

# **LAYOUT E FLEXBOX: INTRODUÇÃO AO POSICIONAMENTO DE ELEMENTOS NA PAGINA**

**(AULA 23 - PARTE1)**

**CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT**

**MAYARA MARQUES**

**mmrosatab@gmail.com**

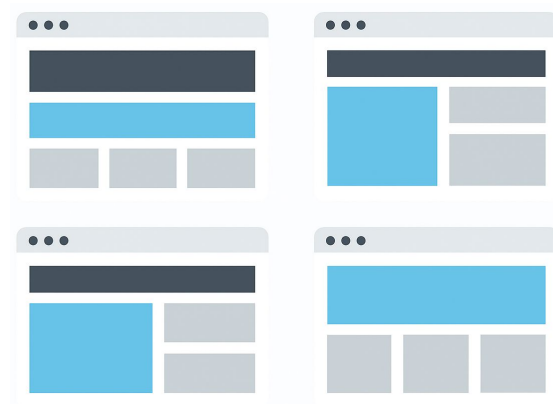
# *SUMÁRIO*

- Introdução a Layout e Posicionamento
  - Fluxo normal (block vs inline)
  - Box model (margin, border, padding, content)
  - Tipos de position (relative, absolute, fixed, sticky)
- Mão na massa

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

O que é Layout no Front-end?

Layout é a estrutura visual de uma página. Define **como** os elementos (textos, imagens, botões, menus) são organizados. Um bom layout melhora usabilidade, experiência do usuário e estética.



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

## A web no início

Nos anos 90, páginas eram lineares, como um documento de texto corrido. Tudo ficava em uma coluna única, sem muita estilização. Elementos eram posicionados de forma estática e sem recursos de interação. Desenvolvedores usavam até tabelas HTML para simular layout, o que hoje é considerado uma má prática.



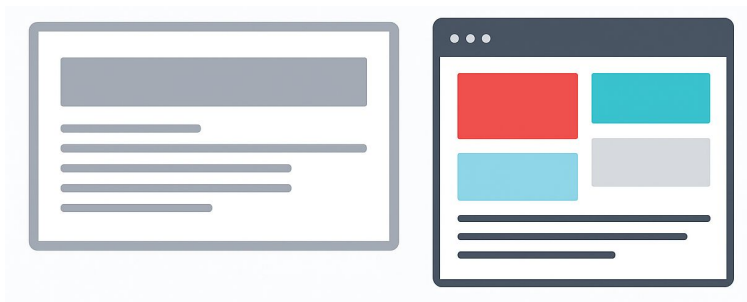
# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

A evolução com CSS

Com o CSS, o layout deixou de ser apenas "texto na tela".

Foi possível:

- Controlar cores, margens e espaçamentos.
- Organizar elementos em blocos.
- Criar layouts mais ricos.



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

A era dos layouts modernos

Hoje, temos recursos que facilitam a construção de páginas responsivas. O layout pode se adaptar a diferentes tamanhos de tela (desktop, tablet, celular). Isso é fundamental porque o acesso à internet é feito, em grande parte, pelo mobile.



# ***INTRODUÇÃO A LAYOUT E POSICIONAMENTO***

Por que estudar Layout?

- Permite criar páginas organizadas e bonitas.
- Ajuda a destacar informações importantes.
- Melhora a experiência do usuário (UX).
- É a base para construir qualquer interface profissional no front-end.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Fluxo normal (block vs inline)

Quando **não aplicamos nenhuma regra especial de CSS em um elemento**, ele segue o chamado fluxo “*normal*” da página. Esse fluxo define como os elementos são dispostos naturalmente no HTML.



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Fluxo normal (block vs inline)

**Block** (em bloco)

- Ocupam toda a largura disponível do elemento pai, mesmo que o conteúdo seja pequeno.
- Sempre começam em uma nova linha, empurrando os próximos elementos para baixo.
- Permitem aplicar propriedades como width, height, margin e padding em todas as direções.
- São usados para estruturar a página em seções.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Block (em bloco)

Ex:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div>Sou um bloco</div>
  <p>Parágrafo é bloco</p>
  <h1>Título é bloco</h1>
  <script src="./script.js">
</script>
</body>
</html>
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Sou um bloco

Parágrafo é bloco

Título é bloco

O elemento **h1** ocupou toda a largura disponível do elemento **body**.

```
<!DOCTYPE html>
<html lang="pt-br" == $0
  <head> ... </head>
  <body>
    <div>Sou um bloco</div>
    <p>Parágrafo é bloco</p>
    <h1>Título é bloco</h1>
    <script src="./script.js"> </script>
    <!-- Code injected by live-server -->
    <script> ... </script>
  </body>
</html>
```

html

Styles Computed Layout Event Listeners >>

Filter :hov .cls + - [ ]

```
element.style {
}

html[Attributes Style] {
  -webkit-locale: "pt-br";
}
```

:root { user agent stylesheet

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Fluxo normal (block vs inline)

**Inline** (em linha)

- Elementos inline ocupam somente o espaço necessário para seu conteúdo (texto, imagem, link).
- Eles não iniciam uma nova linha; ao contrário, ficam "fluindo" dentro da mesma linha com outros elementos.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Inline (em linha)

Ex:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>Texto <span>inline</span> continua na mesma linha.</p> }
  <script src="./script.js">
  </script>
</body>
</html>
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/index.html". The page content shows the text "Texto inline continua na mesma linha." where the word "inline" is highlighted in blue. A small tooltip above the word "span" indicates its dimensions: "36x14 x 17.71". A callout box points to the word "span" with the text: "O elemento **span** ocupou apenas o espaço necessário dentro do elemento **p**."

The browser's developer tools are open, showing the "Elements" panel on the right. The HTML structure is as follows:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <p>
      "Texto "
      <span>inline</span>
      " continua na mesma linha."
    </p>
    <script src="script.js"></script>
    <!-- Code injected by live-server -->
    <script></script>
  </body>
</html>
```

The "Styles" panel is also visible, showing the default user agent styles for the `html` element:

```
html {
  -webkit-locale: "pt-br";
}
```

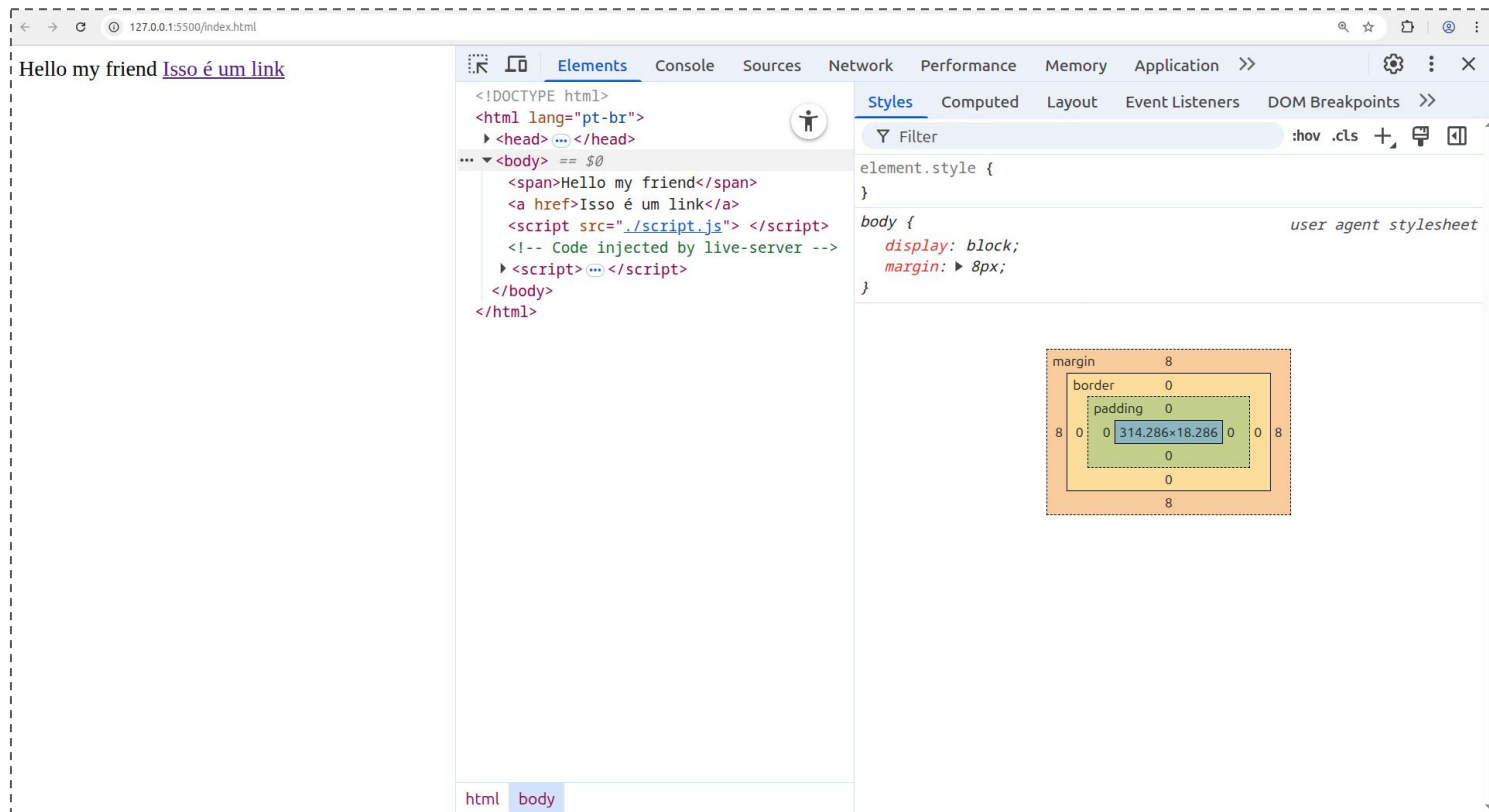
# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Inline (em linha)

Ex:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <span>Hello my friend</span> }
  <a href="">Isso é um link</a>
  <script src="./script.js">
</script>
</body>
</html>
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO





# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

## Resumo

Block: ocupa toda a largura disponível, quebra a linha.

Ex: <div>, <p>, <h1>.

Inline: ocupa apenas o espaço necessário, não quebra a linha.

Ex: <span>, <a>, <strong>.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Elementos de bloco mais comuns

- `<div>`
- `<p>`
- `<h1>`, `<h2>`, ... `<h6>`
- `<section>`, `<article>`, `<header>`, `<footer>`, `<main>`
- `<ul>`, `<ol>`, `<li>`
- `<form>`
- `<table>`

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Elementos inline mais comuns

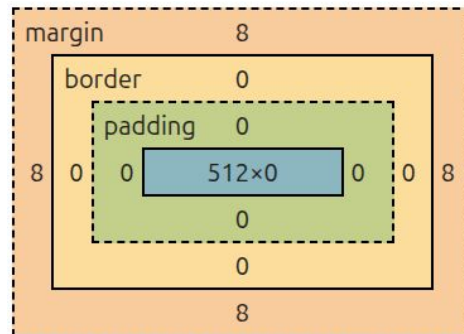
- `<span>`
- `<a>`
- `<strong>`, `<em>`
- `<b>`, `<i>`
- `<img>`
- `<label>`

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

O Box Model descreve como o navegador calcula o espaço ocupado por um elemento na página. Todo elemento HTML é representado como uma caixa retangular, composta por quatro camadas:

1. Content (conteúdo)
2. Padding (preenchimento interno)
3. Border (borda)
4. Margin (margem)



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

### 1. Content (conteúdo)

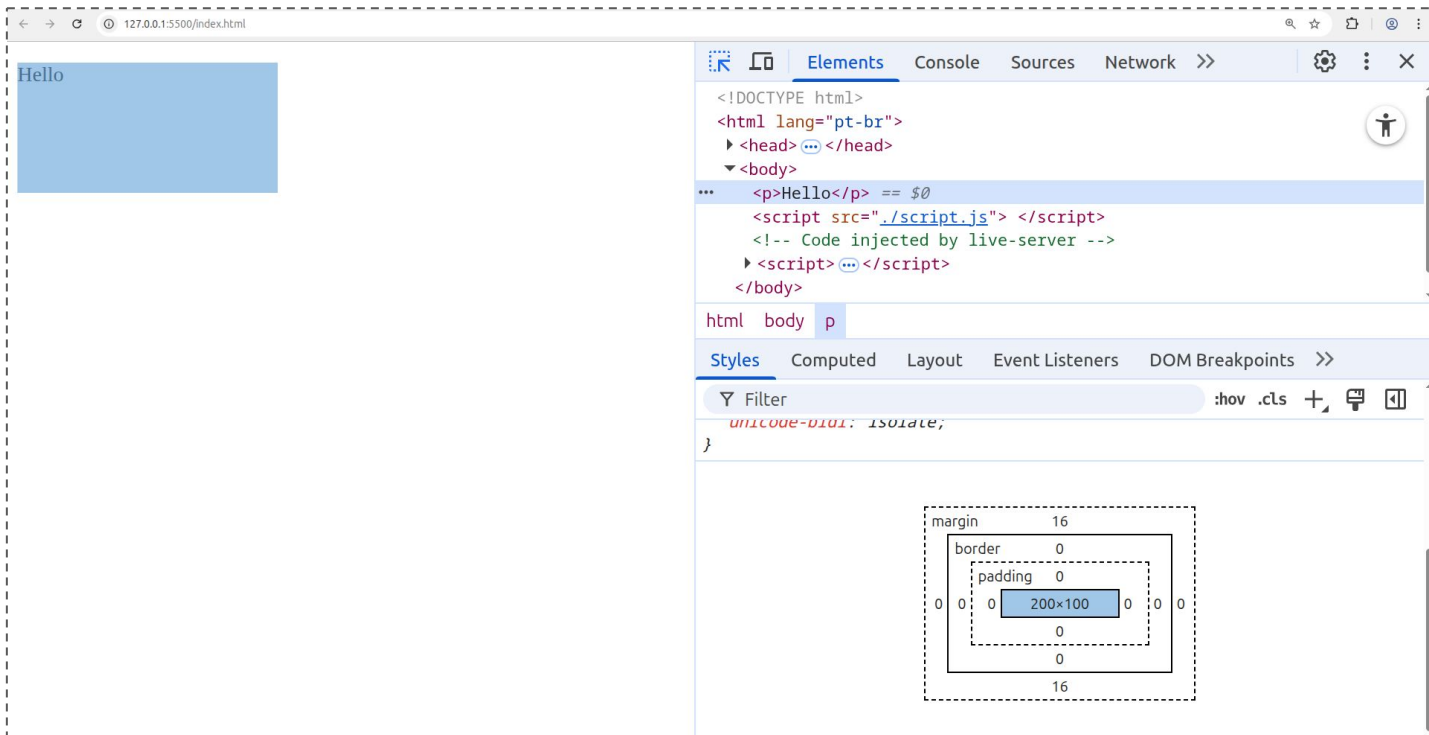
É a parte central da caixa. Contém o texto, imagens ou outros elementos renderizados.

Propriedades relacionadas: width e height.

Exemplo:

```
p {  
  width: 200px;  
  height: 100px;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Box model

