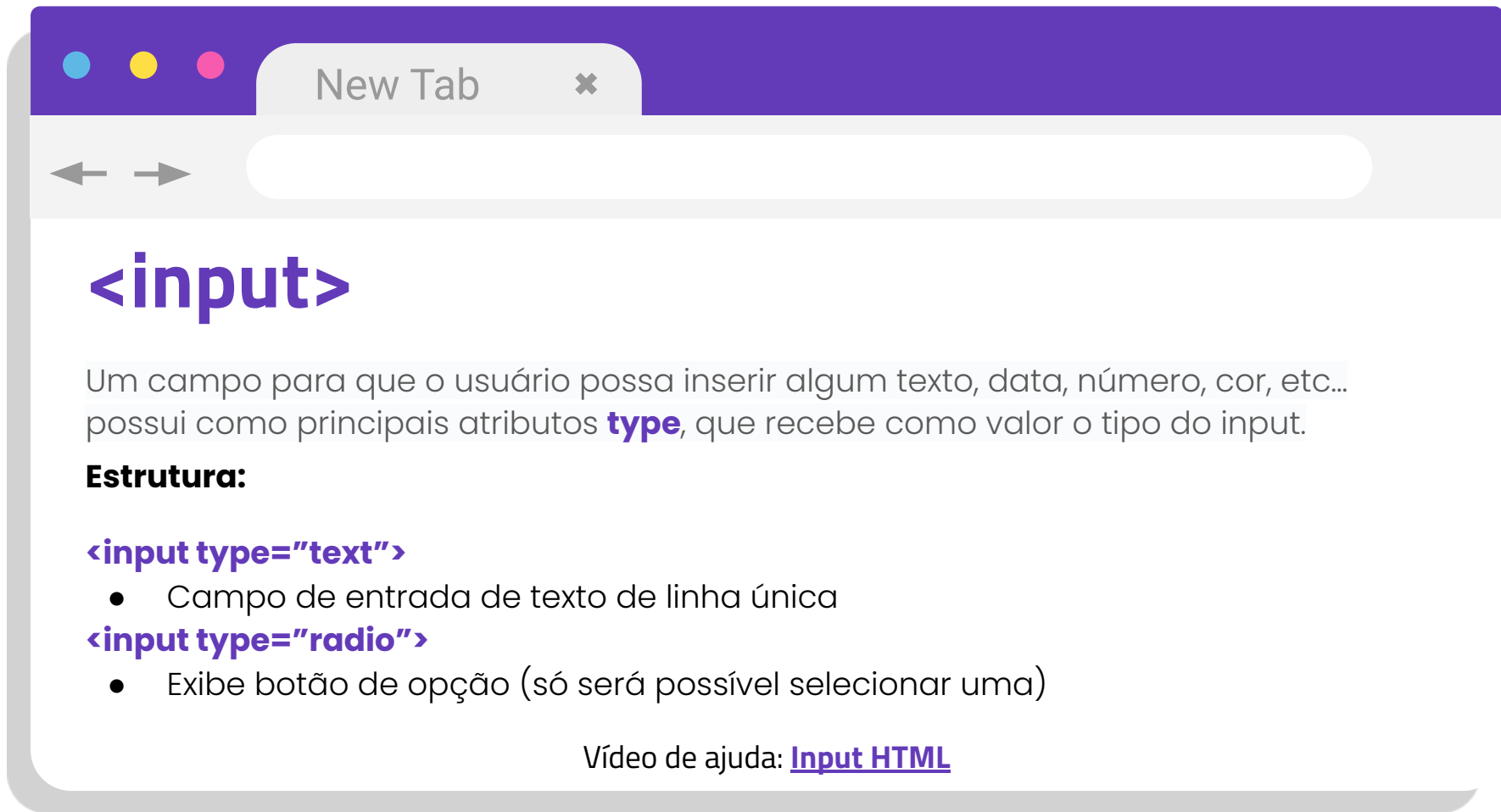


Formulários

Essa tag indica que estamos iniciando um formulário, recebe como principais atributos **method** que recebe como valor o método http que esse formulário irá executar (get, post) e **action** que especifica para onde enviar os dados do formulário quando um formulário é enviado.

Estrutura:

```
<form method="" action="">  
    Adicionar os elementos  
</form>
```



`<input>`

Um campo para que o usuário possa inserir algum texto, data, número, cor, etc... possui como principais atributos **type**, que recebe como valor o tipo do input.

Estrutura:

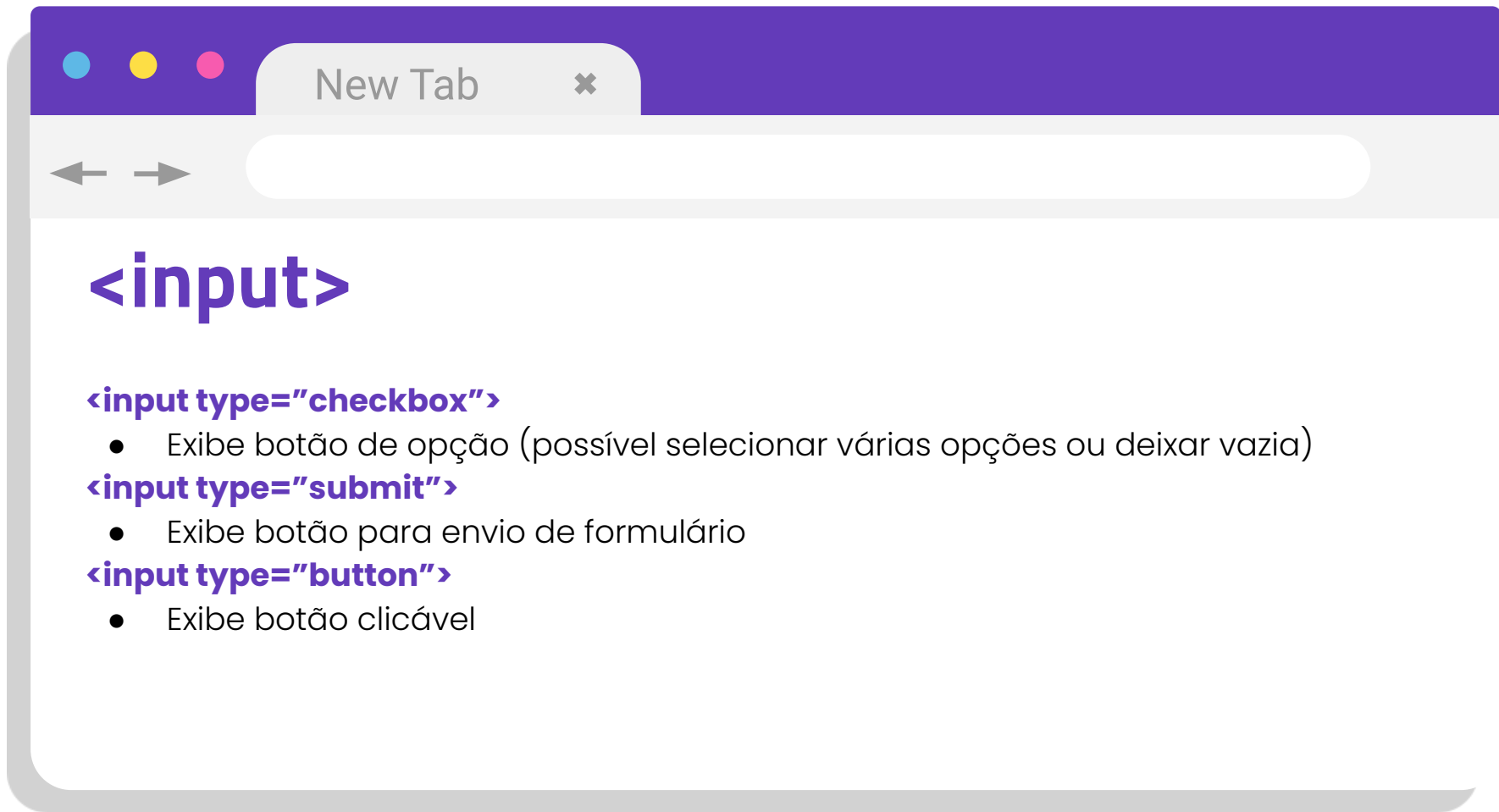
`<input type="text">`

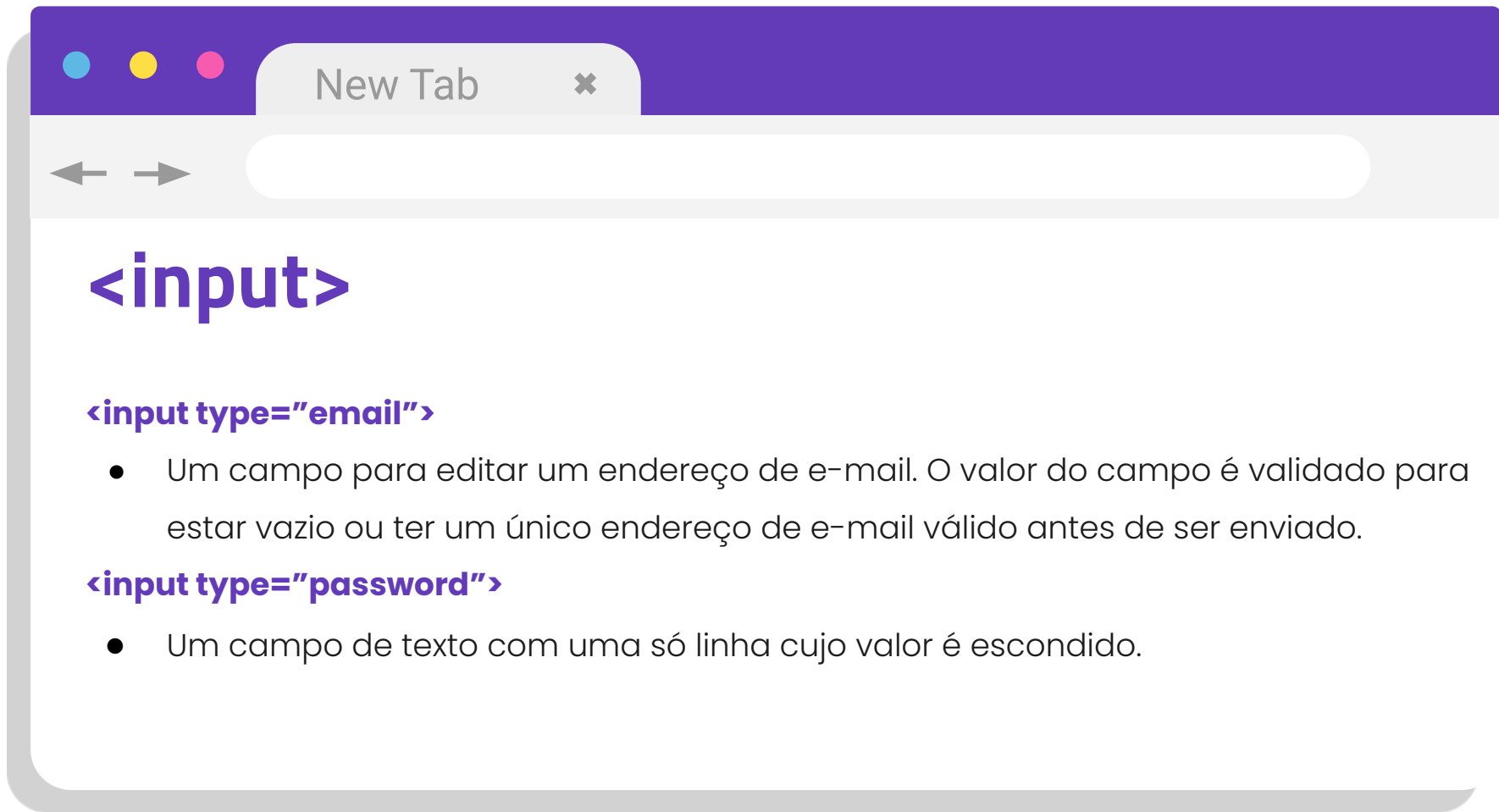
- Campo de entrada de texto de linha única

`<input type="radio">`

- Exibe botão de opção (só será possível selecionar uma)

Vídeo de ajuda: [Input HTML](#)





Tabelas

Definição

As tabelas são listas de dados em duas dimensões e são compostas por linhas e colunas.

A tag utilizada para criar uma tabela HTML é a tag **<table>**, posteriormente fechada com **</table>**

As tags que vão formar a estrutura básica de uma tabela em HTML são as tags **<tr>** e **<td>**.

A tag **<tr>** representa uma linha e a tag **<td>** representa uma célula.

<th> representam títulos

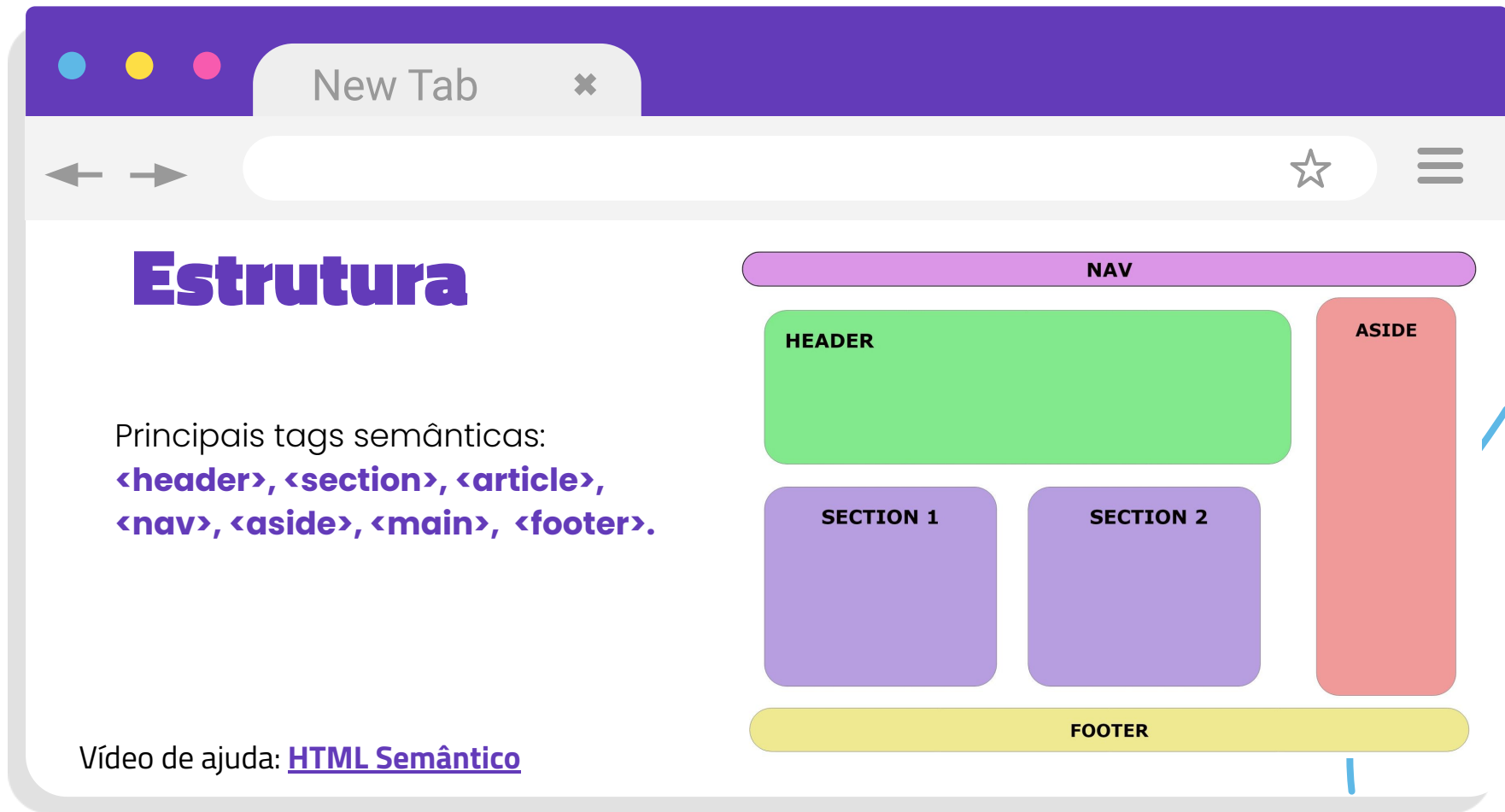
```
<table>
  <tr>
    <th> Turma 1 <th/>
    <th> Turma 2 <th/>
    <th> Turma 3 <th/>
  <tr/>
  <tr>
    <td> Débora <td/>
    <td> Jailson <td/>
    <td> Gabriel <td/>
  <tr/>
</table/>
```

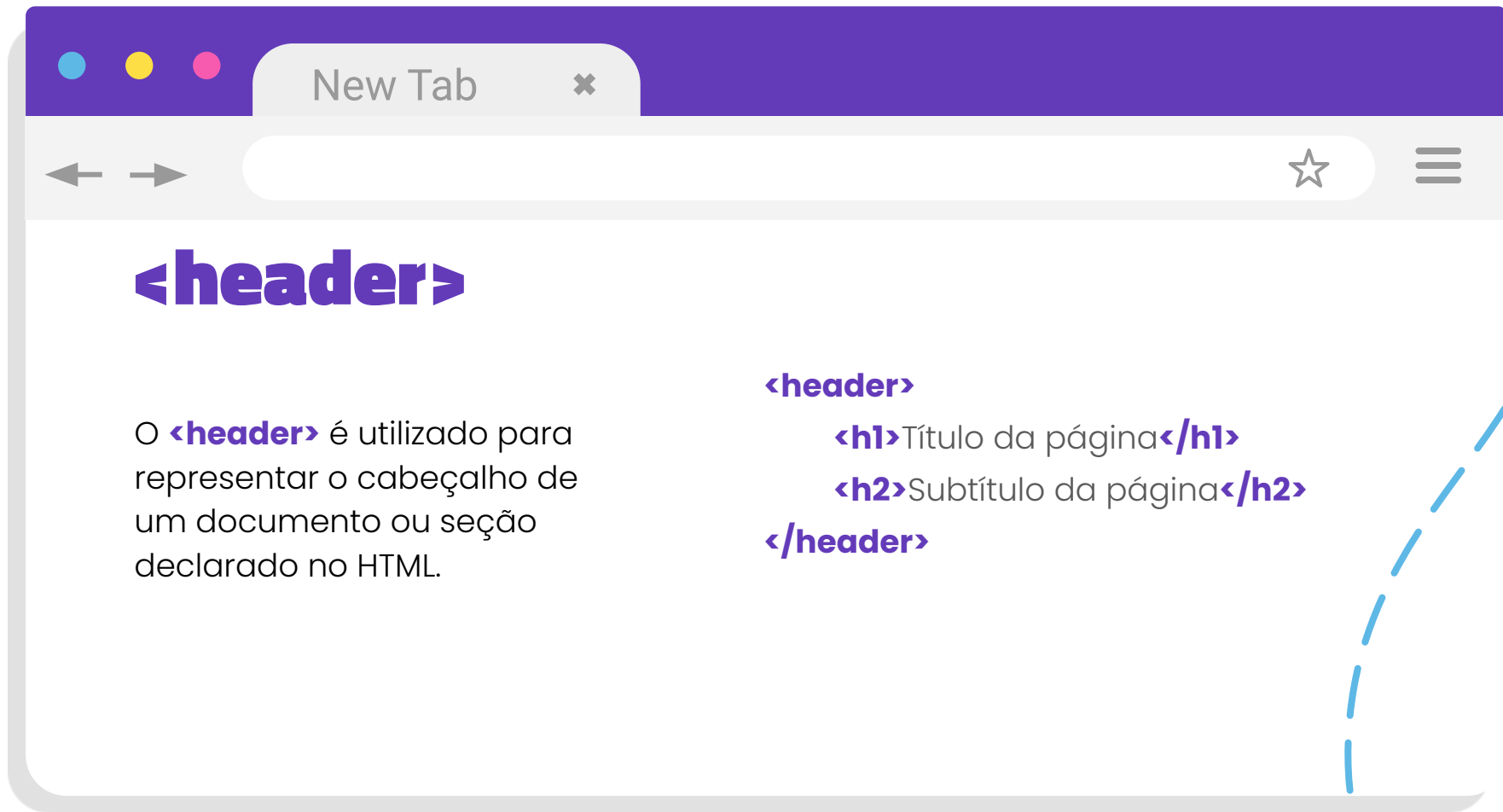


New Tab

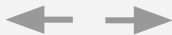
HTML Semântico

O HTML semântico é a forma de deixar o site com suas informações bem explicadas e compreensíveis para o computador, ajudando até mesmo em sua busca no Google e facilitando o entendimento de leitores de acessibilidade. Com o HTML semântico, ficou muito mais fácil de interpretar páginas.