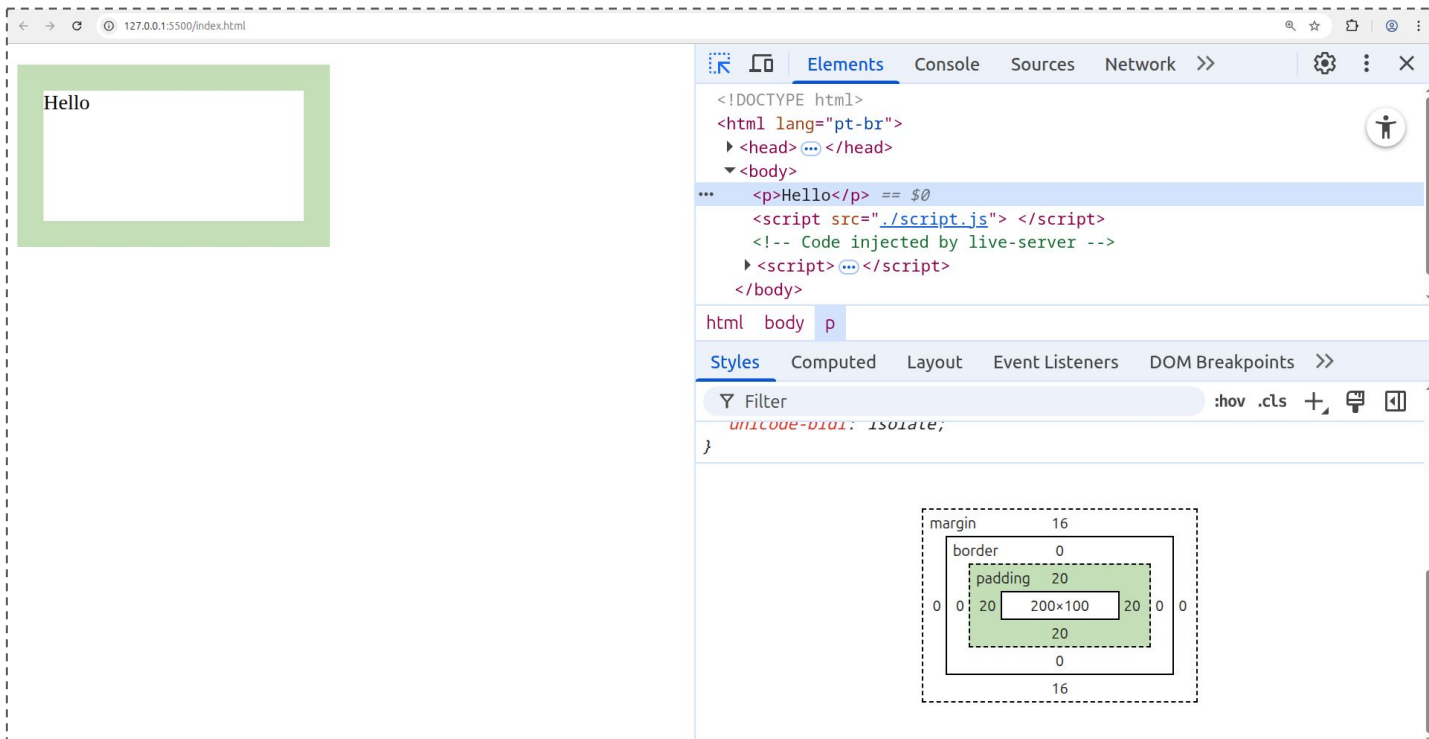
## 2. Padding (preenchimento interno)

Espaço entre o conteúdo e a borda. Faz o "respiro" interno, sem mover a caixa para fora.

Exemplo:

```
p {  
  padding: 20px;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO





# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Box model

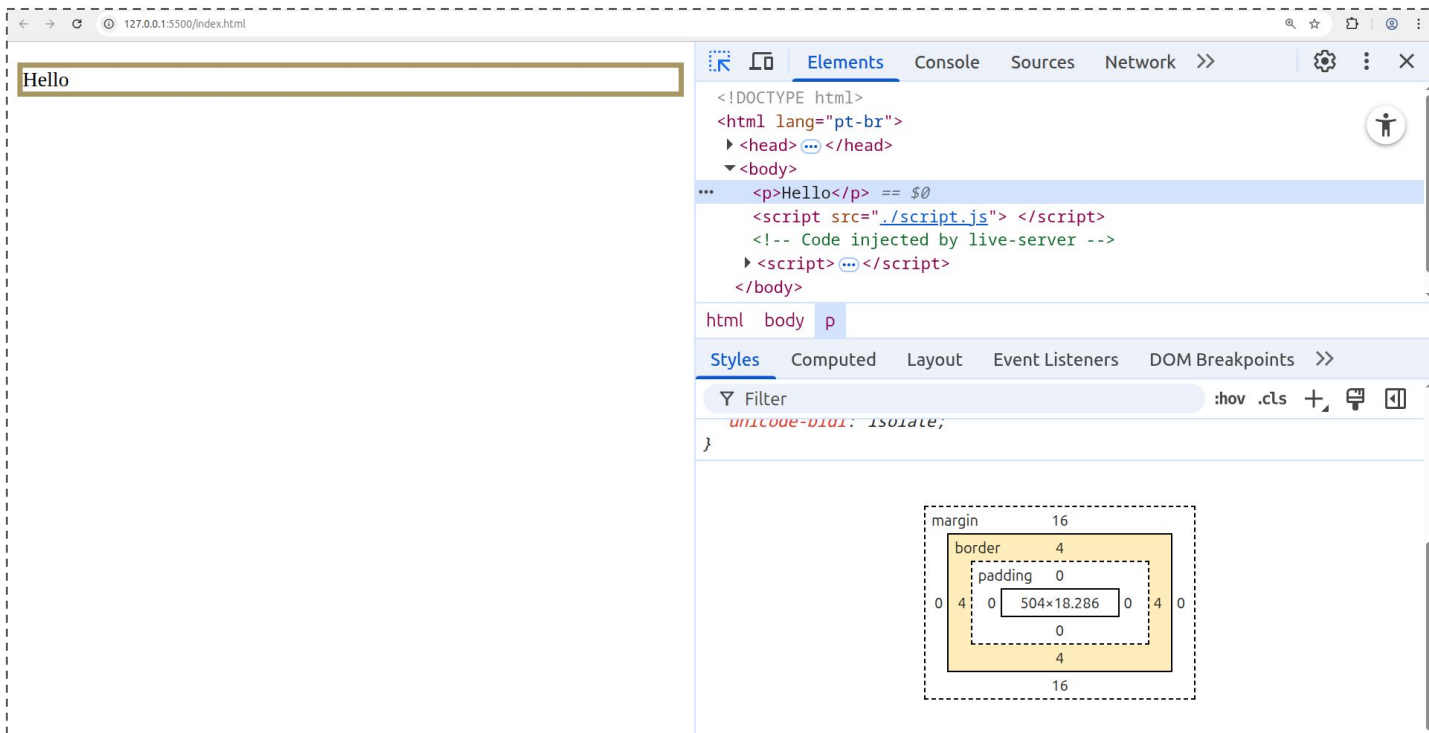
## 3. Border (borda)

Envolve o padding e o conteúdo. Pode ter largura, cor e estilo (solid, dashed, dotted etc.).

Exemplo:

```
p {  
  border: 4px solid black;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Box model

## 4. Margin (margem)

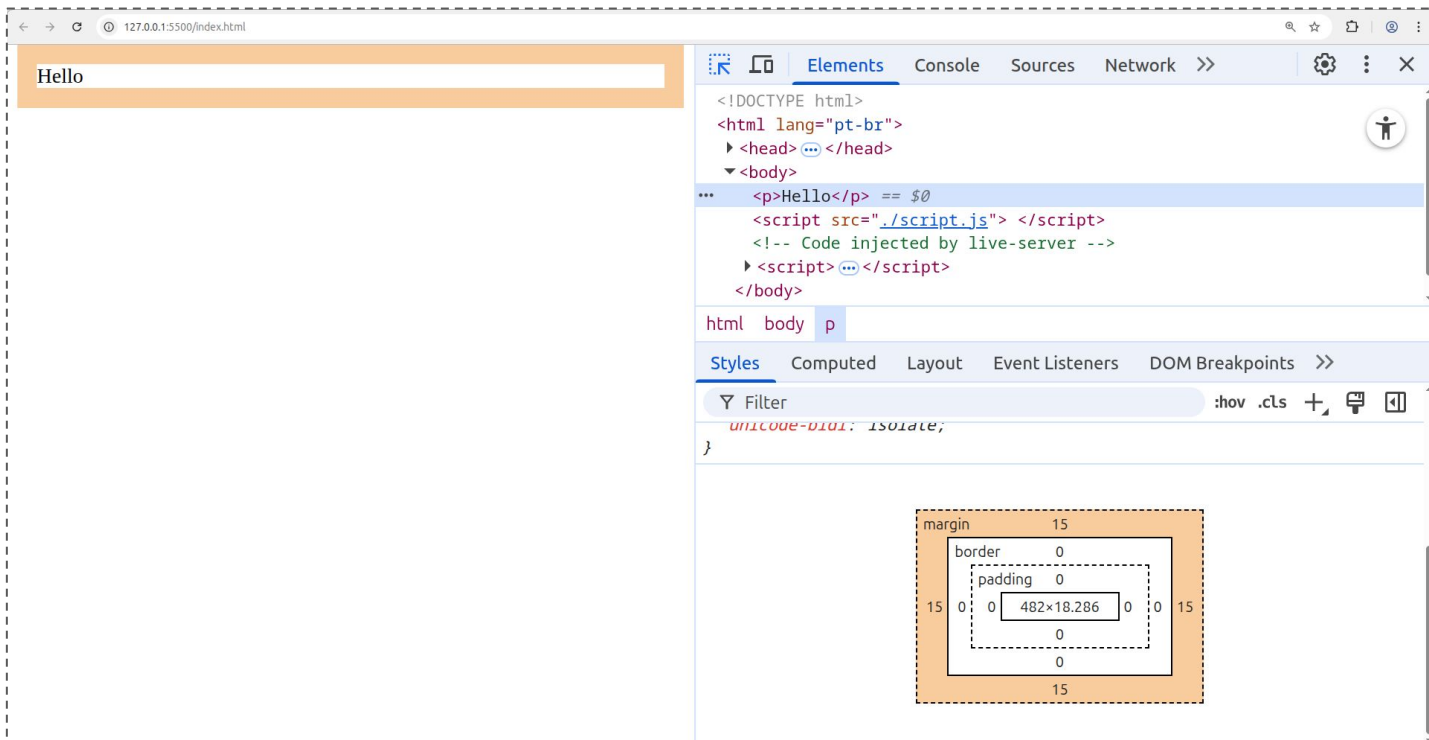
Espaço externo, entre a borda do elemento e os elementos vizinhos.

Afasta um elemento do outro.

Exemplo:

```
p {  
  margin: 15px;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

Como o navegador calcula o tamanho total da caixa?

```
div {  
  width: 200px;  
  padding: 10px;  
  border: 8px solid;  
  margin: 20px;  
}
```

Content/width = 200px

Padding = 10px + 10px = 20px

Border = 8px + 8px = 16px

Margin = 20px + 20px = 40px

Largura total = 276px

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

## Box model

Como o navegador calcula o tamanho total da caixa?

```
largura total = width + padding esquerdo + padding direito + border esquerda + border direita + margin  
esquerda + margin direita
```

```
altura total = height + padding top + padding bottom + border top + border bottom + margin top + margin  
bottom
```

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Box model

Mas o comportamento muda conforme o tipo do elemento:

Elementos de bloco (block)

- Respeitam width, height, margin, padding, border normalmente.
- Ex: <div>, <p>, <section>

Se você coloca width: 300px, a caixa realmente tem essa largura.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Box model

Elementos inline

O box model existe, mas com algumas restrições:

- Não dá pra usar width e height diretamente (são ignorados).
- O tamanho do content depende do texto ou do conteúdo interno.
- Padding, border e margin funcionam só nos lados horizontais (esquerda/direita).
- Nos lados verticais (cima/baixo), até aparecem visualmente, mas não deslocam os elementos vizinhos no fluxo.
- Ex: <span>, <a>, <strong>



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

Por padrão, o CSS soma tudo (content + padding + border).

```
div {  
  height: 100px;  
  padding: 20px;  
  border: 5px solid;  
  margin: 15px;  
  box-sizing: content-box; /* padrão */  
}
```

Altura final da caixa renderizada:

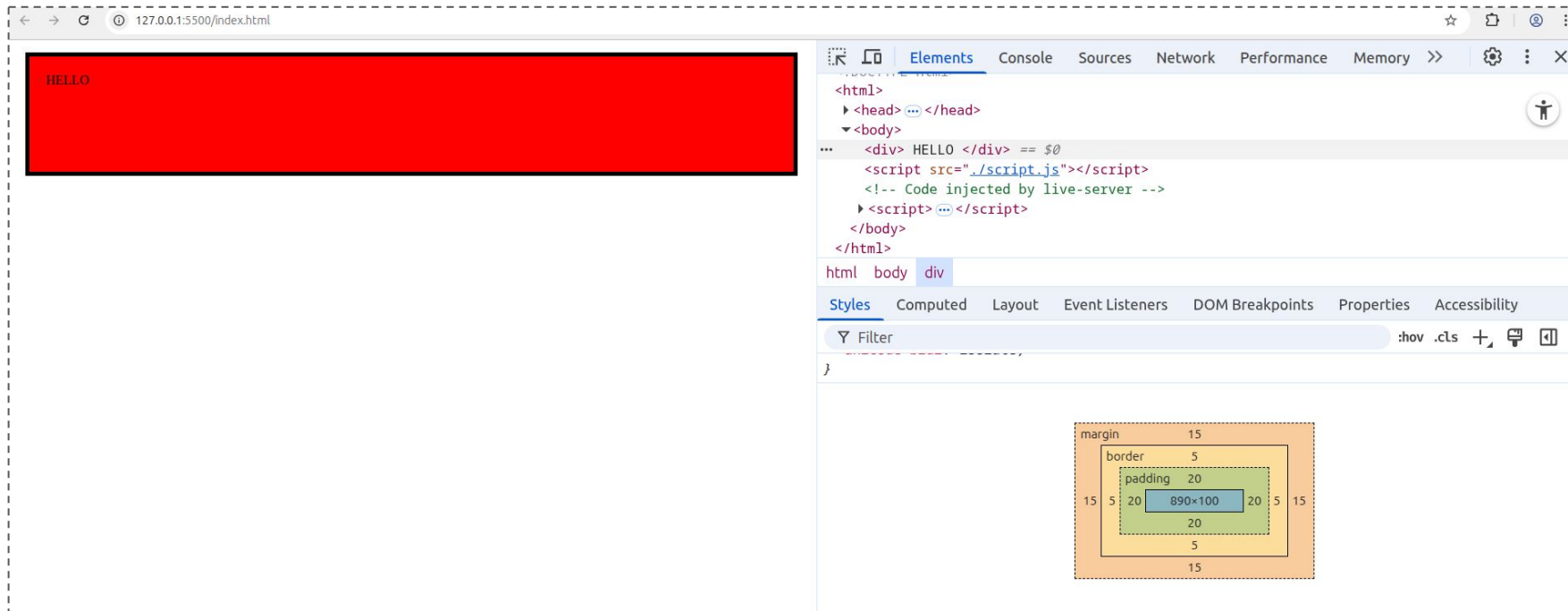
100 (content) + 20 + 20 (padding top/bottom) + 5 + 5 (border) = **150px**

Espaço total ocupado na tela (considerando margin):

**150px** + 15px (top) + 15px (bottom) = 180px

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model



The screenshot displays a web browser window with a red rectangular box containing the text "HELLO". The browser's address bar shows the URL "127.0.0.1:5500/index.html". The Chrome DevTools interface is open, showing the "Elements" panel on the right, which displays the HTML structure of the page. The "Styles" panel is also open, showing the default styles for the selected "div" element. The box model diagram is visible in the Styles panel, illustrating the layout of the box with its margin, border, padding, and content dimensions.

**Elements Panel:**

```
<html>
  <head> </head>
  <body>
    <div> HELLO </div> == $0
    <script src="./script.js"></script>
    <!-- Code injected by live-server -->
    <script> </script>
  </body>
</html>
```

**Styles Panel:**

html body div

Filter

margin 15

border 5

padding 20

890x100

20 5 15

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

Por padrão, o CSS soma tudo (content + padding + border).

Se quisermos que width e height já inclua **padding** e **border**, podemos utilizar o border-box:

```
box-sizing: border-box;
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

Por padrão, o CSS soma tudo (content + padding + border).

Se quisermos que width e height já inclua **padding** e **border**, podemos utilizar o **border-box**:

```
div {  
  height: 100px;  
  padding: 20px;  
  border: 5px solid;  
  margin: 15px;  
  box-sizing: border-box;  
}
```

Altura final da caixa renderizada:  
100px fixos (content + padding + border juntos).  
O conteúdo interno **diminui** pra caber dentro.

Espaço total ocupado na tela:  
 $100\text{px} + 15\text{px}(\text{margin}) + 15\text{px}(\text{margin}) = 130\text{px}$

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Box model

```
box-sizing: border-box;
```

The screenshot shows a web browser window with a red rectangular box containing the text "HELLO". The browser's address bar shows the URL "127.0.0.1:5500/index.html". The Chrome DevTools interface is open, with the "Elements" panel on the right. The "div" element is selected, showing its HTML structure: 

```
<html>  
  <head> ... </head>  
  <body>  
    <div> HELLO </div> == $0  
    <script src="/script.js"></script>  
    <!-- Code injected by live-server -->  
    <script> ... </script>  
  </body>  
</html>
```

  
The "Styles" panel is also open, showing the "border" property with a value of "5px solid black". Below the panels, a diagram illustrates the box model for the selected element. The diagram shows a blue box (content) with dimensions 890x50. It is surrounded by a green box (padding) with dimensions 20x20. This is further surrounded by an orange box (border) with dimensions 5x5. The outermost box (margin) has dimensions 15x15. The total dimensions of the box model are 930x75.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

A screenshot of a web browser showing a red rectangular box with the text "HELLO". The browser's developer tools are open, displaying the HTML and CSS for the page.