New Tab



<header>

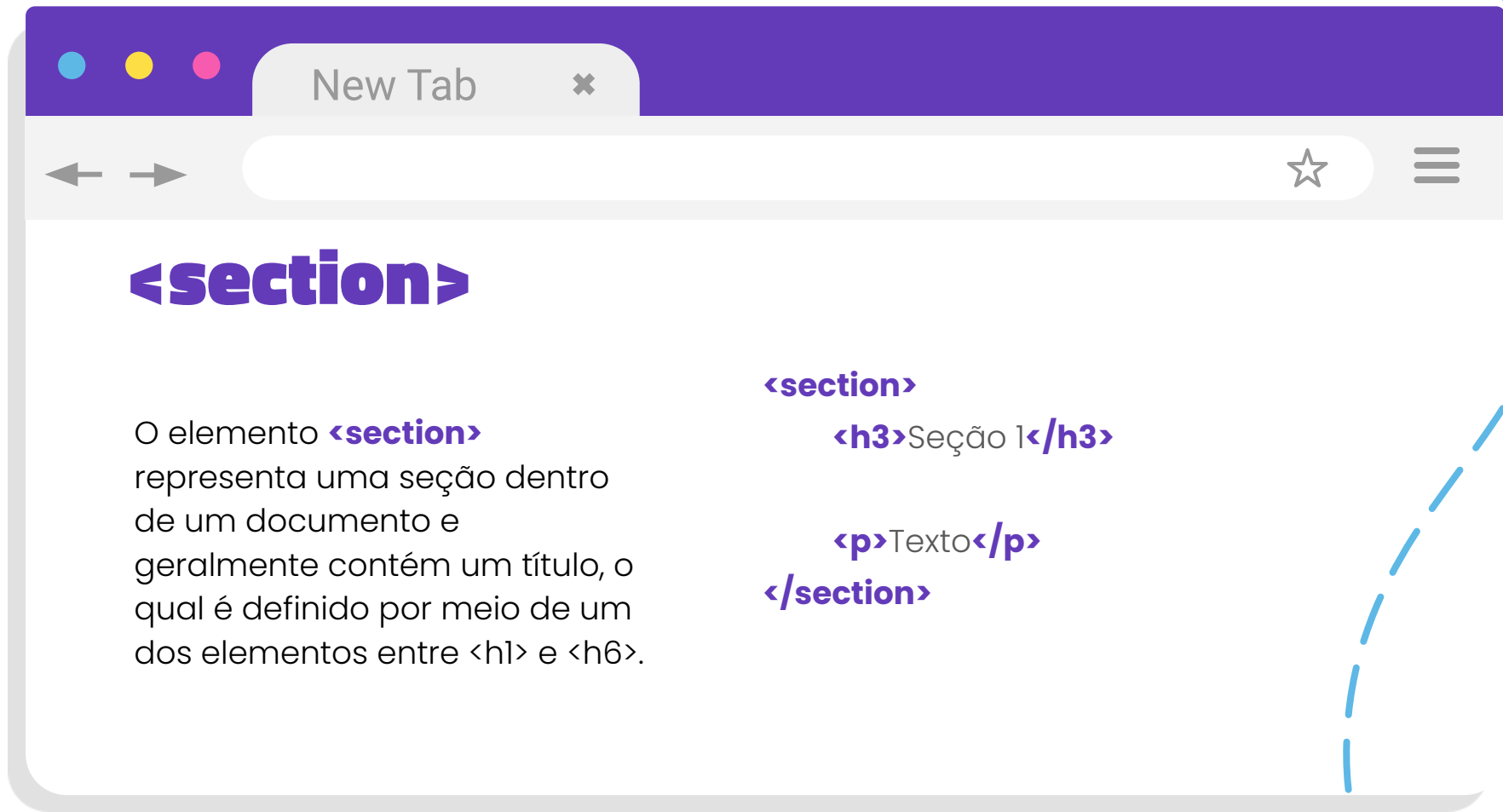
O **<header>** é utilizado para representar o cabeçalho de um documento ou seção declarado no HTML.

<header>

<h1>Título da página**</h1>**

<h2>Subtítulo da página**</h2>**

</header>



<section>

O elemento **<section>** representa uma seção dentro de um documento e geralmente contém um título, o qual é definido por meio de um dos elementos entre `<h1>` e `<h6>`.

<section>

<h3>Seção 1**</h3>**

<p>Texto**</p>**

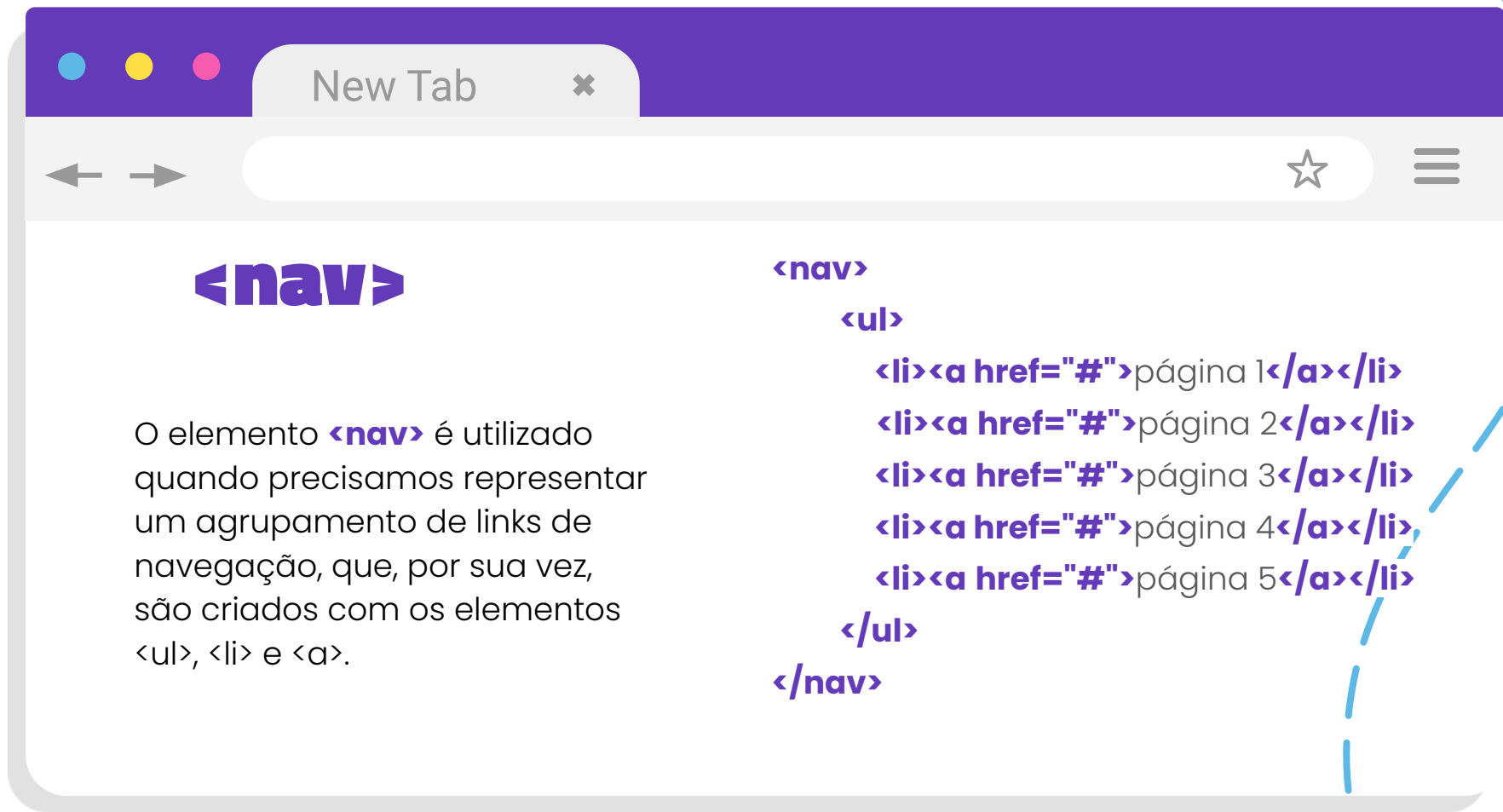
</section>



<article>

Utilizamos o elemento **<article>** quando precisamos declarar um conteúdo que não precisa de outro para fazer sentido em um documento HTML, por exemplo, um artigo em um blog.

```
<article>  
  <h3>Título do artigo 1</h3>  
  <h2>Texto</h2>  
</article>  
<article>  
  <h3>Título do artigo 2</h3>  
  <h2>Texto</h2>  
</article>
```

<nav>

O elemento **<nav>** é utilizado quando precisamos representar um agrupamento de links de navegação, que, por sua vez, são criados com os elementos ``, `` e `<a>`.

```
<nav>
```

```
<ul>
```

```
<li><a href="#">página 1</a></li>
```

```
<li><a href="#">página 2</a></li>
```

```
<li><a href="#">página 3</a></li>
```

```
<li><a href="#">página 4</a></li>
```

```
<li><a href="#">página 5</a></li>
```

```
</ul>
```

```
</nav>
```

<aside>

O elemento **<aside>** é utilizado quando precisamos criar um conteúdo de apoio/adicional ao conteúdo principal. Por exemplo, ao falar de HTML semântico, podemos indicar ao leitor outros conteúdos sobre a linguagem HTML como sugestão de leitura complementar.

```
<aside>
  <nav>
    <ul>
      <li>Link 1</li>
      <li>Link 2</li>
      <li>Link 3</li>
      <li>Link 4</li>
      <li>Link 5</li>
    </ul>
  </nav>
</aside>
```



<main>

O elemento **<main>** especifica o conteúdo principal e, consequentemente, de maior relevância dentro da página. Para ser considerada bem construída, uma página deve apresentar apenas um conteúdo principal.

<main>

<h2>Título**</h2>**

<p>Texto**</p>**

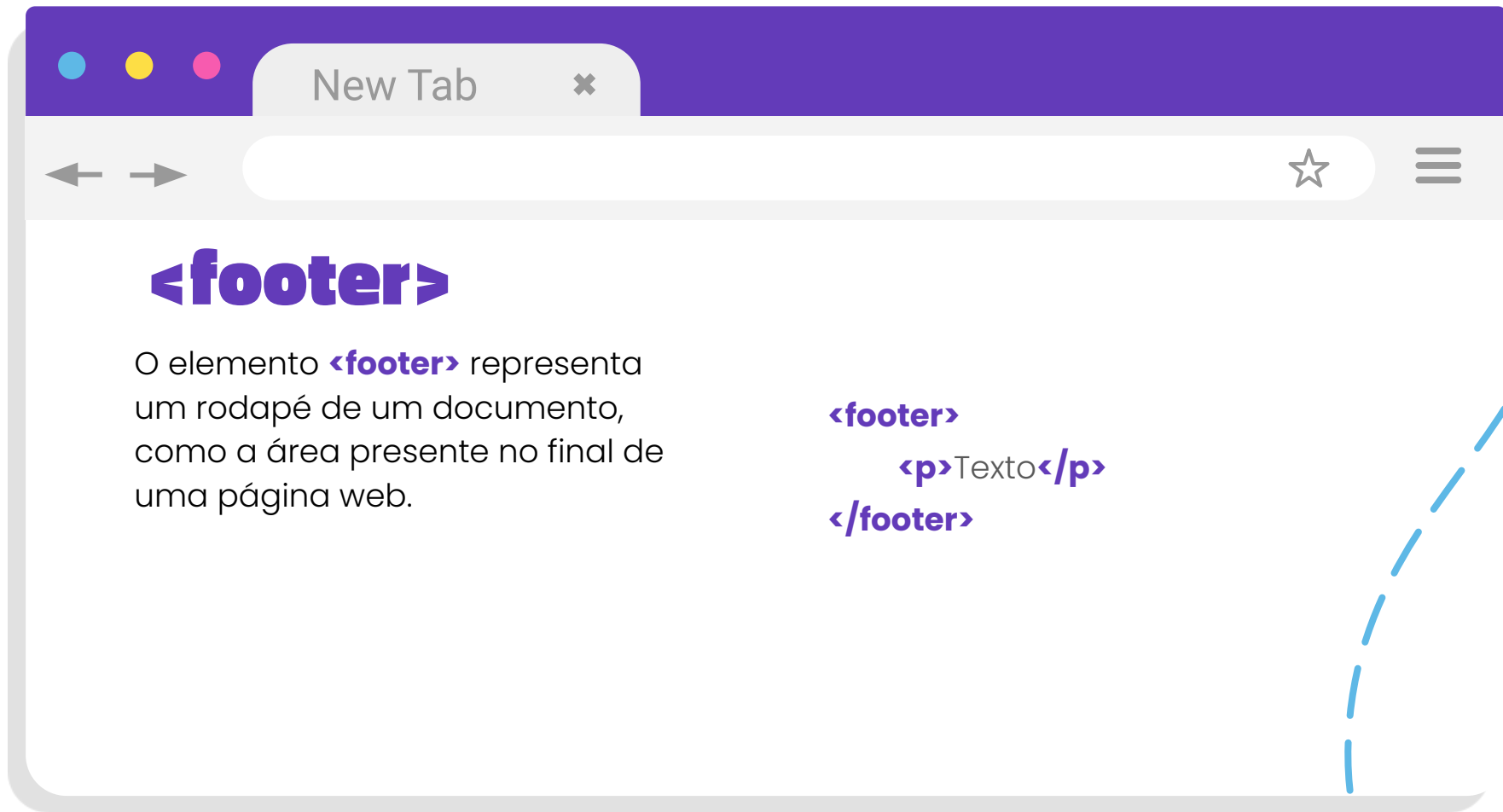
<article>

<h3>Subtítulo**</h3>**

<p>Texto**</p>**

</article>

</main>



<footer>

O elemento **<footer>** representa um rodapé de um documento, como a área presente no final de uma página web.

```
<footer>  
  <p>Texto</p>  
</footer>
```



CSS

Introdução

O que é CSS?

Definição

CSS é chamado de linguagem **Cascading Style Sheet** e é usado para estilizar elementos escritos em uma linguagem de marcação como HTML. O CSS separa o conteúdo da representação visual do site. Pense na decoração da sua página. Utilizando o CSS é possível alterar a cor do texto e do fundo, fonte e espaçamento entre parágrafos. Também pode criar tabelas, usar variações de layouts, ajustar imagens para suas respectivas telas e assim por diante.

Vídeo de ajuda [O que é css?](#)

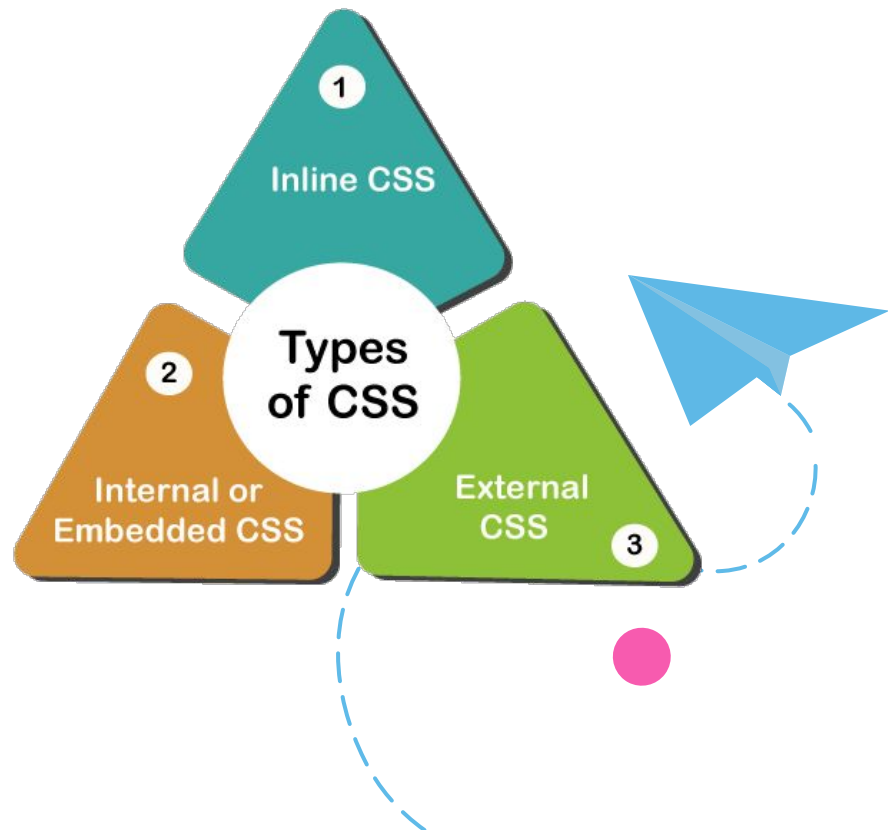


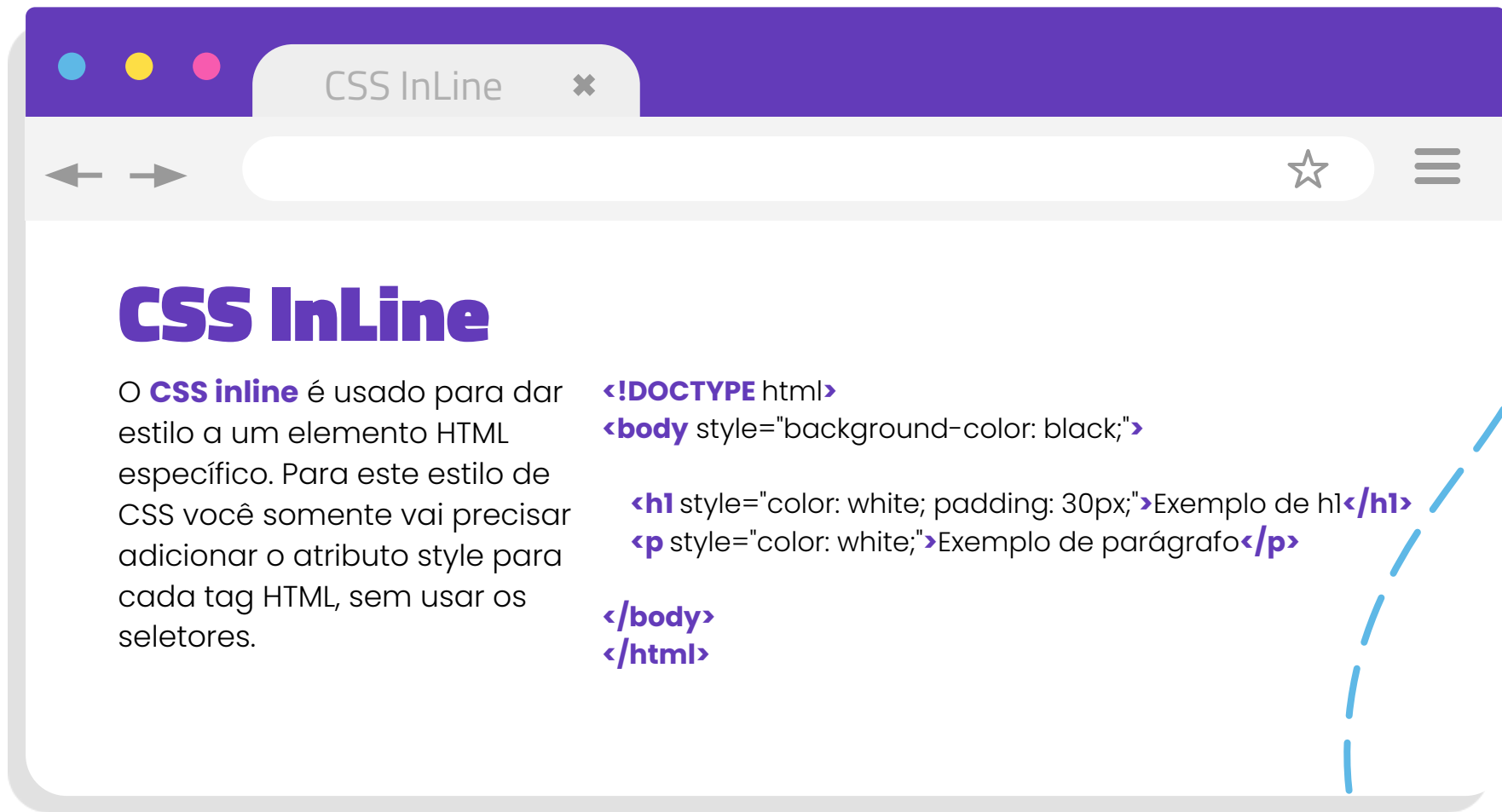
Tipos de CSS

CSS In Line

Temos 3 formas de utilizar o CSS.