The HTML structure is as follows:

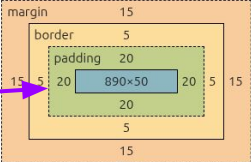
```
<!DOCTYPE html>
<html>
  <head>
    <script src="./script.js"></script>
  </head>
  <body>
    <div> HELLO </div>
  </body>
</html>
```

The CSS rules for the `div` element are:

```
div {
  height: 100px;
  padding: 20px;
  border: 5px solid;
  margin: 15px;
  box-sizing: border-box;
  background-color: red;
}
```

The `display` property is set to `block` in the user agent stylesheet.

A diagram at the bottom right illustrates the `border-box` model. It shows a central content area (890x50) surrounded by padding (20px), a border (5px), and a margin (15px). The total dimensions of the box are 990x120px.



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

The screenshot shows a web browser at `127.0.0.1:5500/index.html` displaying a red rectangular box with the text "HELLO". The Chrome DevTools interface is open, showing the **Elements** and **Styles** panels.

**Elements Panel:**

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    ... <div> HELLO </div> == $0
    <script src="./script.js"></script>
    <!-- Code injected by live-server -->
  </body>
</html>
```

**Styles Panel:**

**element.style {**

```
}
div {
  height: 100px;
  padding: 20px;
  border: 5px solid;
  margin: 15px;
  box-sizing: border-box;
  background-color: red;
}
```

**user agent stylesheet**

```
div {
  display: block;
  unicode-bidi: isolate;
}
```

A purple arrow points from the `box-sizing: border-box;` property in the Styles panel to the red box in the browser. Another purple arrow points from the same property to a diagram illustrating the box model.

**Diagram:**

The diagram illustrates the box model for the red box. It shows a central content area (890x100) surrounded by padding (20), a border (5), and a margin (15). The total width is 1000 (890 + 20 + 5 + 15 + 20 + 5 + 15).

Sem border-box,  
default content-box.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

- Static (padrão)
- Relative
- Absolute
- Fixed
- Sticky



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Tipos de position no css

Static (padrão): segue o *fluxo normal*.

- Todo elemento por padrão vem Static.
- Ele respeita o fluxo normal da página: aparece um **embaixo do outro (se for block)** ou **na mesma linha (se for inline)**.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Position static (default)

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
}
```

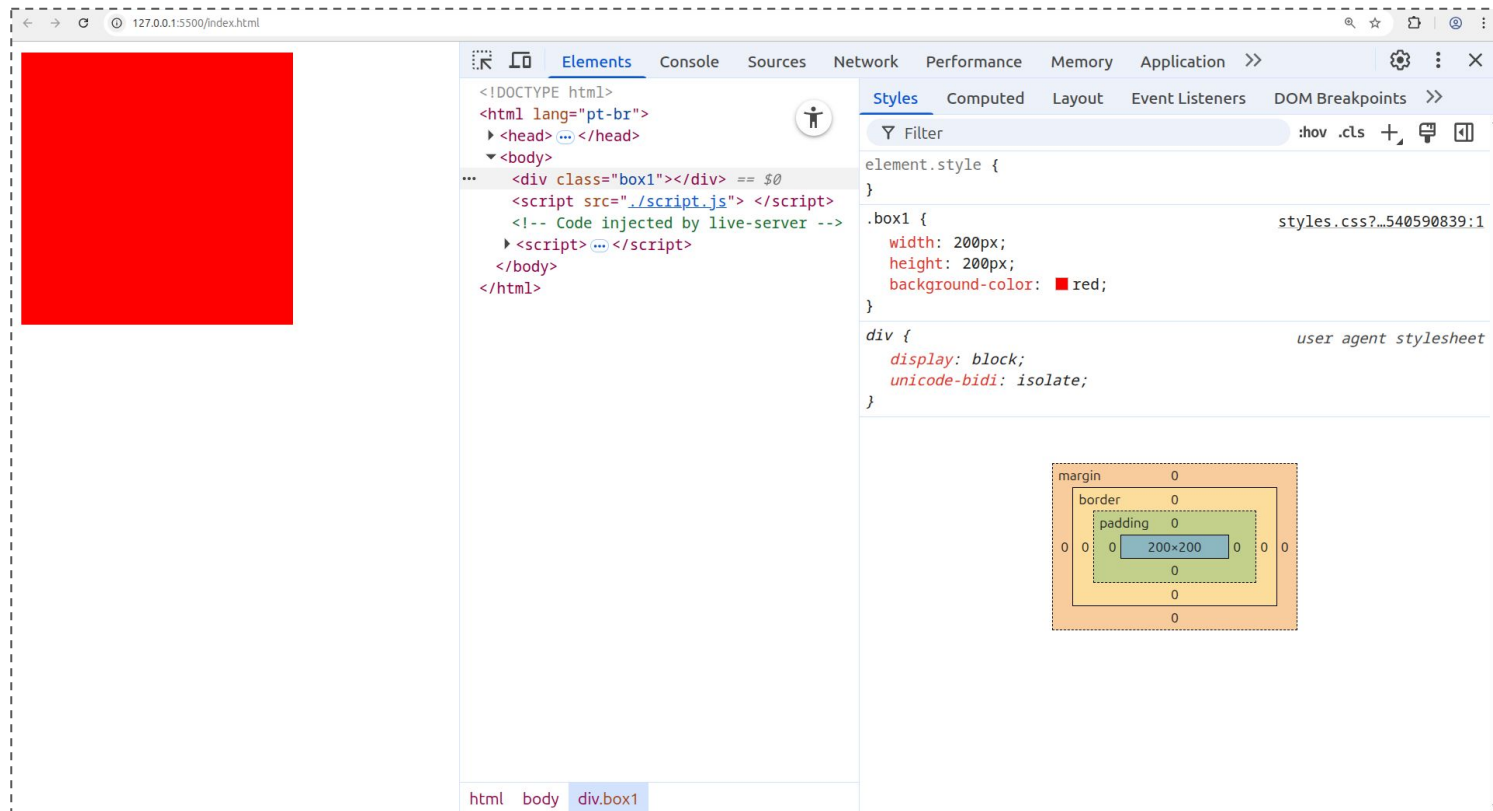
# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Position static (default)

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
  position: static;  
}
```

Ou declarar  
propriedade position:  
**static**

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

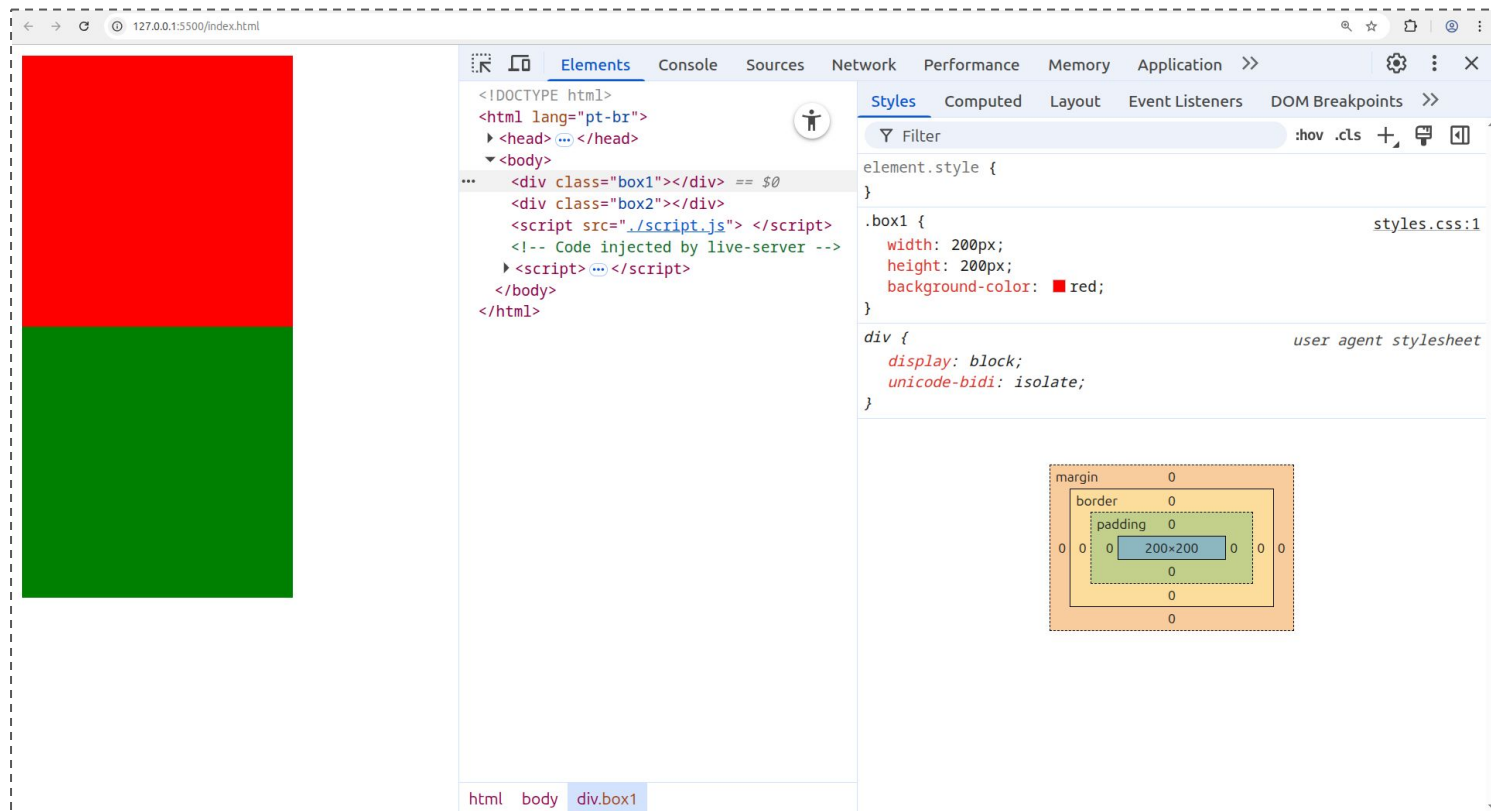


# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

Position static (default)

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
}  
  
.box2 {  
  width: 200px;  
  height: 200px;  
  background-color: green;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

Relative: ajusta posição em relação a si mesmo.

- O elemento continua no fluxo normal, mas pode ser deslocado em relação à sua posição original.
- Usa top, left, right, bottom para “empurrar” o elemento.
- O espaço original dele continua reservado, mesmo que ele seja movido.

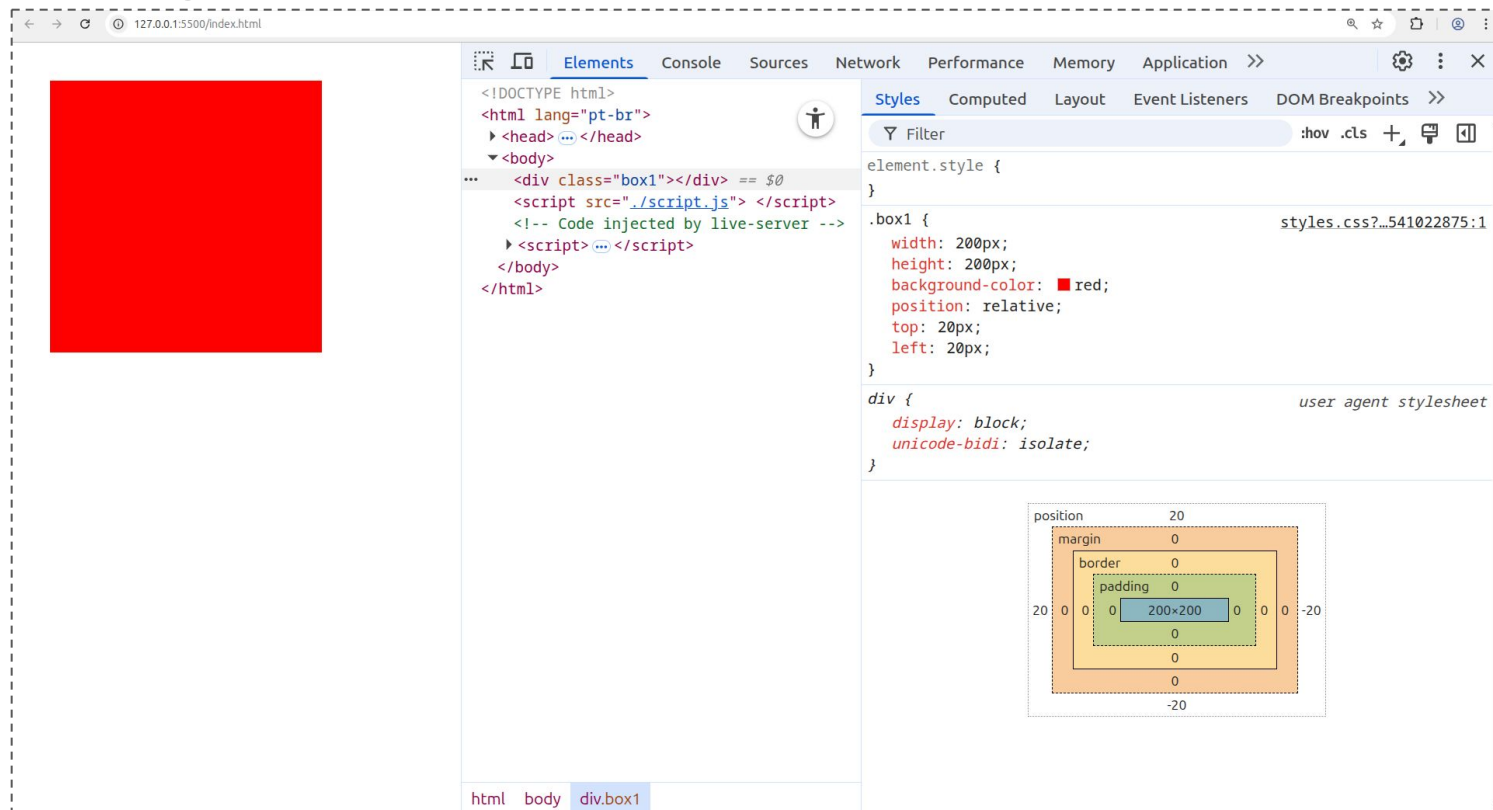
# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Position relative

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
  position: relative;  
  top: 20px;  
  left: 20px;  
}
```