- Inline
- Interno
- Externo





CSS InLine

CSS InLine

O **CSS inline** é usado para dar estilo a um elemento HTML específico. Para este estilo de CSS você somente vai precisar adicionar o atributo style para cada tag HTML, sem usar os seletores.

```
<!DOCTYPE html>
```

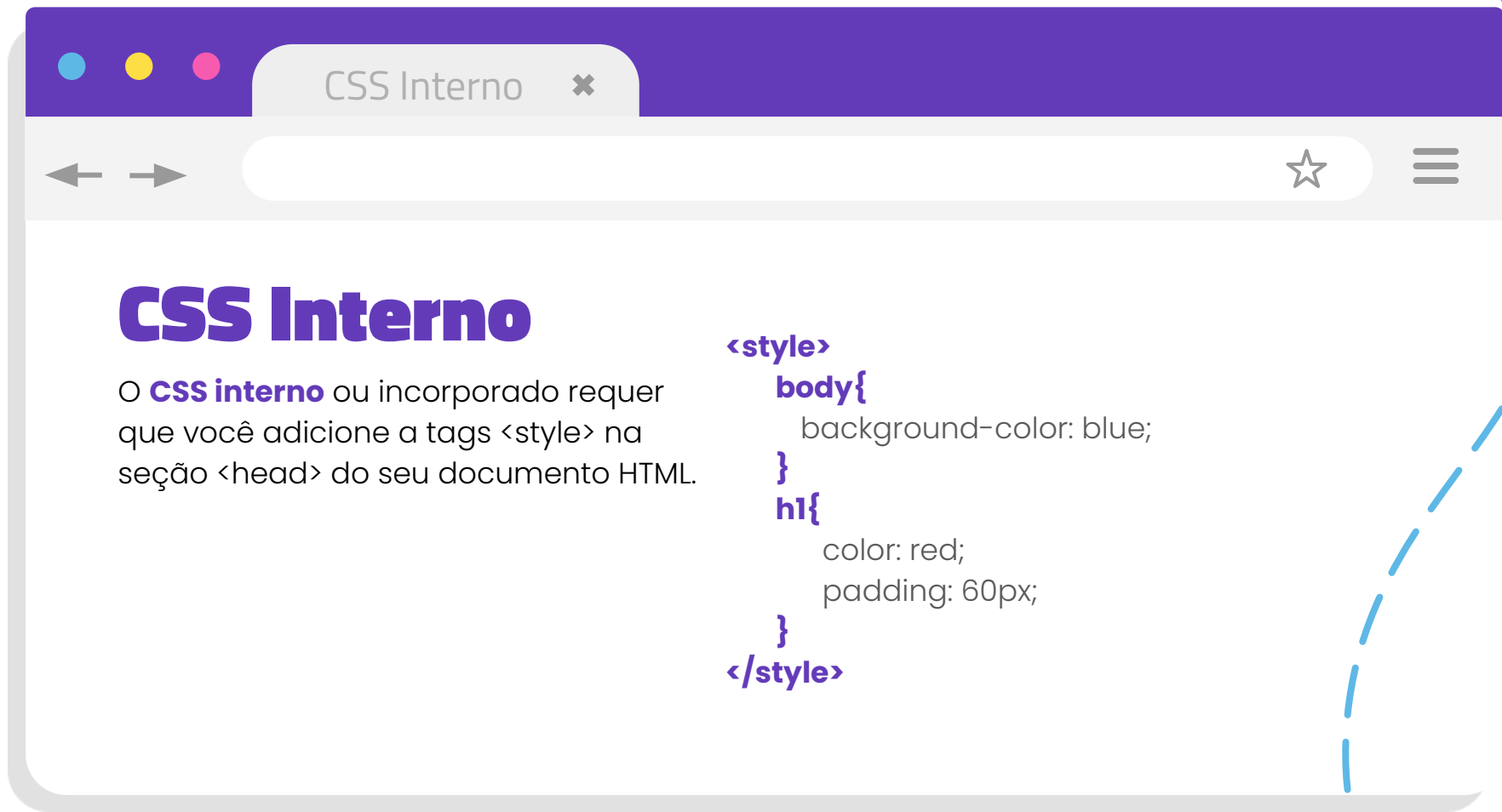
```
<body style="background-color: black;">
```

```
<h1 style="color: white; padding: 30px;">Exemplo de h1</h1>
```

```
<p style="color: white;">Exemplo de parágrafo</p>
```

```
</body>
```

```
</html>
```

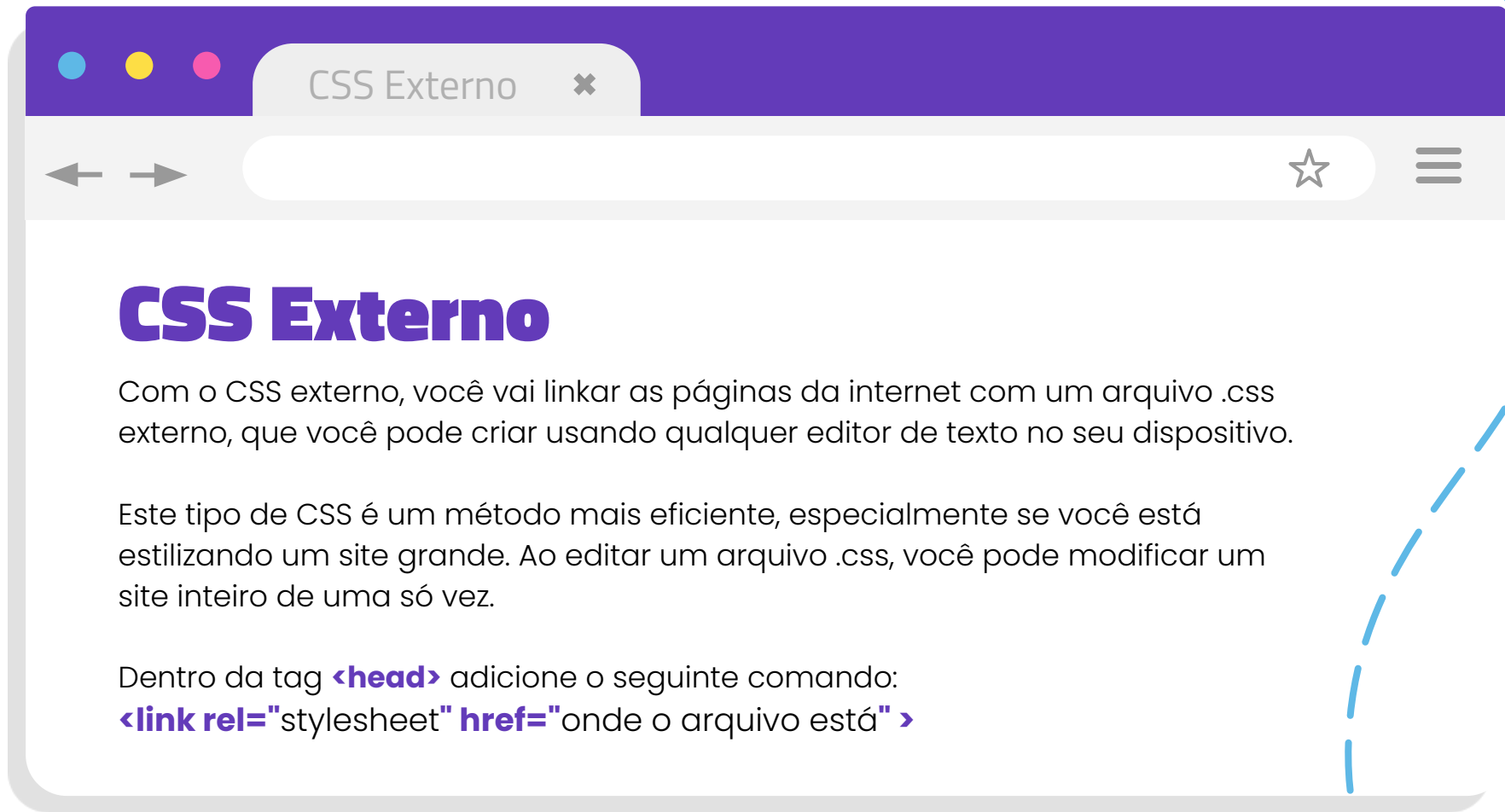



CSS Interno

CSS Interno

O **CSS interno** ou incorporado requer que você adicione a tags <style> na seção <head> do seu documento HTML.

```
<style>
  body{
    background-color: blue;
  }
  h1{
    color: red;
    padding: 60px;
  }
</style>
```



CSS Externo

Com o CSS externo, você vai linkar as páginas da internet com um arquivo .css externo, que você pode criar usando qualquer editor de texto no seu dispositivo.

Este tipo de CSS é um método mais eficiente, especialmente se você está estilizando um site grande. Ao editar um arquivo .css, você pode modificar um site inteiro de uma só vez.

Dentro da tag **<head>** adicione o seguinte comando:
<link rel="stylesheet" href="onde o arquivo está" >



Seletores CSS



Podemos entender seletores CSS como uma regra que vai definir em quais elementos HTML aplicamos o estilo especificado. Seletores podem representar diversas características de um elemento HTML, como uma **tag**, um **id**, uma **classe** ou um **atributo**, e podem ser combinados para especificar ainda mais onde se deve aplicar a formatação.

Tag

```
h1{ color: red; }
```

Id

```
#nome{ color: red; }
```

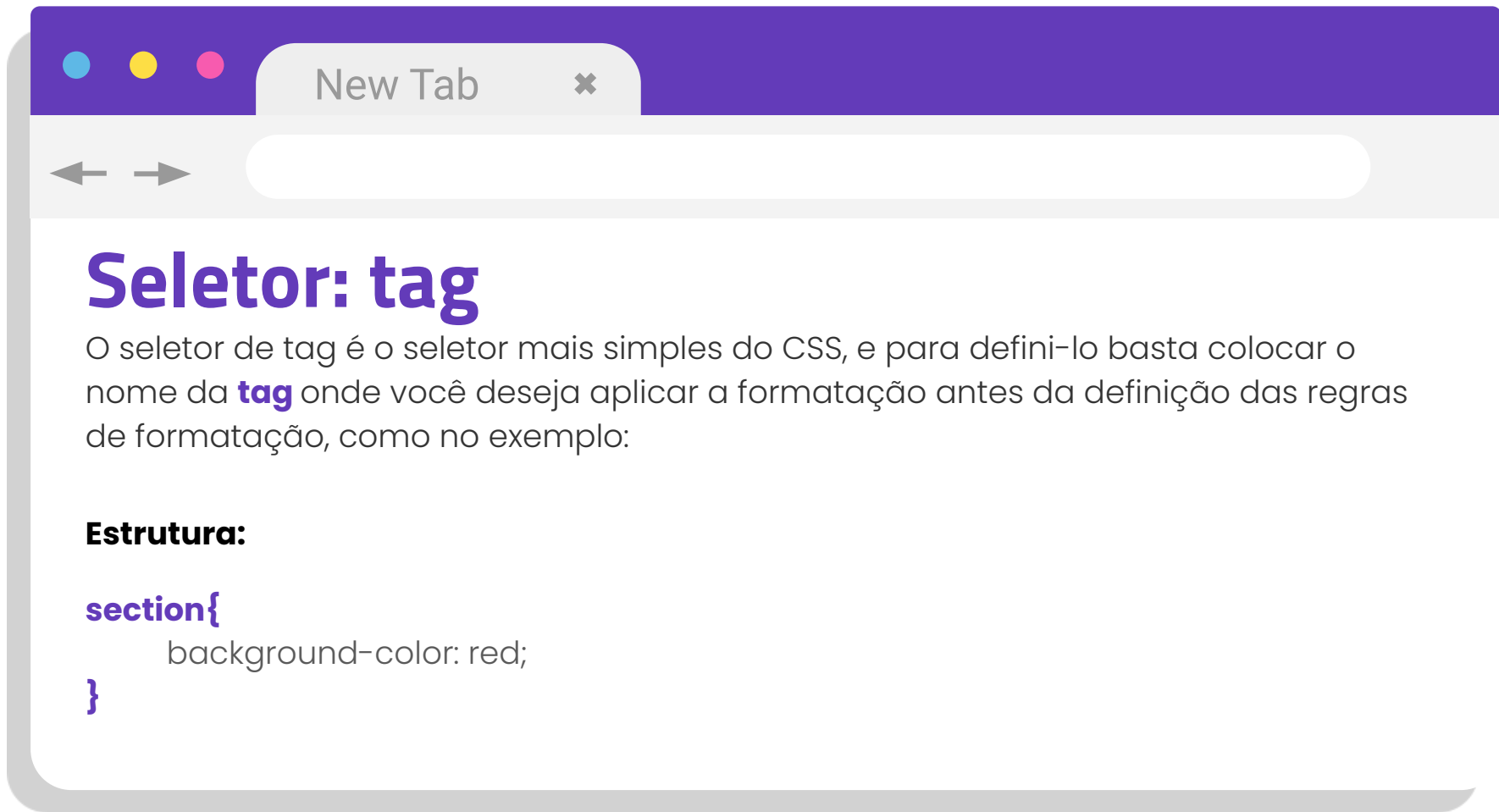
Atributo

```
input[type="text"]{ color: red; }
```

Classe

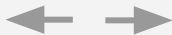
```
.nome{ color: red; }
```







New Tab



Seletor: atributo

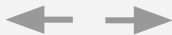
Utilizamos estes tipos de seletores principalmente para realizar a formatação de tags input em formulários, já que escolhemos tipo de uma tag input de acordo com o valor do atributo **type**, como `type = "text"`. Um atributo no CSS é identificado por estar entre colchetes, e um seletor de atributo possui uma sintaxe parecida com **`tag[atributo = "valor"]`**, assim como no exemplo:

Estrutura:

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



New Tab



Seletor: id

Em uma tag HTML, é possível especificar um identificador, chamado de id. Uma tag pode ter apenas um id, e este id não pode se repetir no resto do documento. Dessa forma, a utilização de um seletor por id será muito útil quando você quer aplicar uma formatação a um elemento específico. Representamos os ids no CSS com um #, então um seletor CSS de id possui a sintaxe #nome-do-id, como é possível ver no exemplo:

Estrutura:

```
#conteudo{  
    background-color: red;  
}
```



New Tab

Seletor: classe

Definimos uma classe na tag de abertura de um elemento HTML assim como um id, porém, enquanto um id deve ser único e representar a um único elemento, uma classe pode representar vários elementos, e um elemento pode possuir mais de uma classe.

As classes no CSS são identificadas com um . (ponto), então um seletor CSS de classe possui a sintaxe `.nome-da-classe`, como é possível ver no exemplo:

Estrutura:

```
.conteudo{  
    background-color: red;  
}
```

Id vs Classe

Id

Só poderá ser usado em apenas um elemento, não podendo ser repetido

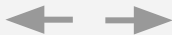
Classe

Pode ser usado em diversos elementos, podendo ser repetido quantas vezes quiser.

Mesmo tendo funções diferentes, o id e class podem ser usados no mesmo elemento ao mesmo tempo. Mas pra que isso? Imagine que você quer aplicar uma função, mas o elemento compartilha a mesma classe com outros? Para conseguir individualizar ele, podemos recorrer ao id.