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

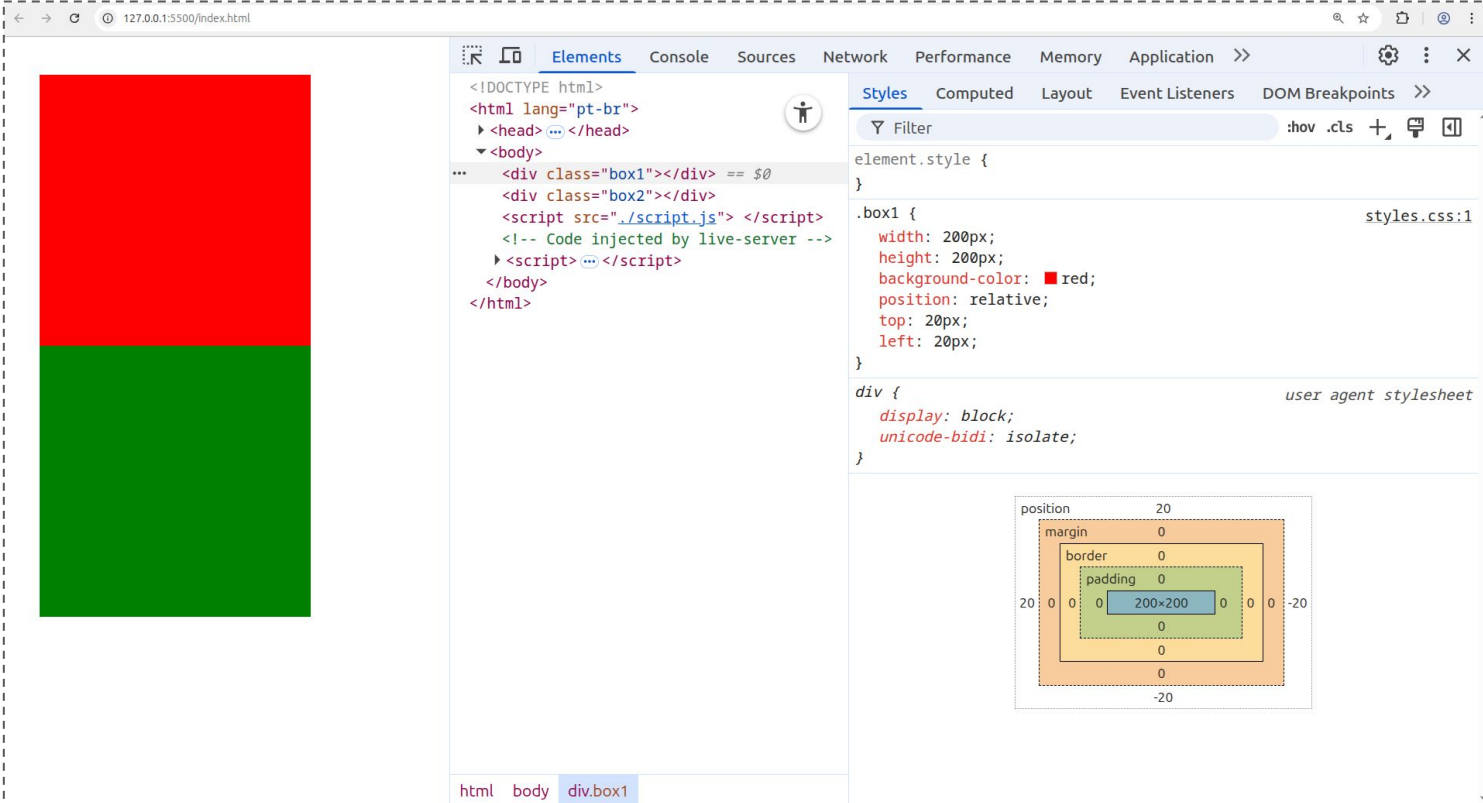


# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Position relative

```
.box2 {  
  width: 200px;  
  height: 200px;  
  background-color: green;  
  position: relative;  
  top: 20px;  
  left: 20px;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



The screenshot displays a web browser window with a live-server interface. The browser shows a red and green box on the left. The right panel is divided into three sections: Elements, Styles, and a diagram.

**Elements Panel:**

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <script src="./script.js"> </script>
  </head>
  <body>
    <div class="box1"></div>
    <div class="box2"></div>
  </body>
</html>
```

**Styles Panel:**

```
element.style {
}

.box1 {
  width: 200px;
  height: 200px;
  background-color: red;
  position: relative;
  top: 20px;
  left: 20px;
}

div {
  display: block;
  unicode-bidi: isolate;
}
```

**Diagram:**

The diagram illustrates the box model for the selected element. It shows a central blue box (200x200) with a green border (padding) and an orange border (margin). The dimensions are labeled as follows:

- margin: 20px (top and bottom), 0px (left and right)
- border: 0px (all sides)
- padding: 0px (all sides)
- Content: 200x200px

The total width and height of the box are 200px. The diagram also shows the box's position relative to the parent container, with a top offset of 20px and a left offset of 20px.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

Absolute: posicionado em relação ao **primeiro ancestral com posição definida**.

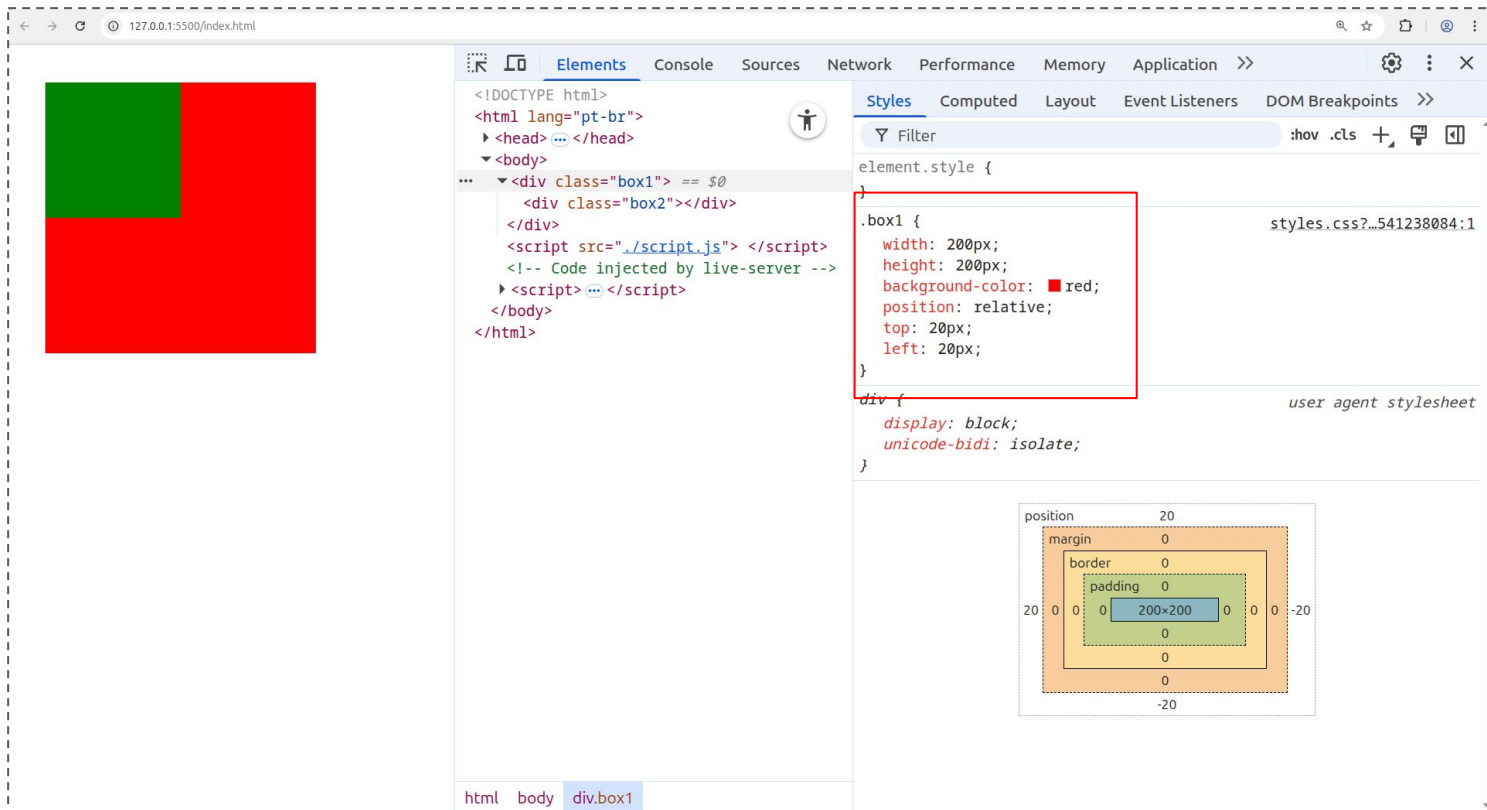
- O elemento sai do fluxo normal (como se fosse “flutuante”).
- Ele se posiciona em relação ao primeiro ancestral com position diferente de static.
- Se não houver, será em relação ao <html> (a tela).

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Tipos de position no css

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
  position: relative;  
  top: 20px;  
  left: 20px;  
}  
  
.box2 {  
  width: 100px;  
  height: 100px;  
  background-color: green;  
  position: absolute;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



The screenshot shows a web browser window with a URL of 127.0.0.1:5500/index.html. The browser's developer tools are open, displaying the HTML structure and the CSS styles for the selected element, `div.box1`.