New Tab



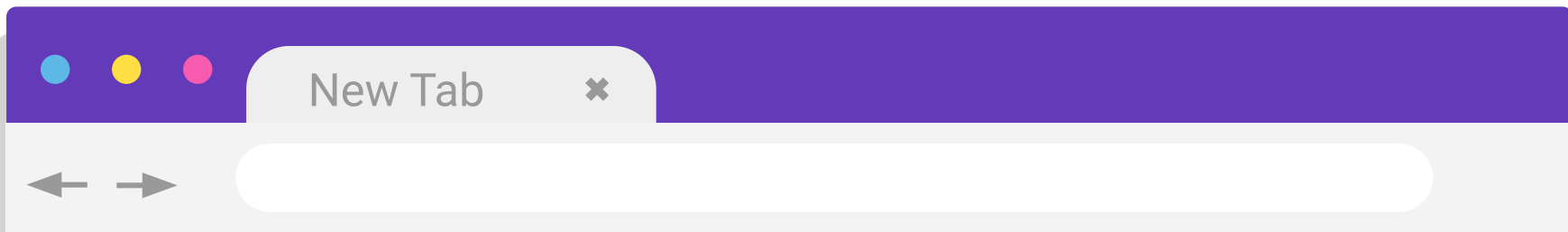
Seletor: universal

O seletor * é conhecido como seletor universal, pois uma formatação associada a este seletor se aplicará a todos os elementos do HTML

Exemplo:

Estrutura:

```
*{  
    background-color: red;  
}
```



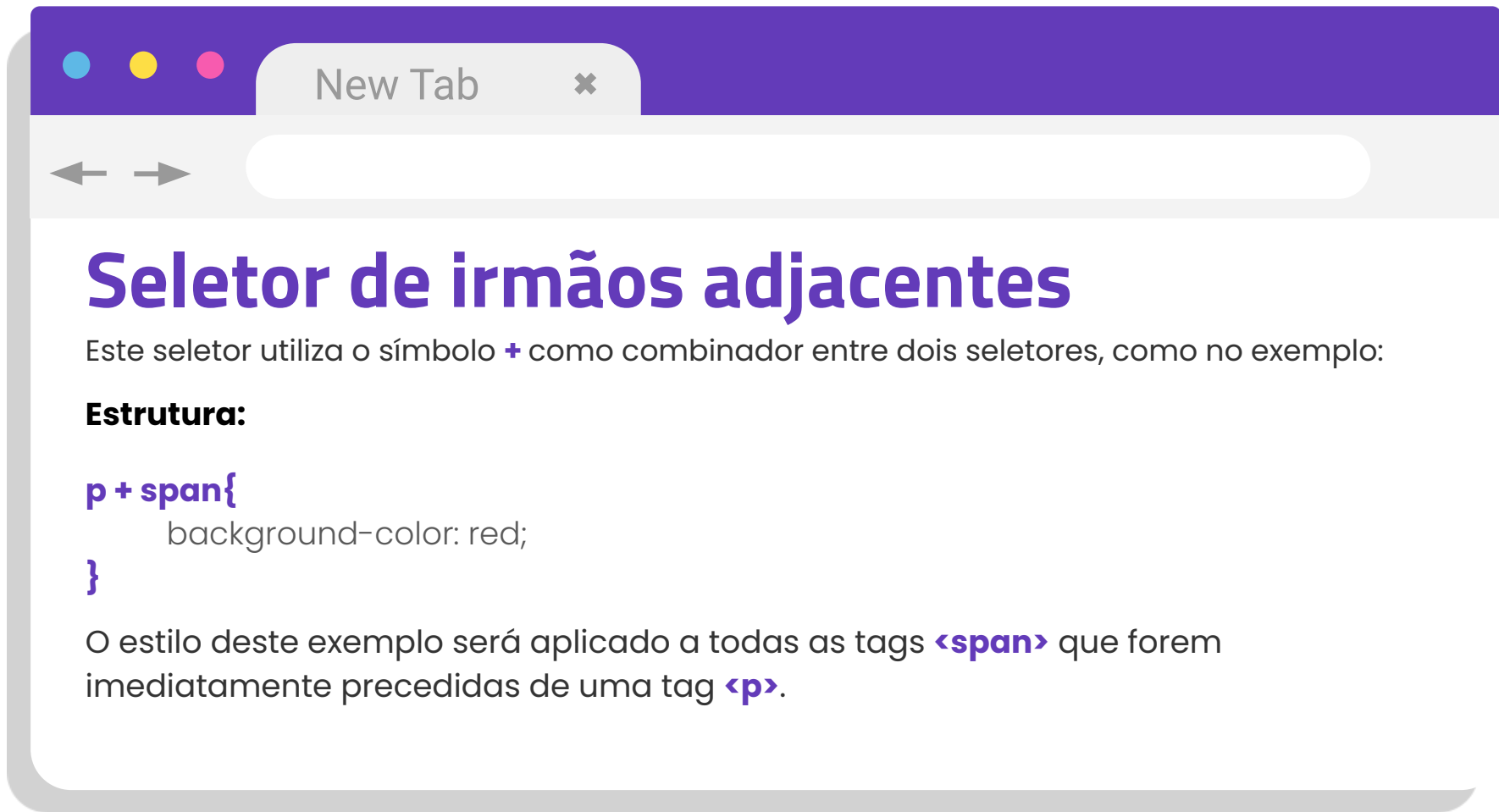
Seletor de descendentes

Este seletor utiliza o espaço como combinador entre dois seletores, assim como no exemplo:

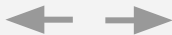
Estrutura:

```
p span{  
    background-color: red;  
}
```

O estilo deste exemplo se aplicará a todas as tags `` que estejam dentro de uma tag `<p>`, mesmo que a tag `` não seja filha direta da tag `<p>`.



New Tab



Seletor de irmãos adjacentes

Este seletor utiliza o símbolo **+** como combinador entre dois seletores, como no exemplo:

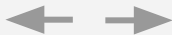
Estrutura:

```
p + span{  
    background-color: red;  
}
```

O estilo deste exemplo será aplicado a todas as tags **** que forem imediatamente precedidas de uma tag **<p>**.



New Tab



Seletor de irmãos gerais

Este seletor utiliza o símbolo `~` como combinador entre dois seletores, assim como no exemplo:

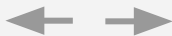
Estrutura:

```
p ~ span{  
    background-color: red;  
}
```

O estilo deste exemplo será aplicado a todas as tags `` que forem precedidas de uma tag `<p>` dentro de um mesmo elemento pai.



New Tab



Seletor de filhos

Este seletor utiliza o símbolo **>** como combinador entre dois seletores, assim como no exemplo:

Estrutura:

```
p > span{  
    background-color: red;  
}
```

O estilo deste exemplo se aplicará apenas às tags **** que forem filhas diretas de uma tag **<p>**.

Cores



Definição

Quando queremos adicionar cores dentro do nosso site, temos algumas opções para escolhas, porém essas são as mais utilizadas:

- **Hexadecimal;**
- **RGB;**
- **RGBA;**
- **nomes das cores.**

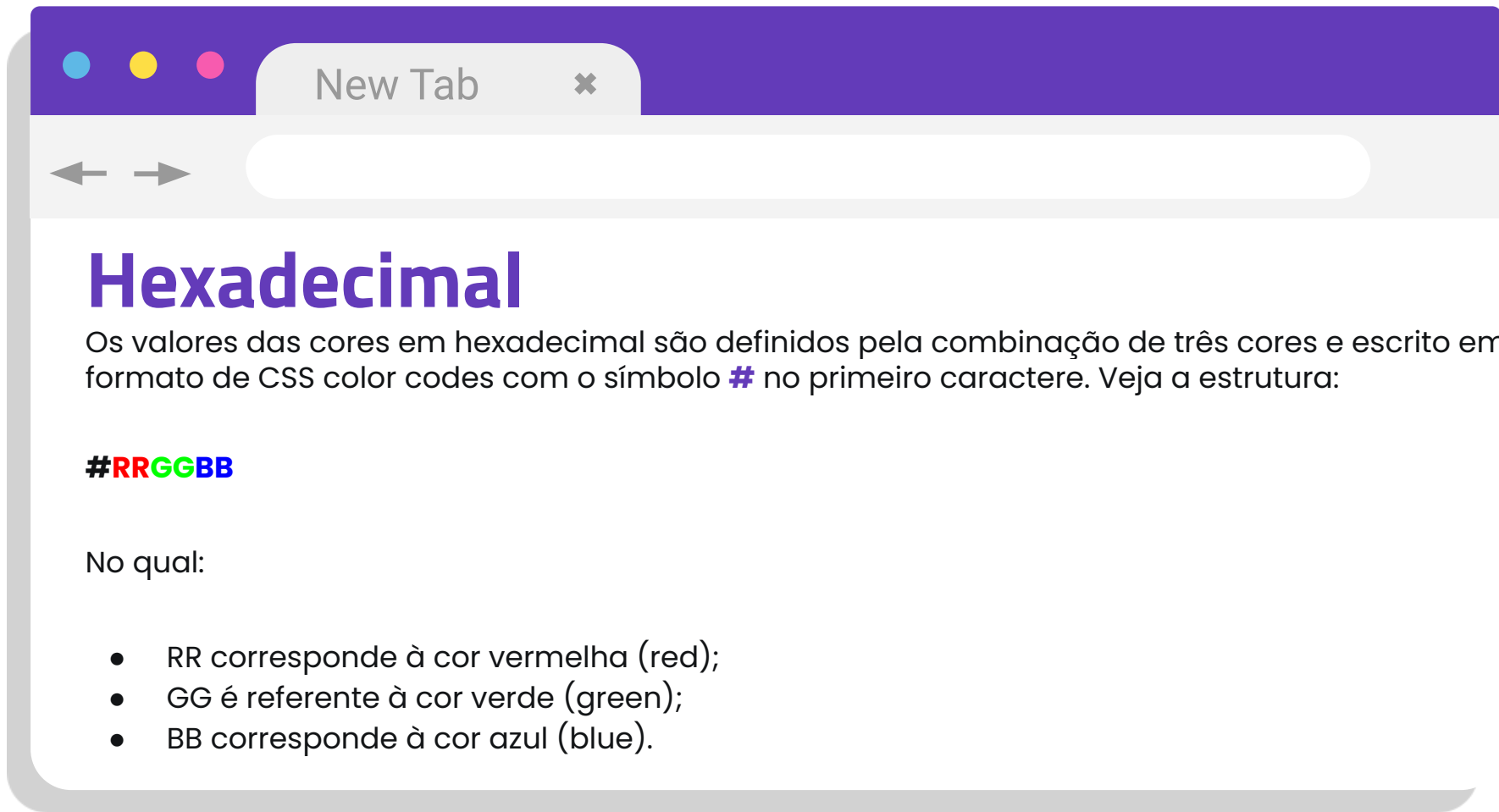
#FFFFFF

rgb(255, 255, 255)

rgba(255, 255, 255, 1)

White





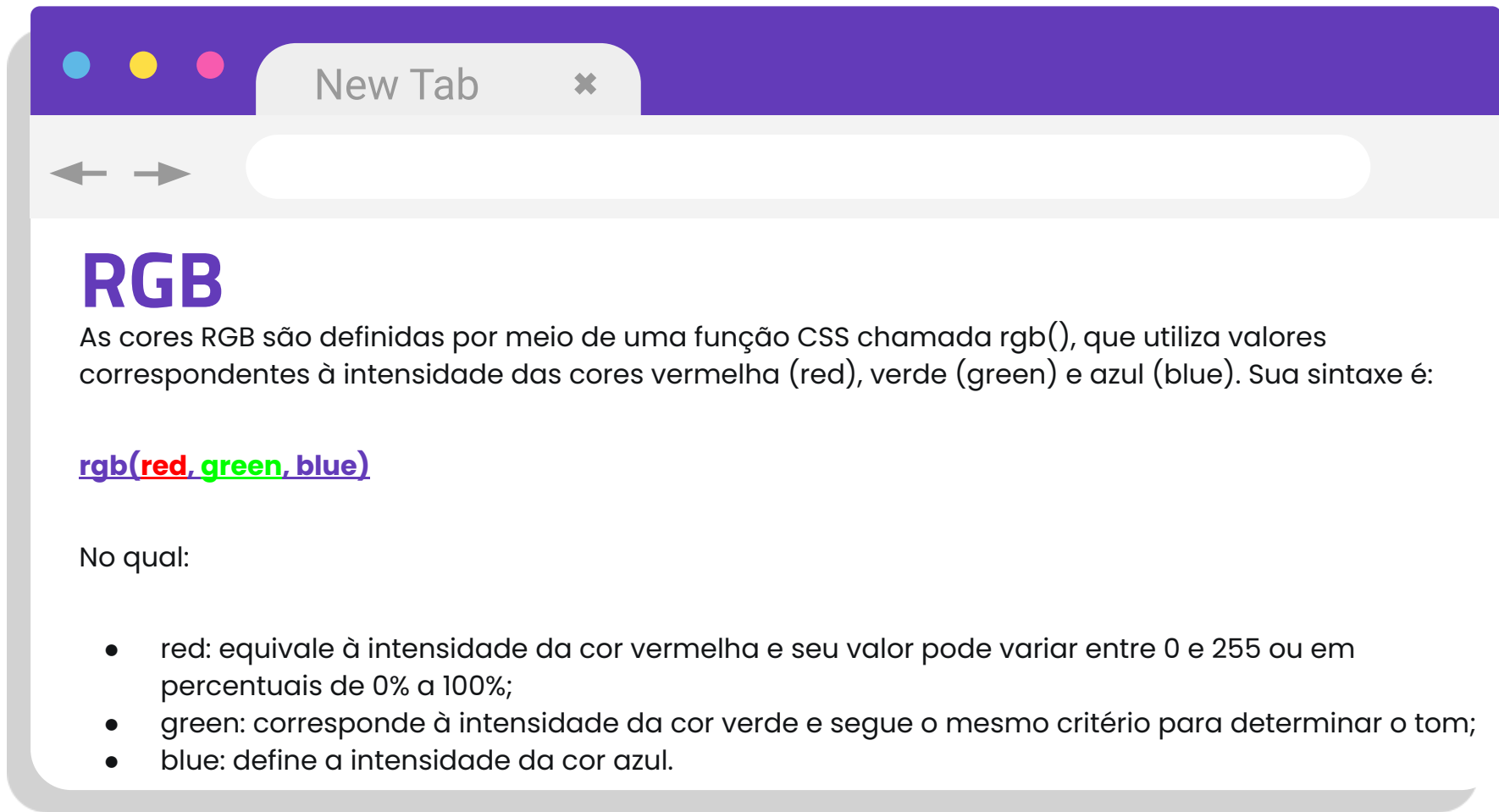
Hexadecimal

Os valores das cores em hexadecimal são definidos pela combinação de três cores e escrito em formato de CSS color codes com o símbolo # no primeiro caractere. Veja a estrutura:

#RRGGBB

No qual:

- RR corresponde à cor vermelha (red);
- GG é referente à cor verde (green);
- BB corresponde à cor azul (blue).



New Tab



RGB

As cores RGB são definidas por meio de uma função CSS chamada `rgb()`, que utiliza valores correspondentes à intensidade das cores vermelha (red), verde (green) e azul (blue). Sua sintaxe é:

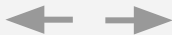
`rgb(red, green, blue)`

No qual:

- red: equivale à intensidade da cor vermelha e seu valor pode variar entre 0 e 255 ou em percentuais de 0% a 100%;
- green: corresponde à intensidade da cor verde e segue o mesmo critério para determinar o tom;
- blue: define a intensidade da cor azul.



New Tab

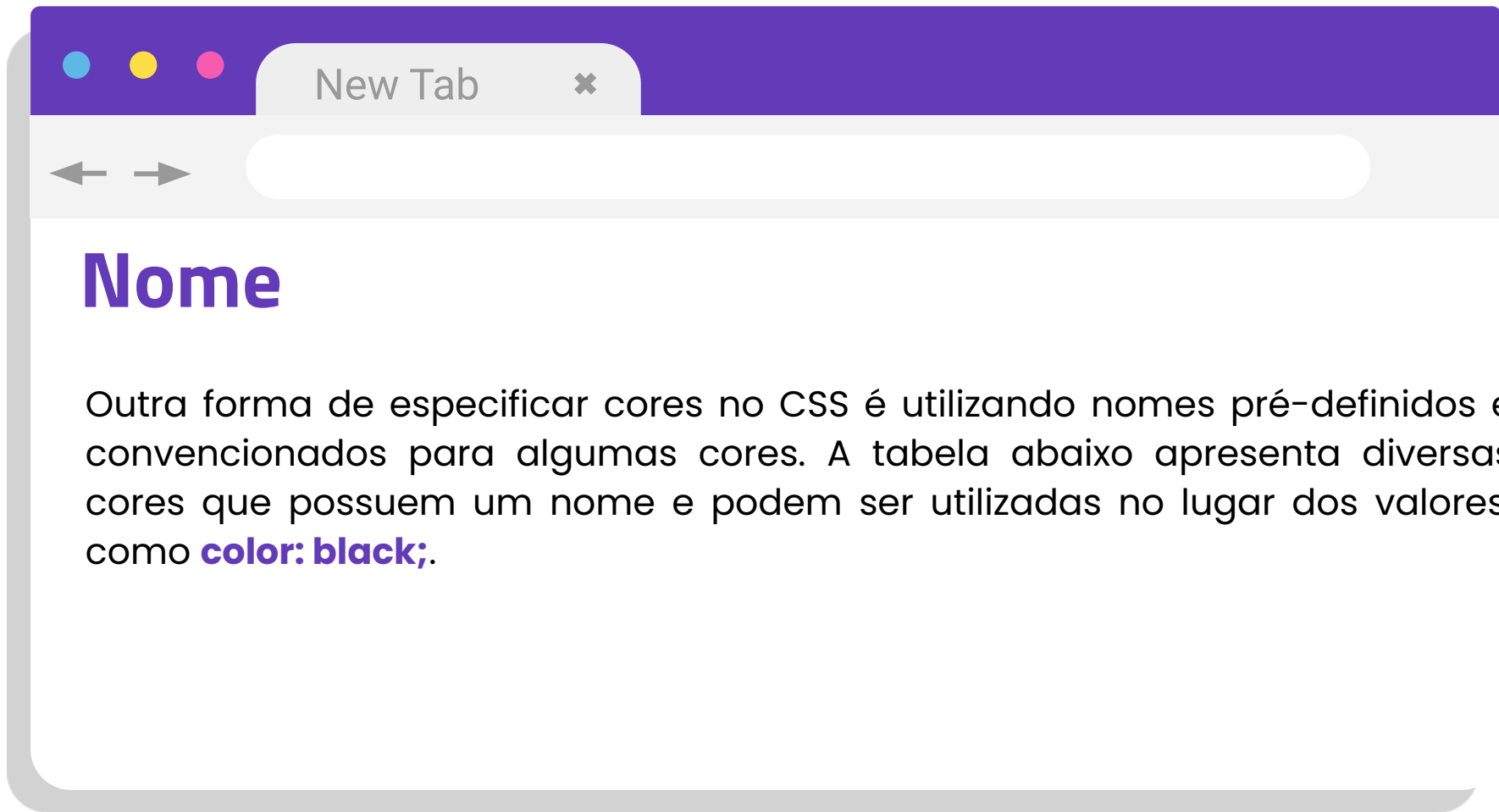


RGBA

As cores RGB são definidas por meio de uma função CSS chamada `rgb()`, que utiliza valores correspondentes à intensidade das cores vermelha (red), verde (green) e azul (blue). Sua sintaxe é:

`rgba(vermelho, verde, azul, alfa)`

- red: equivale à intensidade da cor vermelha e seu valor pode variar entre 0 e 255 ou em percentuais de 0% a 100%;
- green: corresponde à intensidade da cor verde e segue o mesmo critério para determinar o tom;
- blue: define a intensidade da cor azul.
- Alfa: corresponde a opacidade da cor, sendo 0 transparente e 1 opaco



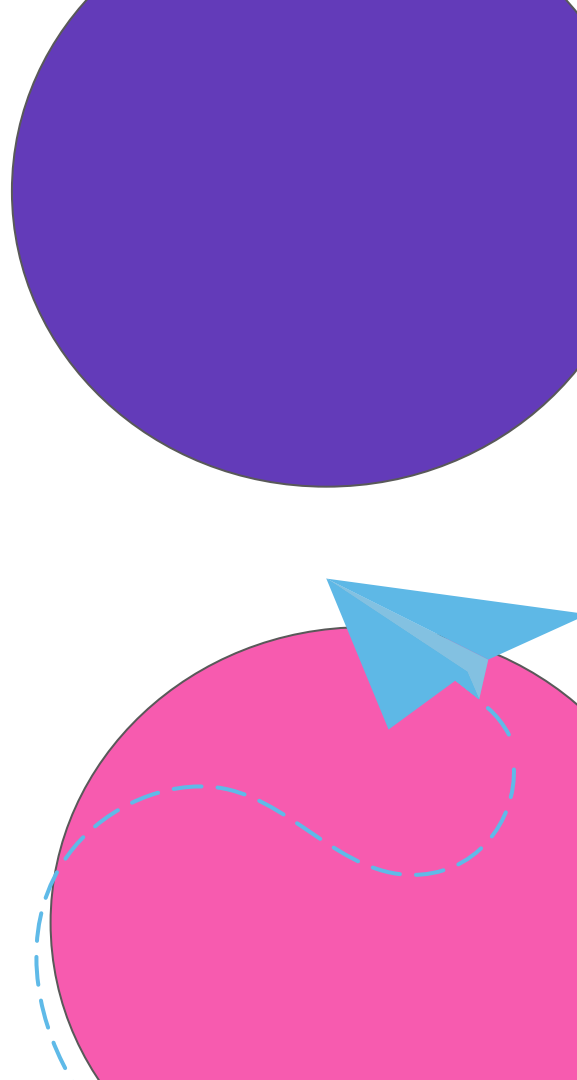

Border



Definição

Ao inserir um elemento em um documento HTML, há várias possibilidades de estilizar sua borda. Você pode utilizar estilos que a propriedade já possui, além de poder também alterar sua cor, espessura e até mesmo seu formato! Utilizando apenas o poder do CSS! E caso queira aplicar na borda de algum elemento uma imagem bem bonita que você encontrou no Google, também é possível!

Vídeo de ajuda [Border](#)



Border

Utilizando apenas a propriedade border, é possível aplicar três valores:

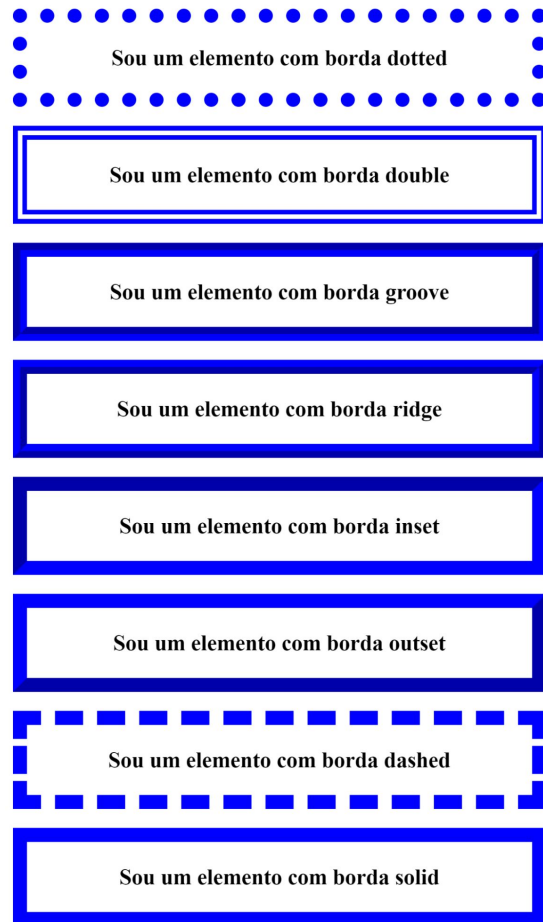
border: 30px solid blue;

O código acima é uma forma abreviada de escrever os mesmos valores abaixo:

border-width: 30px; – Estiliza a largura

border-style: solid; – Estiliza o estilo

border-color: blue; – Estiliza a cor



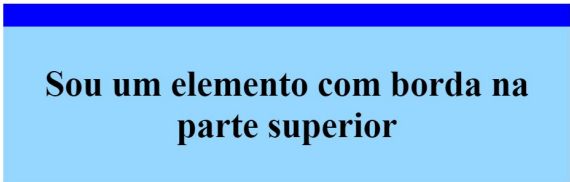
Border

É possível que um elemento tenha borda em apenas um de seus lados

Para realizar isso, então, devemos utilizar apenas a propriedade **border-top**. Veja abaixo o HTML da borda da imagem acima.

O código acima é uma forma abreviada de escrever os mesmos valores abaixo:

border-top: 30px solid blue;


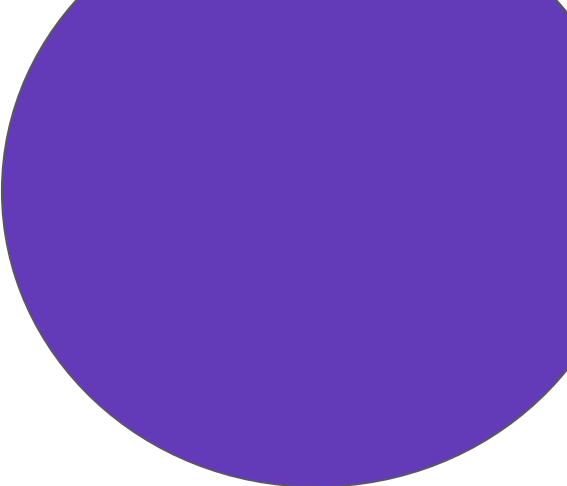
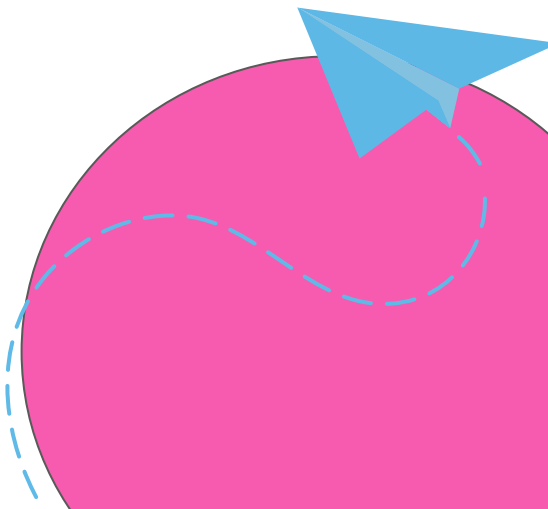


Sou um elemento com borda na
parte superior

Border



Para cada lado do elemento HTML, temos uma propriedade específica. Confira a seguir quais são:

- **border-top**: Aplica borda na parte superior de um elemento.
 - **border-left**: Aplica borda no lado esquerdo de um elemento.
 - **border-right**: Aplica borda no lado direito de um elemento.
 - **border-bottom**: Aplica borda na parte inferior de um elemento.
- 
- 
- 

Border

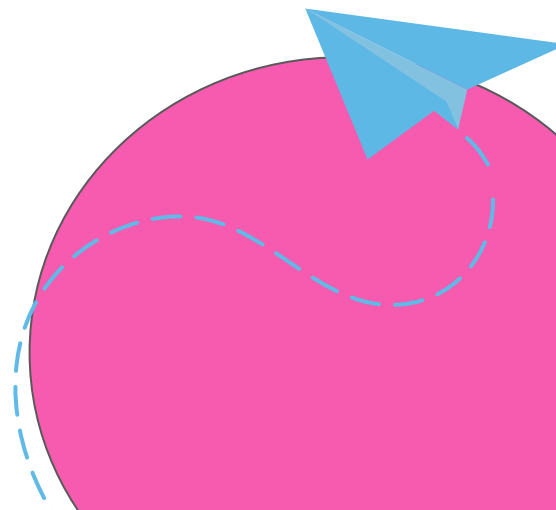
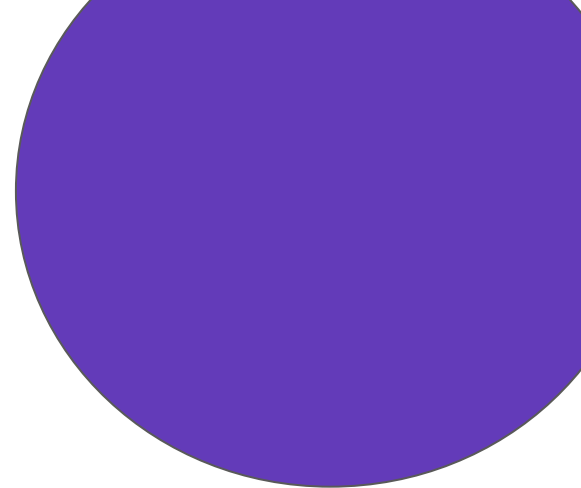
A **border-radius** propriedade define o raio dos cantos do elemento.

Dica: Esta propriedade permite adicionar cantos arredondados aos elementos!

border: 30px solid blue;

border-radius: 20px;

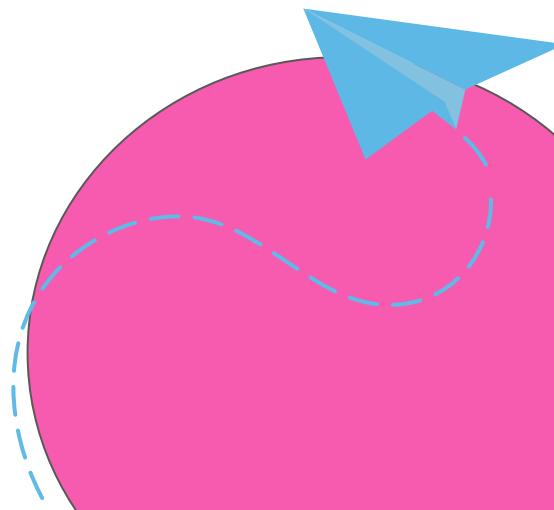
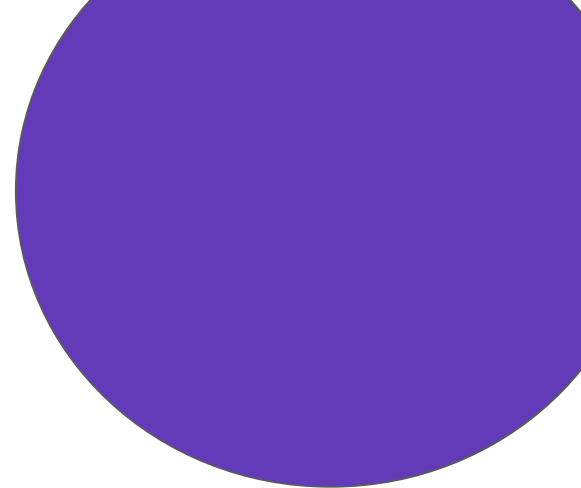
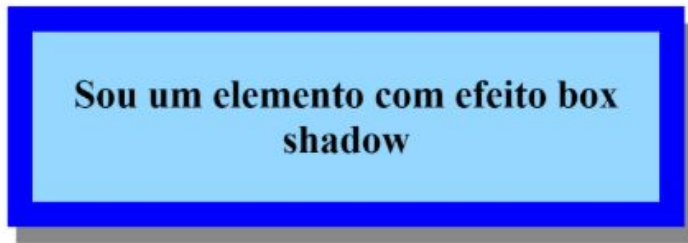
Sou um elemento com border-radius

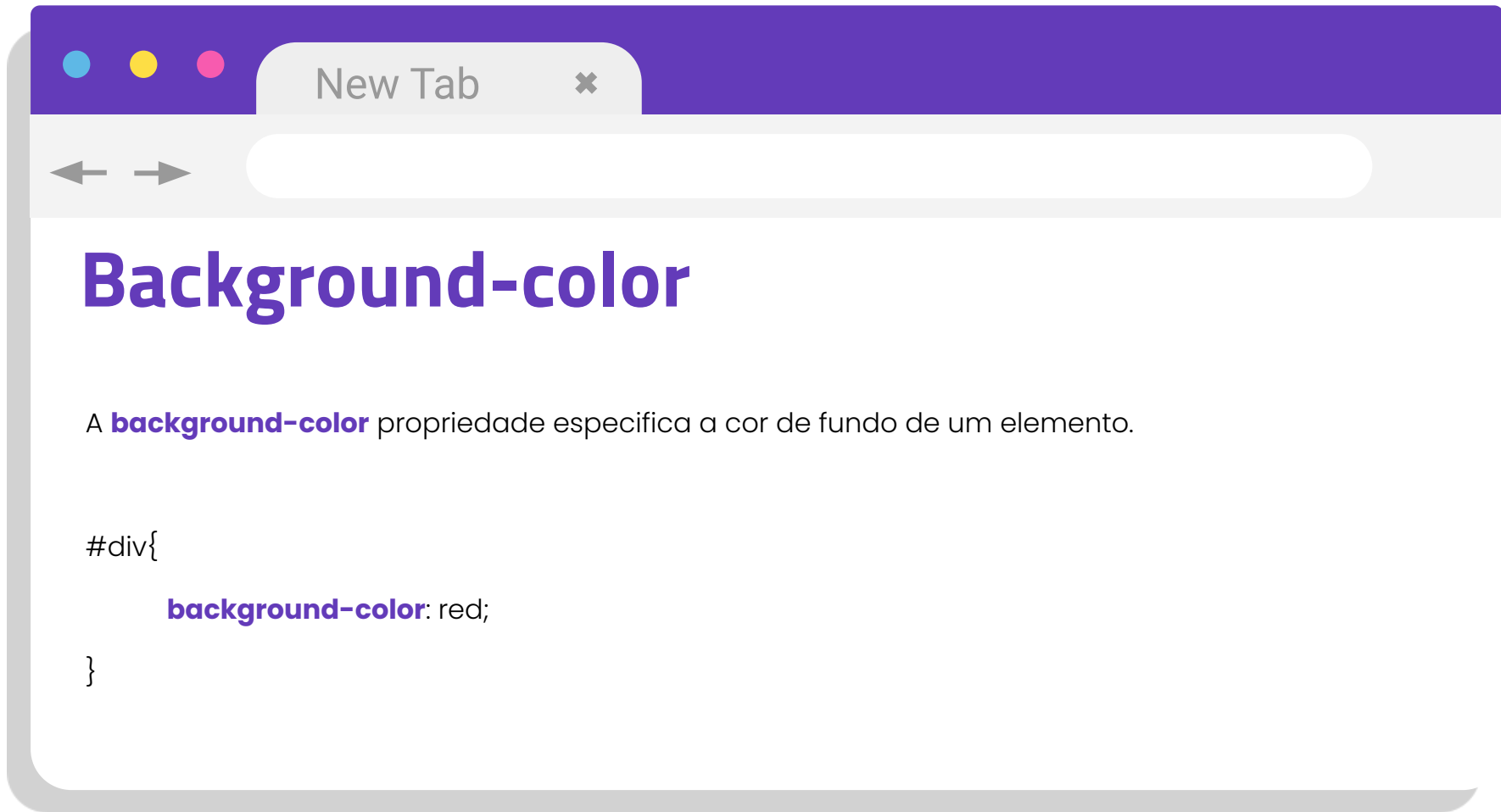


Box Shadow

Esta propriedade permite aplicar efeito de sombra em um elemento.

box-shadow: 10px 20px grey

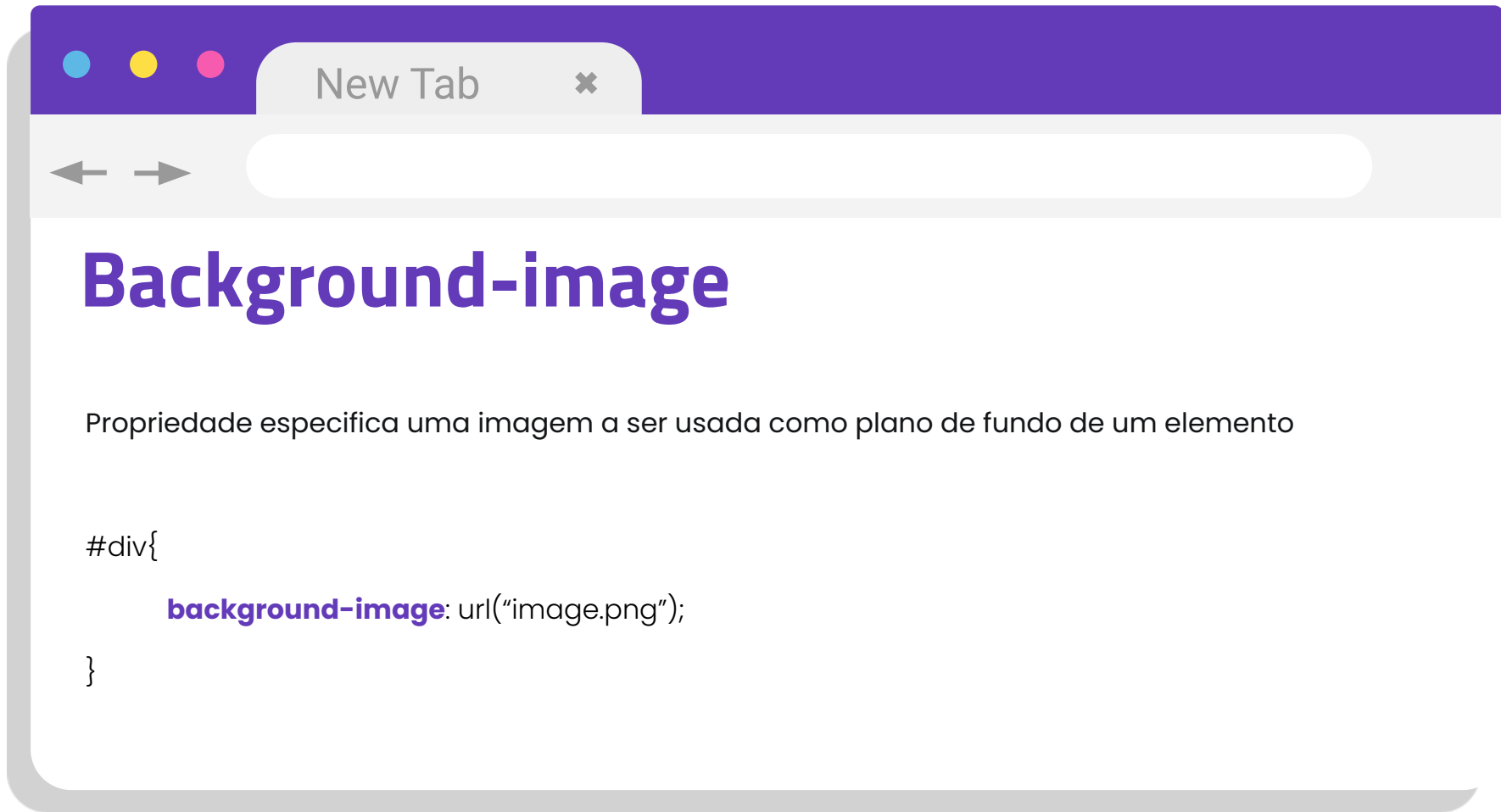




Background-color

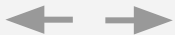
A **background-color** propriedade especifica a cor de fundo de um elemento.

```
#div{  
    background-color: red;  
}
```





New Tab



Background-repeat

A propriedade define como uma imagem de fundo será repetida.

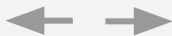
```
#div{
```

```
    background-repeat: propriedade;
```

```
}
```



New Tab



Background-repeat

Propriedades:

repeat - A imagem de fundo é repetida verticalmente e horizontalmente. A última imagem será cortada se não couber. Isso é padrão.

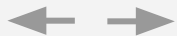
repeat-x - A imagem de fundo é repetida apenas horizontalmente

repeat-y - A imagem de fundo é repetida apenas verticalmente

no-repeat - A imagem de fundo não é repetida. A imagem será mostrada apenas uma vez



New Tab



Background-repeat

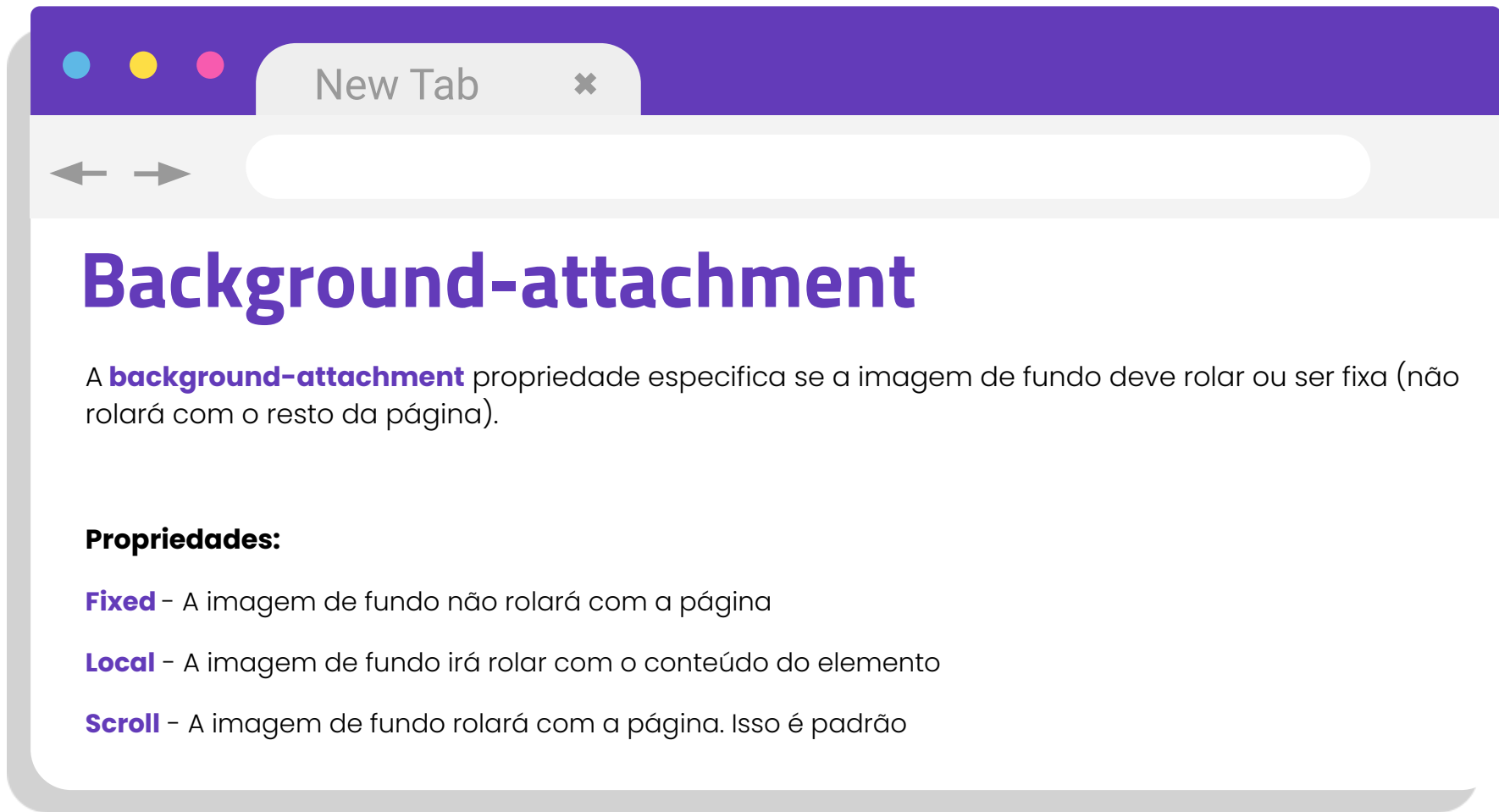
Propriedades:

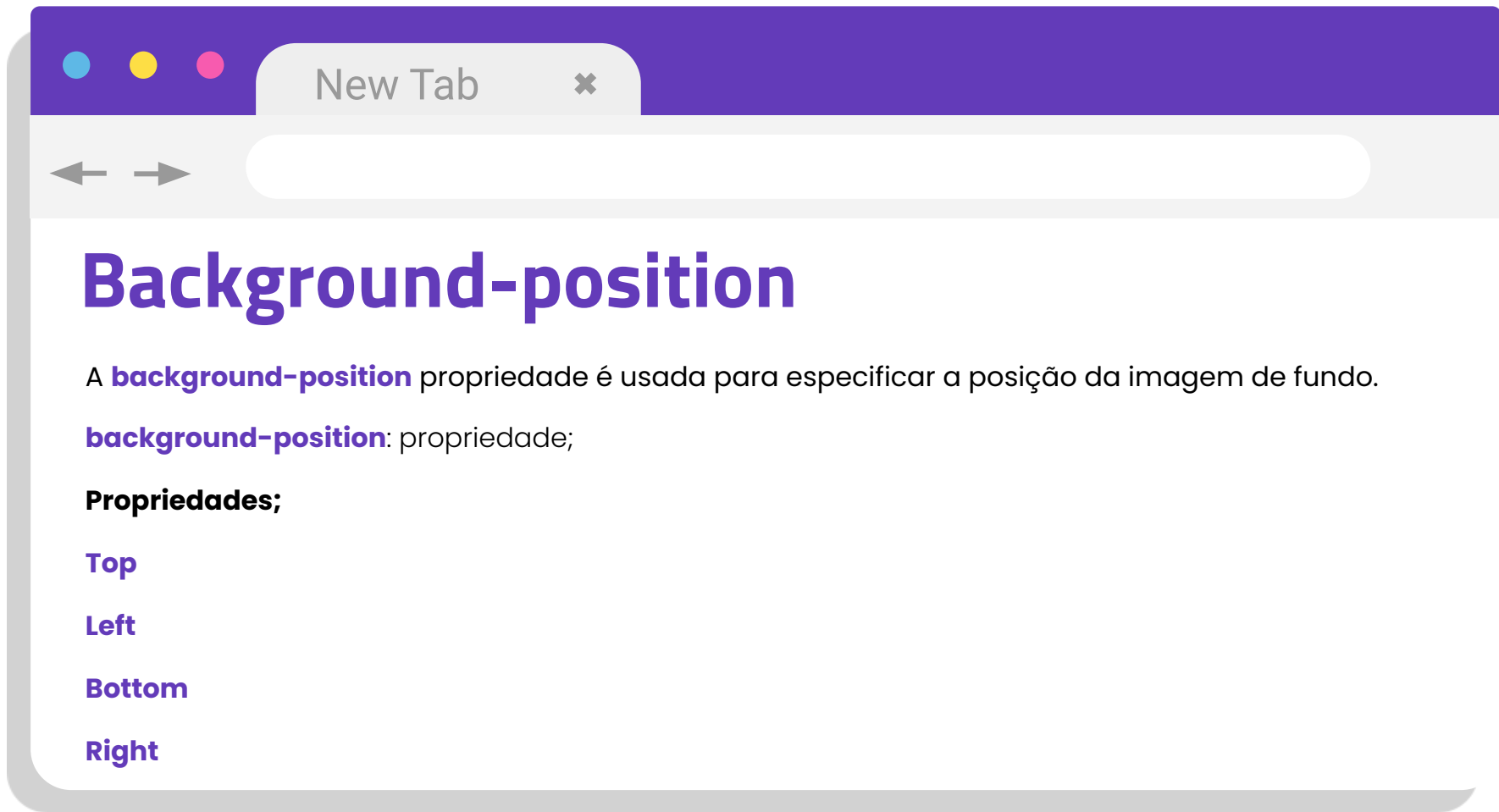
space - A imagem de fundo é repetida o máximo possível sem recorte. A primeira e a última imagem são fixadas em cada lado do elemento e o espaço em branco é distribuído uniformemente entre as imagens.

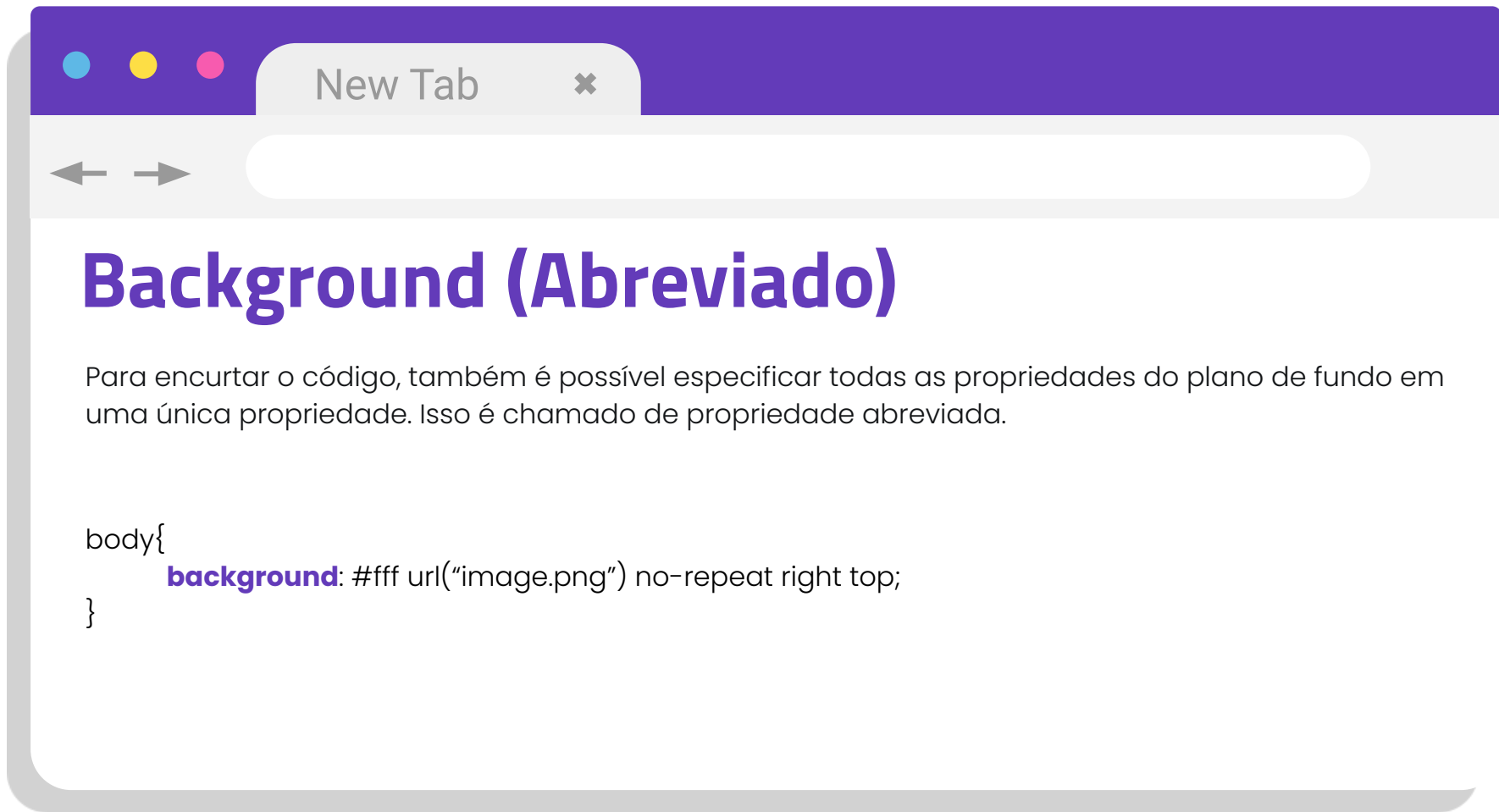
round - A imagem de fundo é repetida e comprimida ou esticada para preencher o espaço (sem lacunas)

initial - Configura esta propriedade para seu valor padrão.

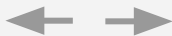
inherit - Herda esta propriedade de seu elemento pai.







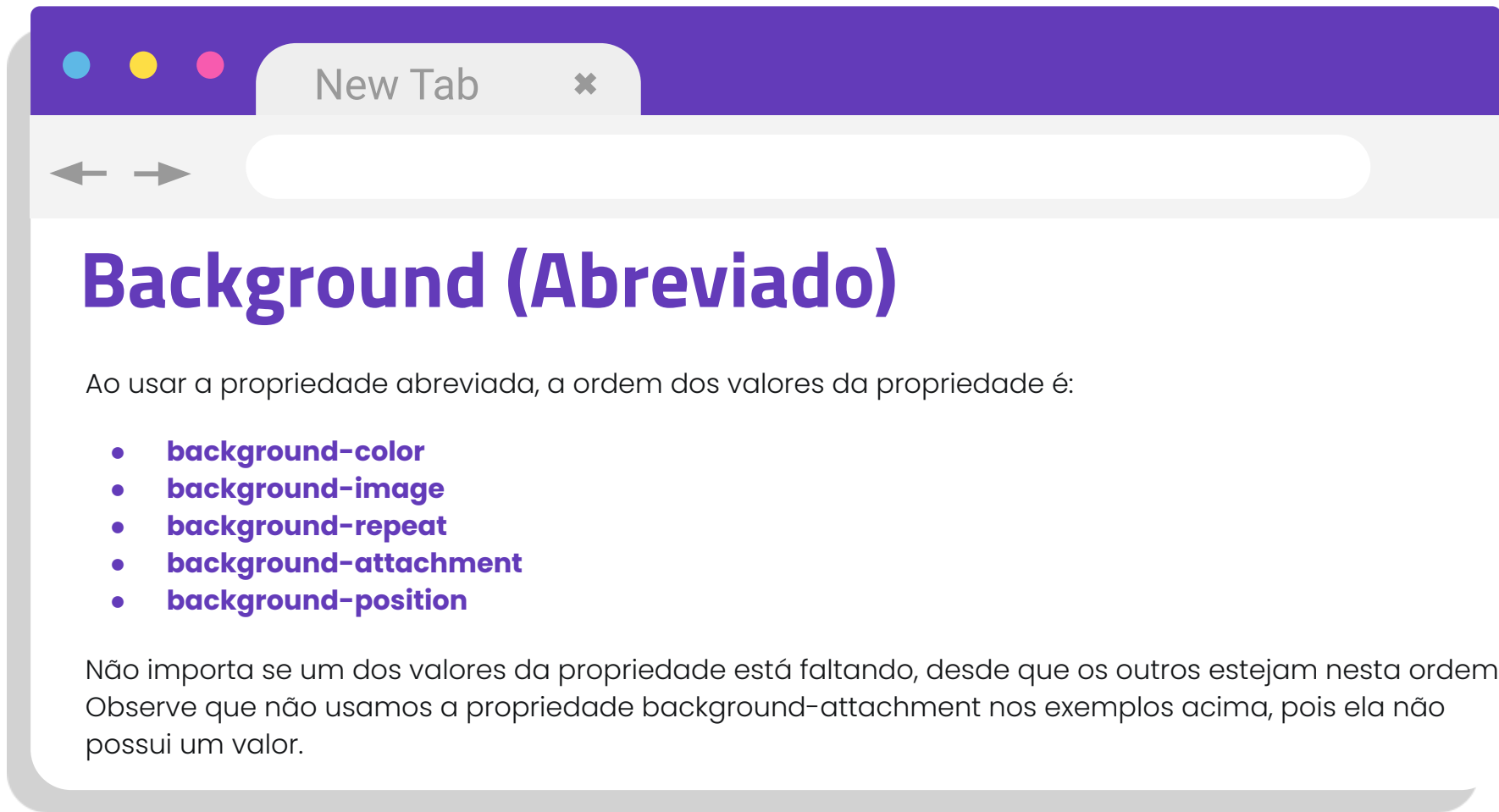
New Tab



Background (Abreviado)

Para encurtar o código, também é possível especificar todas as propriedades do plano de fundo em uma única propriedade. Isso é chamado de propriedade abreviada.

```
body{  
  background: #fff url("image.png") no-repeat right top;  
}
```

Background (Abreviado)

Ao usar a propriedade abreviada, a ordem dos valores da propriedade é:

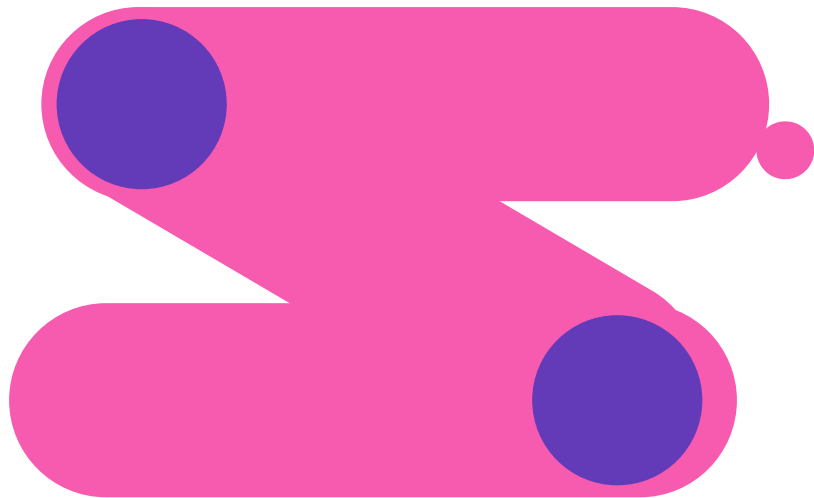
- **background-color**
- **background-image**
- **background-repeat**
- **background-attachment**
- **background-position**

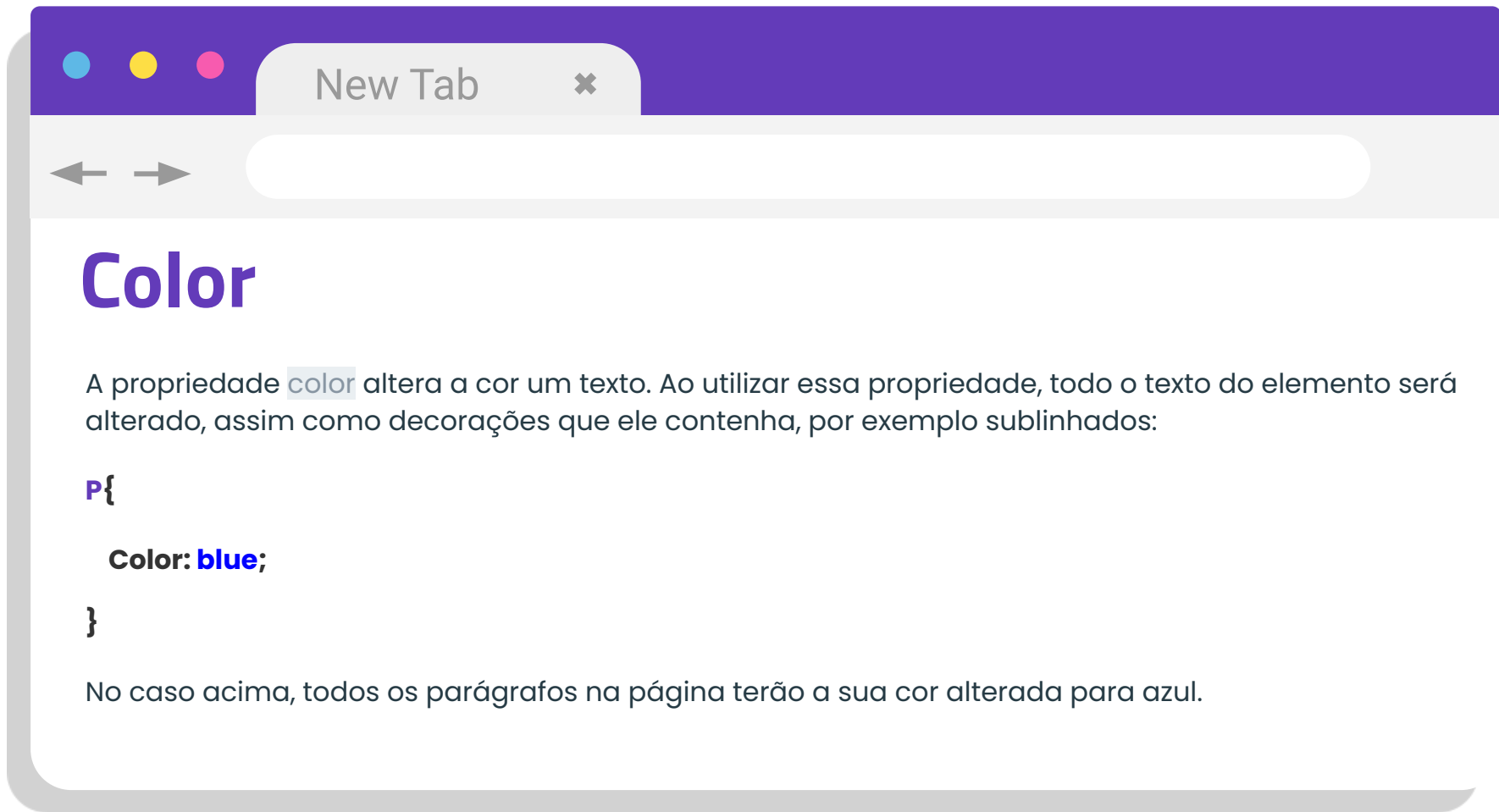
Não importa se um dos valores da propriedade está faltando, desde que os outros estejam nesta ordem. Observe que não usamos a propriedade background-attachment nos exemplos acima, pois ela não possui um valor.

Font

Apesar de vivermos na era da comunicação em vídeo, o texto ainda é a forma mais básica e direta de comunicação em sites. Nesta seção da documentação de CSS veremos como utilizar negritos, itálicos e sublinhados, alterar a fonte e aplicar efeitos sobre elementos de texto. Você aprenderá também a definir o espaçamento entre as linhas e letras.

● Vídeo de ajuda [Font](#)





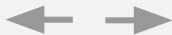
Color

A propriedade `color` altera a cor um texto. Ao utilizar essa propriedade, todo o texto do elemento será alterado, assim como decorações que ele contenha, por exemplo sublinhados:

```
P{  
  Color: blue;  
}
```

No caso acima, todos os parágrafos na página terão a sua cor alterada para azul.

New Tab



font-family

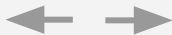
Aplicamos uma fonte ao texto de um elemento com a propriedade **font-family**. Essa propriedade permite informar uma lista composta por uma ou mais fontes que, quando forem encontradas pelo navegador no computador do usuário do site, serão utilizadas.

No exemplo, primeiro o navegador tentará aplicar Trebuchet MS, recorrendo a Verdana e sans-serif em caso de falha, nesta ordem.

```
P{  
  font-family: "Trebuchet MS", Verdana,  
  sans-serif;  
}
```



New Tab

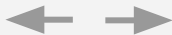


font-size

A propriedade `font-size` altera o tamanho do texto. Podemos utilizar diferentes unidades para determinar esse tamanho, sendo `px` e `em` as mais comuns. No exemplo abaixo definimos em pixels qual será o tamanho da fonte:

```
P{  
  font-size: 16px;  
}
```

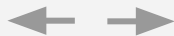
New Tab



Font-style e Font-weight

Com o CSS podemos ainda alterar o peso de um texto. As mais simples são itálico e negrito. Abaixo temos um exemplo onde aplicamos itálico ao parágrafo com a propriedade **font-style** e negrito ao span com a propriedade **font-weight**:

```
p{  
    font-style: italic;  
    font-weight: bold;  
}
```



text-transform

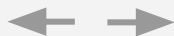
Com o CSS também podemos aplicar transformações ao texto utilizando para isso a propriedade **text-transform**. No trecho de código ao lado deixamos a primeira letra de cada palavra em maiúscula atribuindo a essa propriedade o valor **capitalize**:

valores utilizados com frequência para essa propriedade são listados a seguir:

- **none**: remove todas as transformações;
- **uppercase**: deixa todas as letras em maiúscula;
- **lowercase**: deixa todas as letras em minúscula.

```
P{  
  text-transform: capitalize;  
}
```

Vídeo de ajuda [Text](#)



text-decoration

Outra propriedade muito utilizada para decorações é a [text-decoration](#). Com ela podemos, por exemplo, remover o sublinhado de links, como pode ser visto abaixo:

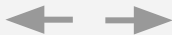
```
P{  
    text-decoration: none;  
}
```

Além do valor none, que remove qualquer transformação aplicada anteriormente, também podemos utilizar os seguintes:

- [underline](#): aplica uma linha abaixo do texto;
- [overline](#): aplica uma linha sobre o texto;
- [line-through](#): risca o texto.



New Tab



text-shadow

Quando utilizadas corretamente, as sombras facilitam a leitura e destacam o texto de imagens ou qualquer outro elemento que seja utilizado como fundo.

A propriedade utilizada para esse fim é a `text-shadow`, que recebe quatro valores, conforme descrito abaixo:

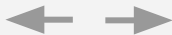
```
P{
```

```
Text-shadow: 2px 4px 5px #eee;
```

```
}
```



New Tab



text-align

A propriedade `text-align` permite controlar como o texto será alinhado dentro do elemento que o contém. No exemplo ao lado, o parágrafo será alinhado no centro do elemento no qual estiver contido.

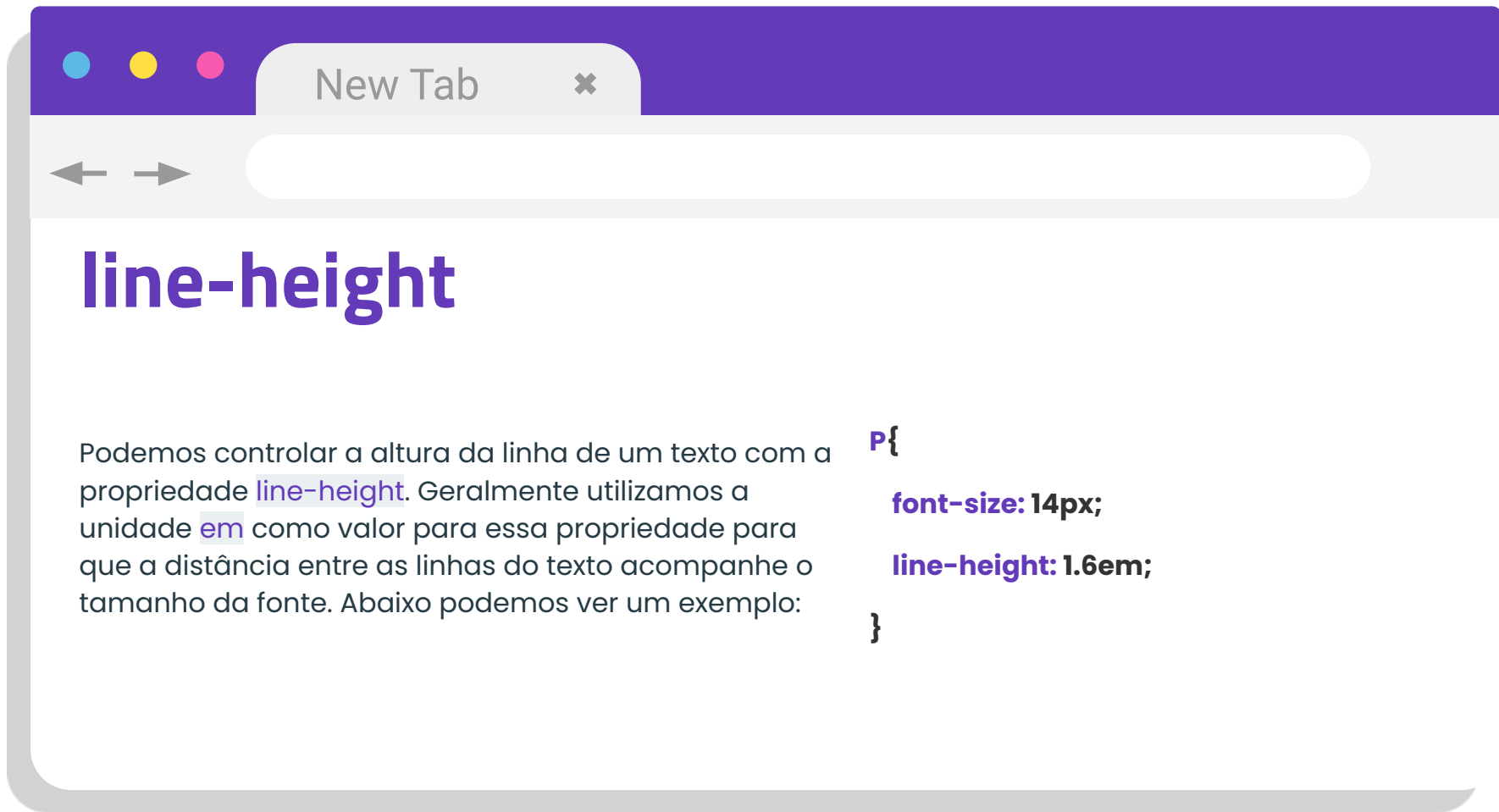
Outros valores que podem ser utilizados para essa propriedade são:

- `left`: justifica o texto a esquerda;
- `right`: justifica o texto a direita;
- `justify`: ajusta o espaço entre as palavras para que o texto ocupe todo o espaço disponível em uma linha.

```
p{
```

```
    text-align: center;
```

```
}
```



line-height

Podemos controlar a altura da linha de um texto com a propriedade `line-height`. Geralmente utilizamos a unidade `em` como valor para essa propriedade para que a distância entre as linhas do texto acompanhe o tamanho da fonte. Abaixo podemos ver um exemplo:

```
P{  
  font-size: 14px;  
  line-height: 1.6em;  
}
```



Box Model



O navegador representa cada elemento como uma caixa retangular, de acordo com o padrão definido pelo CSS conhecido como **box (caixa) model**. Dessa forma, o conteúdo do elemento é uma das quatro partes que compõem o box, sendo as demais o seu preenchimento, borda e margem.

Sempre quando vamos estilizar algum elemento via CSS, é comum, e precisamos estar cientes, que precisamos ver os elementos como caixas. Isso facilita compreender o conceito de box model.



Vídeo de ajuda [Box Model](#)



Box Model



Basicamente, a idéia do box model é composta por quatro partes:

conteúdo
padding
border
margin





Box Model



```
.elemento {  
    width: 50px; /* Largura */  
    height: 50px; /* Altura */  
    border: 1px solid gray; /* Borda */  
    padding: 10px 20px; /* Preenchimento */  
}
```

width

A propriedade CSS width determina a largura da área de conteúdo de um elemento.

height

A propriedade height do CSS determina a altura da área do conteúdo de um elemento.

padding

A propriedade padding define uma distância entre o conteúdo de um elemento e suas bordas.





Box Model



A propriedade **box-sizing** define como a largura e a altura de um elemento são calculadas: devem incluir preenchimento e bordas ou não.

box-sizing: content-box | border-box | initial | inherit;

Propriedades:

Content-box - Predefinição. As propriedades de largura e altura (e propriedades min/max) incluem apenas o conteúdo. Borda e preenchimento não estão incluídos

Border-box - As propriedades de largura e altura (e propriedades min/max) incluem conteúdo, preenchimento e borda

Initial - Configura esta propriedade para seu valor padrão.

Inherit - Herda esta propriedade de seu elemento pai.



Elementos Flutuantes

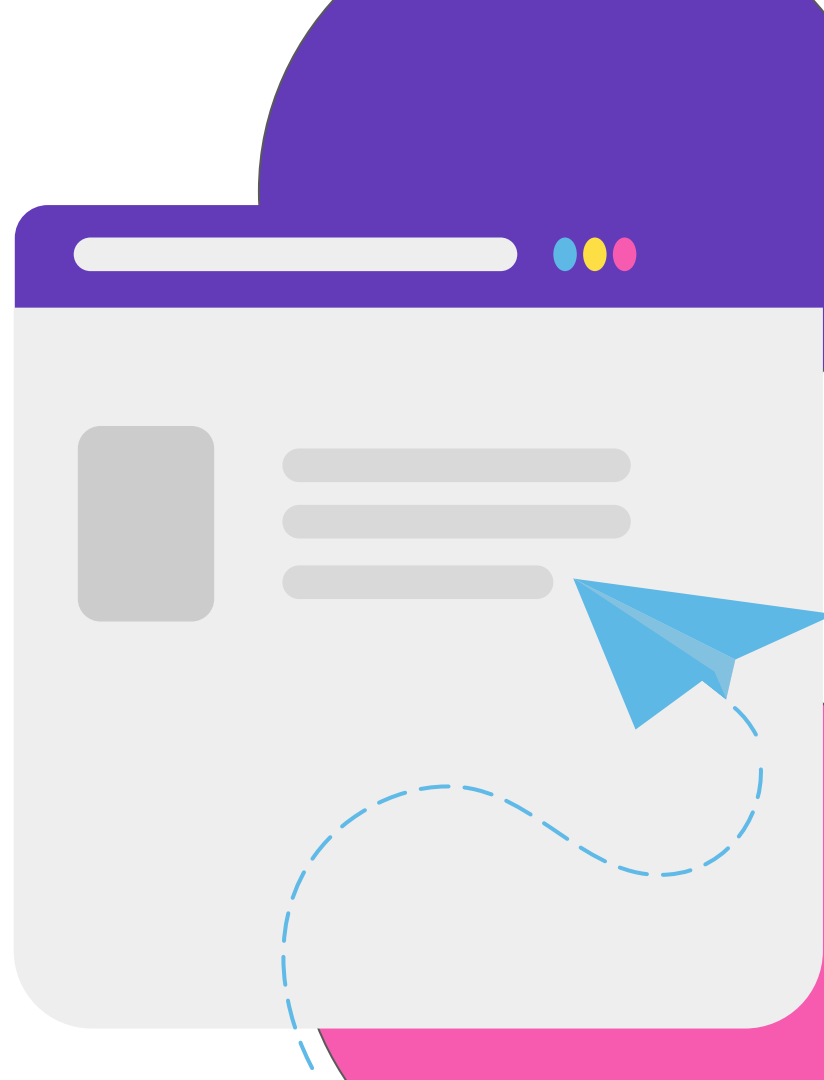
Definição

A propriedade float do CSS determina que um elemento deve ser retirado do seu fluxo normal e colocado ao longo do lado direito ou esquerdo do seu container, onde textos e elementos em linha irão se posicionar ao seu redor.

Sintaxe:

float: propriedade;

Vídeo de ajuda [Float](#)



Elementos Flutuantes

Propriedades

left - O elemento flutua à esquerda de seu contêiner

right - O elemento flutua à direita de seu contêiner

none - O elemento não flutua (será exibido exatamente onde ocorre no texto). Isso é padrão





Posicionamento



A **position** propriedade especifica o tipo de método de posicionamento usado para um elemento (estático, relativo, absoluto, fixo ou fixo).

Relativo - O elemento é posicionado em relação à sua posição normal, então "esquerda:20px" adiciona 20 pixels à posição esquerda do elemento.

Estático - Valor padrão. Os elementos são renderizados em ordem, conforme aparecem no fluxo de documentos

Absoluto - O elemento é posicionado em relação ao seu primeiro elemento ancestral posicionado (não estático)

Fixo - O elemento está posicionado em relação à janela do navegador



Vídeo de ajuda [Position](#)




position: relative;



A primeira opção de posicionamento para um elemento é a **relative**. Como o nome sugere, ela especifica uma posição relativa do elemento em relação a algo: o elemento pai.

A posição é definida através de quatro outras propriedades: **top**, **bottom**, **left** e **right**, que indicam a distância com relação ao topo, base, esquerda e direita, respectivamente, tomando como referência o posicionamento do elemento pai.






position: absolute;



A segunda opção de posicionamento é a **absolute**, ou absoluta. Ela faz com que o elemento “saia” da hierarquia estabelecida pelo HTML – na prática, ele é filho da página e não de um elemento qualquer no HTML definido. Assim, é possível posicionar esse elemento em qualquer lugar da página, independentemente do que temos lá. Isso pode ser útil em casos que precisamos posicionar um alerta em nossa página, por exemplo, e não queremos “quebrar” o layout.

Apesar dessa possibilidade, é importante utilizar o layout absoluto de forma responsável, uma vez que ele, por não fazer parte da hierarquia da página, pode atrapalhar o desenvolvimento responsivo. Assim, o funcionamento em dispositivos com resoluções menores, como tablets e smartphones, pode ser prejudicado.

Na prática, para um HTML simples, como o apresentado a seguir, o <div> será filho de <body>. Entretanto, como especificamos um posicionamento absoluto, o elemento pai não nos interessa, uma vez que não consideramos seu posicionamento como referência.






position: fixed;

Por fim, temos a opção **fixed**, que funciona de forma diferente em relação às demais. Embora seja parecida com o posicionamento absoluto, com **fixed** o elemento “responde”, em termos de layout, ao navegador. Isso significa que, mesmo que navegemos na página para cima e para baixo, o elemento ficará fixo na mesma posição sempre.

Esse tipo de posicionamento é especialmente útil em casos de menu, para que o usuário não perca o contato com a navegação enquanto lê uma notícia, por exemplo. Sites com conteúdo extenso costumam adotar essa opção para melhorar a experiência do usuário.



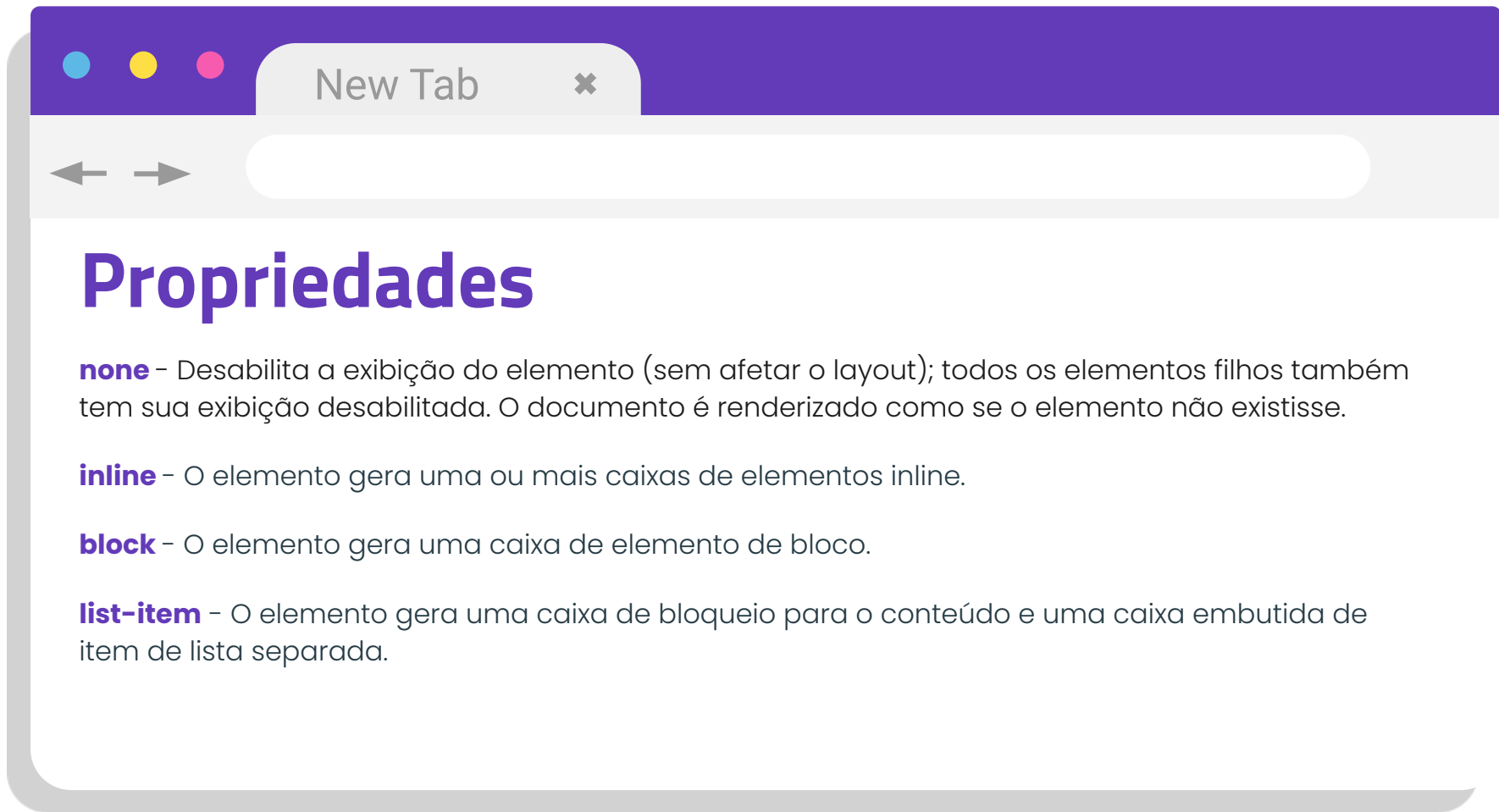
Display

Definição

A propriedade CSS display especifica o tipo de caixa de renderização usada por um elemento. No HTML, os valores padrões da propriedade display são feitos a partir do comportamento descrito nas especificações HTML ou da folha de estilo padrão do navegador/usuário. O valor padrão em XML é inline.

Vídeo de ajuda [Display](#)







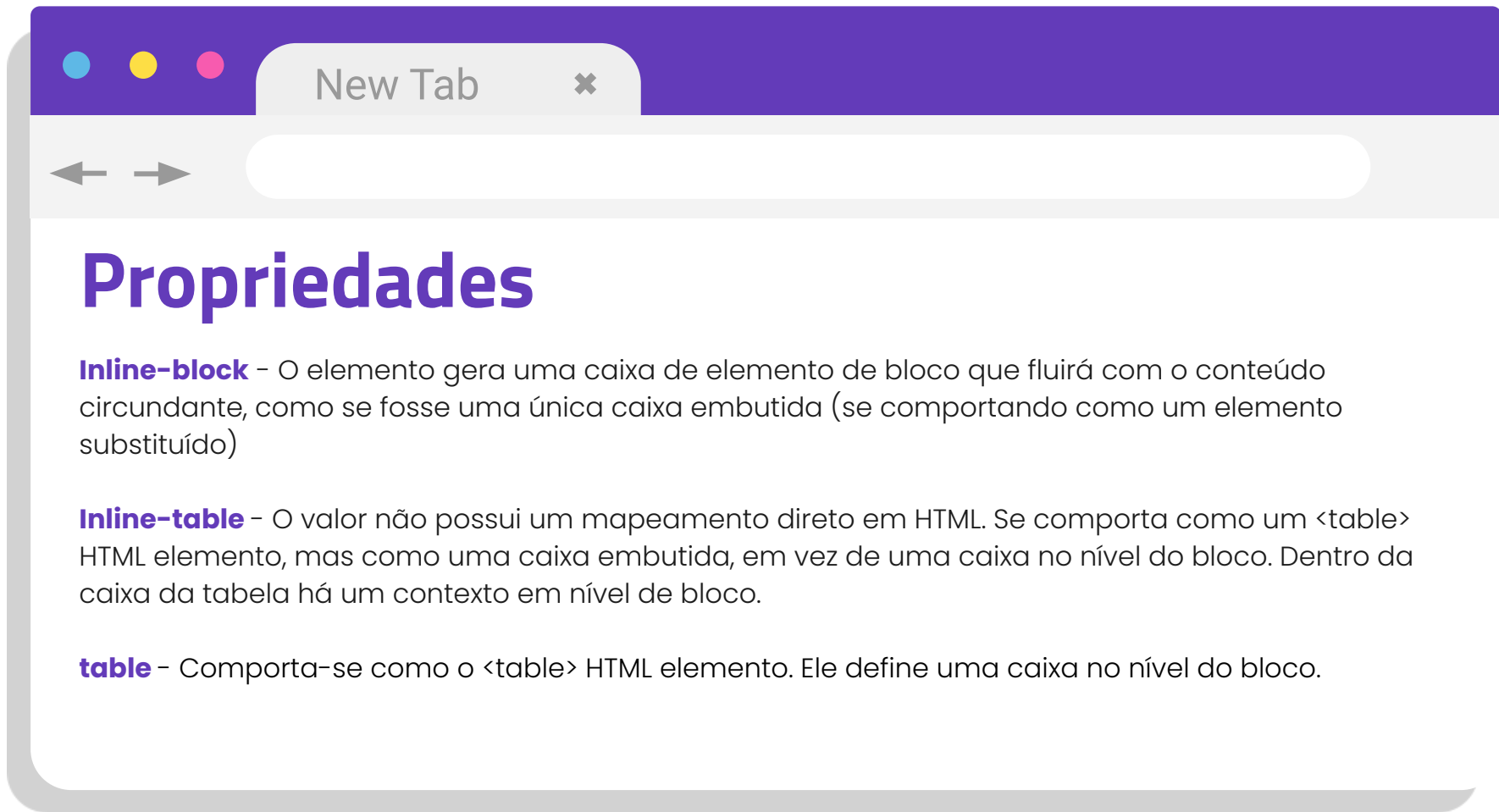
Propriedades

Flex - O elemento se comporta como um elemento de bloco e apresenta seu conteúdo de acordo com o modelo flexbox.

Inline-flex - O elemento se comporta como um elemento embutido e apresenta seu conteúdo de acordo com o modelo flexbox.

Grid - O elemento se comporta como um elemento de bloco e apresenta seu conteúdo de acordo com o modelo de grade.

Inline-grid - O elemento se comporta como um elemento embutido e apresenta seu conteúdo de acordo com o modelo de grade.



Propriedades

Inline-block - O elemento gera uma caixa de elemento de bloco que fluirá com o conteúdo circundante, como se fosse uma única caixa embutida (se comportando como um elemento substituído)

Inline-table - O valor não possui um mapeamento direto em HTML. Se comporta como um `<table>` HTML elemento, mas como uma caixa embutida, em vez de uma caixa no nível do bloco. Dentro da caixa da tabela há um contexto em nível de bloco.

table - Comporta-se como o `<table>` HTML elemento. Ele define uma caixa no nível do bloco.

FlexBox

Tem como função ser um modo mais eficiente para criar layouts, alinhar e distribuir espaços entre itens de um container, até mesmo quando possuem dimensões não declaradas e/ou dinâmicas (foi onde surgiu o termo “flex”)

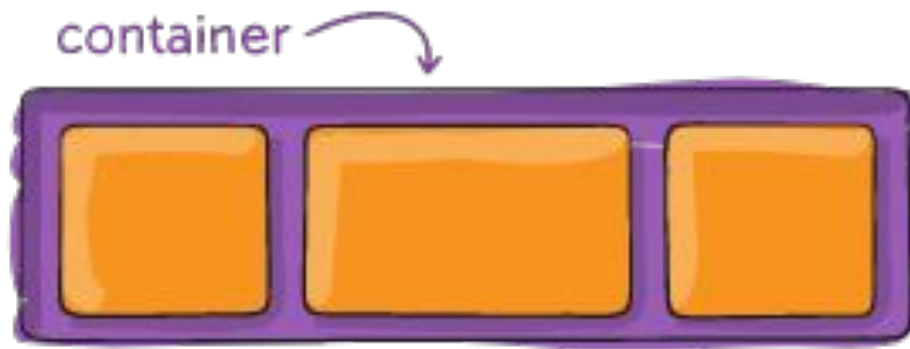
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

```
.flex-container {
  display: flex;
}
```

Vídeo de ajuda [FlexBox](#)

Propriedades do elemento-pai

Quando utilizamos o *Flexbox*, é muito importante saber quais propriedades são declaradas no elemento-pai (por exemplo, uma *div* que irá conter os elementos a serem alinhados) e quais serão declaradas nos elementos-filhos.



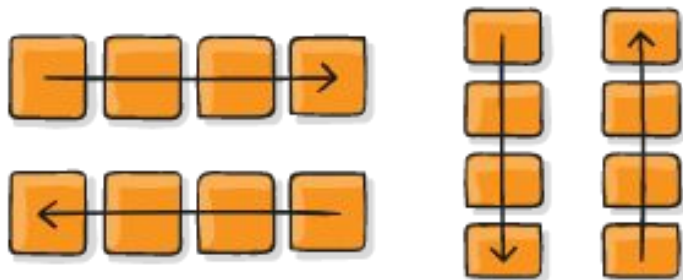


Display

Esta propriedade define um *flex container*; inline ou block dependendo dos valores passados. Coloca todos os elementos-filhos diretos num contexto Flex.

```
.div{  
  display: flex;  
}
```

Flex Direction



Estabelece o eixo principal, definindo assim a direção em que os *flex items* são alinhados no *flex container*. O Flexbox é (com exceção de um wrapping opcional) um conceito de layout de uma só direção. Pense nos *flex items* inicialmente posicionados ou em linhas horizontais ou em colunas verticais.

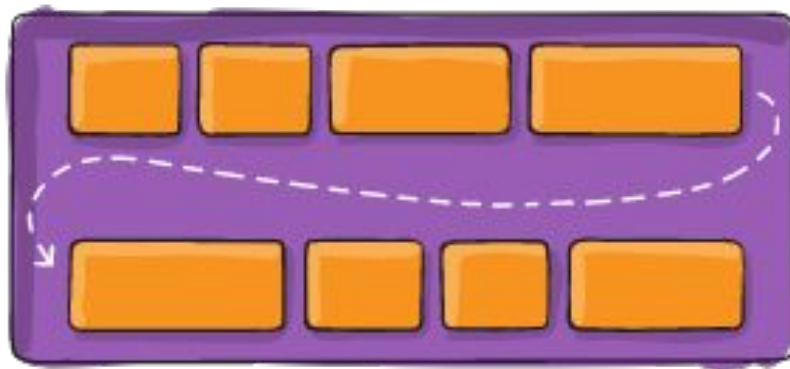
Flex Direction - Propriedades



```
.flex-container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

- **row (padrão):** alinha os itens da esquerda para a direita
- **row-reverse:** alinha os itens da direita para a esquerda
- **column:** ordena os itens um em cima do outro.
- **column-reverse:** ordena os itens um em cima do outro porém em ordem reversa.

Flex Wrap



Por padrão, os *flex items* vão todos tentar se encaixar em uma só linha. Com esta propriedade você pode modificar esse comportamento e permitir que os itens quebrem para uma linha seguinte conforme for necessário.

Flex Wrap - Propriedades



```
.flex-container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

- **nowrap** (padrão): todos os *flex items* ficarão em uma só linha
- **wrap**: os *flex items* vão quebrar em múltiplas linhas, de cima para baixo
- **wrap-reverse**: os *flex items* vão quebrar em múltiplas linhas de baixo para cima

Flex Flow



```
.flex-container {  
  flex-flow: row nowrap | row wrap | column nowrap | column wrap;  
}
```

A propriedade **flex-flow** é uma propriedade *shorthand* (uma mesma declaração inclui vários valores relacionados a mais de uma propriedade) que inclui **flex-direction** e **flex-wrap**. Determina quais serão os eixos principal e transversal do container. O valor padrão é row **nowrap**.

Justify Content

Esta propriedade define o alinhamento dos itens ao longo do eixo principal. Ajuda a distribuir o espaço livre que sobra no container tanto se todos os flex items em uma linha são inflexíveis, ou são flexíveis mas já atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento de itens quando eles ultrapassam o limite da linha.

flex-start



flex-end



center



space-between



space-around



space-evenly



Justify Content

```
.flex-container {  
  justify-content: flex-start | flex-end | center | space-between | space-around |  
}
```

Esta propriedade define o alinhamento dos itens ao longo do eixo principal. Ajuda a distribuir o espaço livre que sobra no container tanto se todos os flex items em uma linha são inflexíveis, ou são flexíveis mas já atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento de itens quando eles ultrapassam o limite da linha.



Justify Content

Justify-content: **propriedade**;

flex-start

Os itens são alinhados na borda superior da direita do container.

flex-end

Os itens são alinhados na borda inferior da esquerda do container.

center

Os itens são alinhados no centro do container.



Justify Content

Justify-content: **propriedade**;

flex-start

Os itens são alinhados na borda superior da direita do container.



flex-end

Os itens são alinhados na borda inferior da esquerda do container.



center

Os itens são alinhados no centro do container.





Justify Content

Justify-content: **propriedade;**

space-between

Os itens são alinhados com o espaço entre eles.



space-evenly

Os itens são alinhados com espaços entre eles e a metade do espaço nas laterais



space-around

Os todos os lados possuem medidas iguais





Align Content

align-content: **propriedade**;

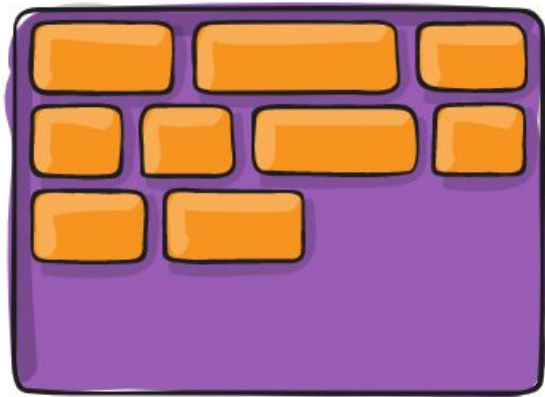
Organiza as linhas dentro de um flex container quando há espaço extra no eixo transversal, similar ao modo como **justify-content** alinha itens individuais dentro do eixo principal.

Importante: Esta propriedade não tem efeito quando há somente uma linha de flex items no container.

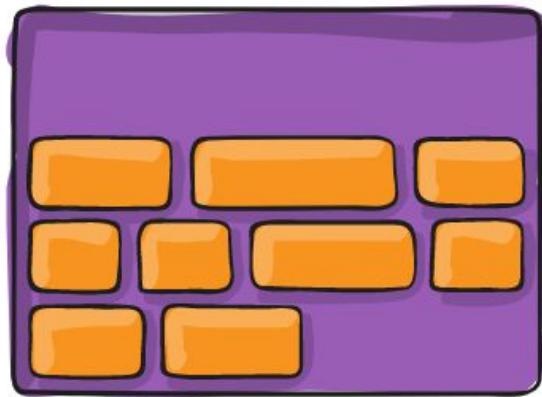
Align Content

```
align-content: propriedade;
```

flex-start



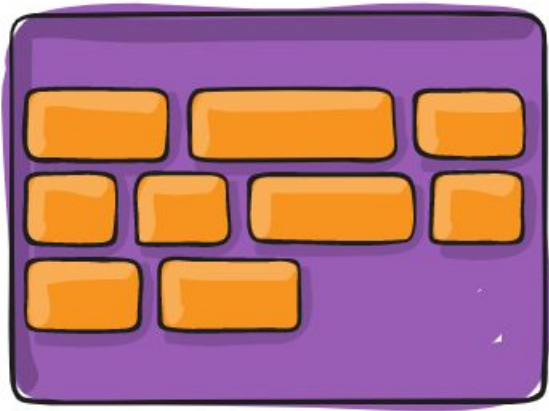
flex-end



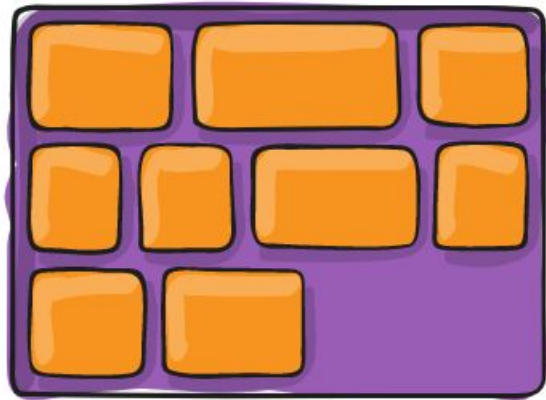
Align Content

```
align-content: propriedade;
```

center



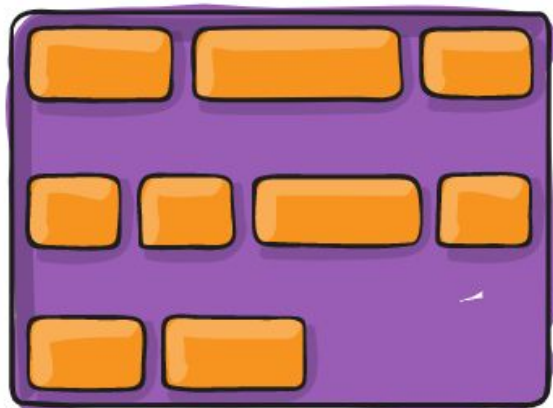
stretch



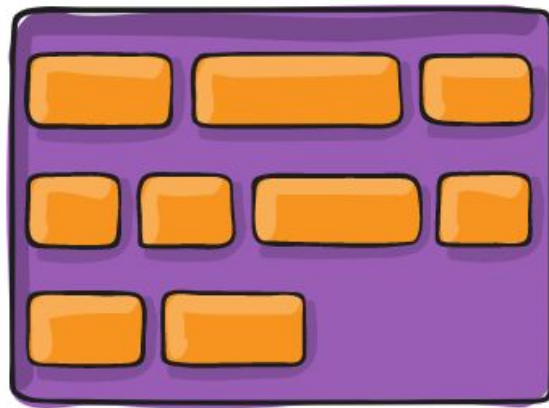
Align Content

```
align-content: propriedade;
```

space-between



space-around



Responsivo

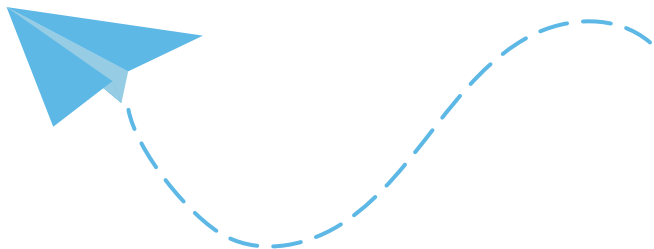
Design Responsivo é uma técnica de estruturação HTML e CSS, que consiste em adaptar o site ao browser do usuário sem que seja necessário definir várias folhas de estilos específicas para cada resolução, ou seja, é um tipo de design onde o layout fica fluído e variante de acordo com a resolução do usuário.



Use **Media Queries**

Para designs responsivos você precisa ter foco nas condições de width, onde dependendo da largura do conteúdo de seu cliente, ou seja, dependendo da resolução do dispositivo que seu cliente estiver acessando o website, irá ser carregada uma folha de estilos diferente ou uma folha de estilo específica.

Vídeo de ajuda [Responsivo](#)



Exemplos



Exemplo de diferentes css em diferentes resoluções

```
<style>  
  @import url(pequeno.css) (min-width: 300px);  
  @import url(medio.css) (min-width: 600px);  
  @import url(grande.css) (min-width: 900px);  
</style>
```





Exemplos



Também podemos ver alguns exemplos de resoluções abaixo:

Também podemos ver alguns exemplos de resoluções abaixo:

<style>

Largura de 200px até 640px

@media screen and (min-width: 200px) **and** (max-width:640px)

Orientação paisagem de pelo menos 600px

@media screen and (min-width: 600px) **and** (orientation:landscape)

Orientação retrato de pelo menos 380px

@media screen and (max-width: 380px) **and** (orientation:portrait)

</style>





Exemplos



Também podemos ver alguns exemplos de resoluções abaixo:

Por exemplo, a seguinte media querye testa se a página atual está sendo exibida como mídia de tela (portanto, não é um documento impresso) e o viewport tem pelo menos 800 pixels de largura. O CSS para o seletor .container será aplicado apenas se essas duas condições forem verdade.

```
@media screen and (min-width: 800px){  
  .container{  
    margin: 1em 2em;  
  }  
}
```