**HTML Structure:**

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    </head>
  <body>
    <div class="box1">
      <div class="box2"></div>
    </div>
    <script src="./script.js"></script>
    <!-- Code injected by live-server -->
    <script></script>
  </body>
</html>
```

**CSS Styles for `div.box1`:**

```
.box1 {
  width: 200px;
  height: 200px;
  background-color: red;
  position: relative;
  top: 20px;
  left: 20px;
}
```

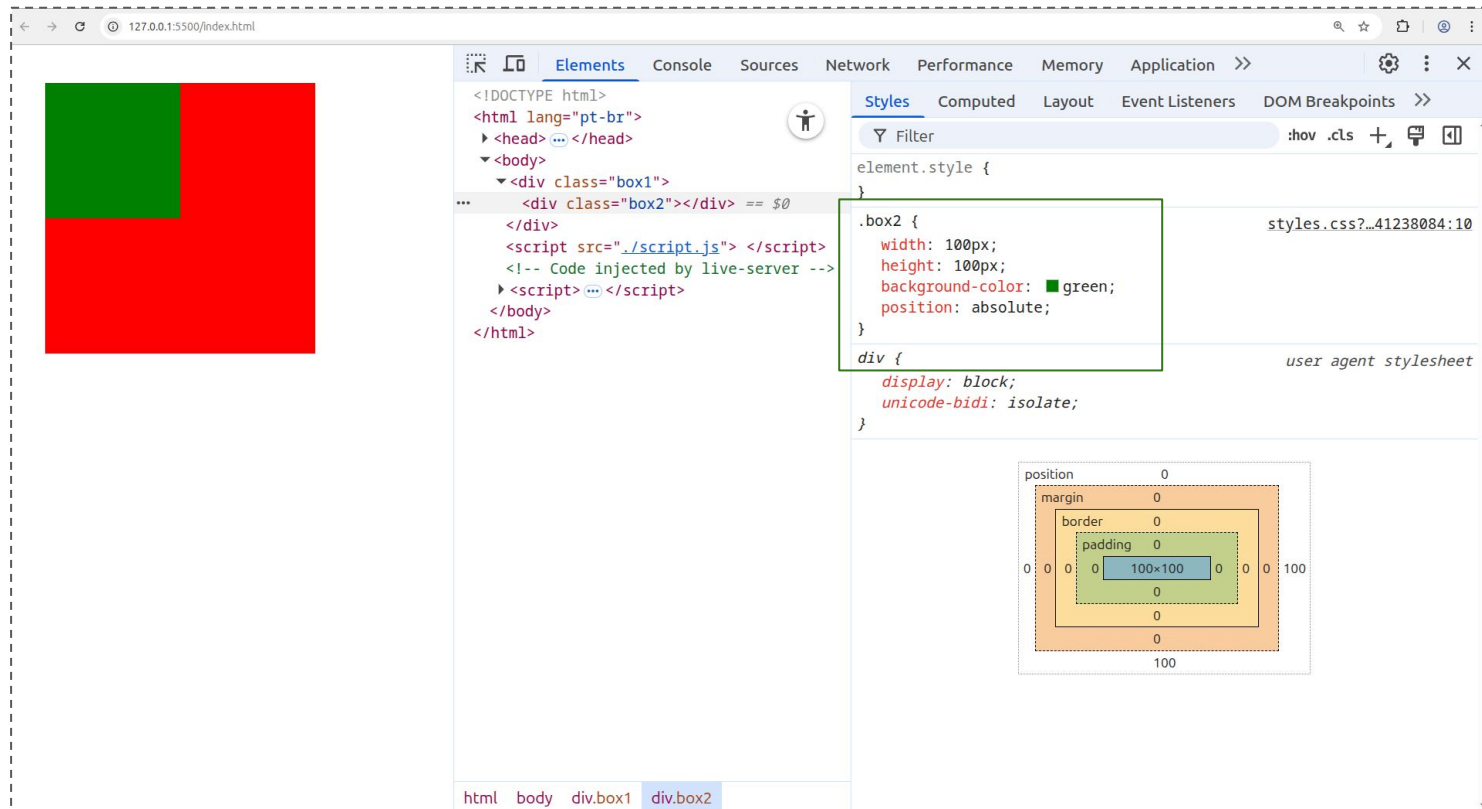
The browser also shows the user agent styles for `div`:

```
div {
  display: block;
  unicode-bidi: isolate;
}
```

**Box Model Diagram:**

The diagram illustrates the box model for the selected element. It shows a central content area of 200x200 pixels, surrounded by padding (0), border (0), and margin (0). The total width and height of the box are 200px. The position is relative, with a top offset of 20px and a left offset of 20px.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



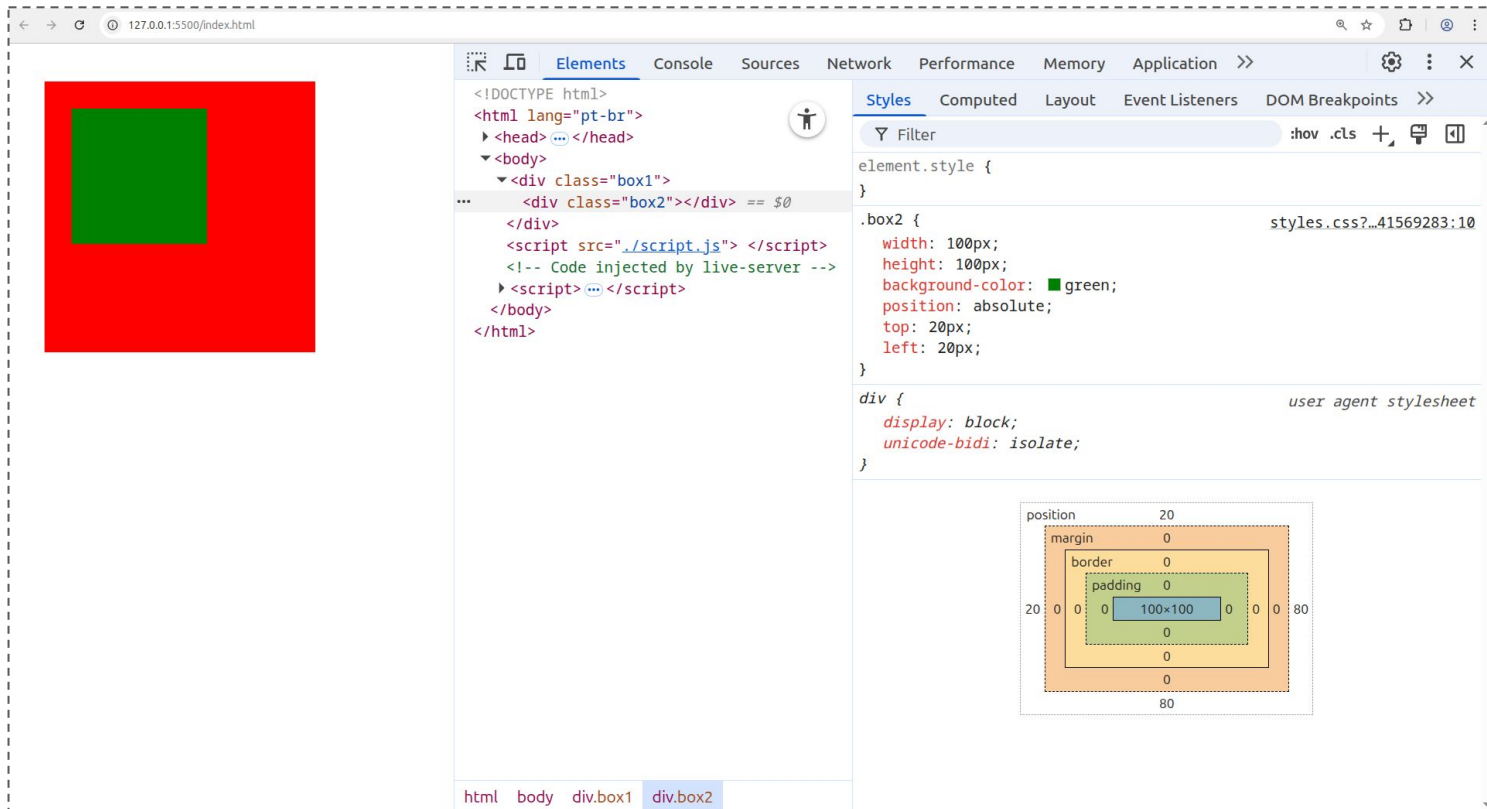
# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Tipos de position no css

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
  position: relative;  
  top: 20px;  
  left: 20px;  
}  
  
.box2 {  
  width: 100px;  
  height: 100px;  
  background-color: green;  
  position: absolute;  
  top: 20px;  
  left: 20px;  
}
```



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



The screenshot shows a web browser with a red square containing a green square. The browser's developer tools are open, displaying the HTML structure and the CSS styles for the green square.

**HTML Structure:**

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <script src="/script.js"> </script>
    <!-- Code injected by live-server -->
  </head>
  <body>
    <div class="box1">
      <div class="box2"></div>
    </div>
  </body>
</html>
```

**CSS Styles:**

```
.box2 {
  width: 100px;
  height: 100px;
  background-color: green;
  position: absolute;
  top: 20px;
  left: 20px;
}

div {
  display: block;
  unicode-bidi: isolate;
}
```

**Box Model Diagram:**

The diagram illustrates the box model for the green square (box2). It shows the following dimensions:

- Content: 100x100 (blue box)
- Padding: 0 (green border)
- Border: 0 (orange border)
- Margin: 0 (yellow border)
- Position: absolute, top: 20px, left: 20px

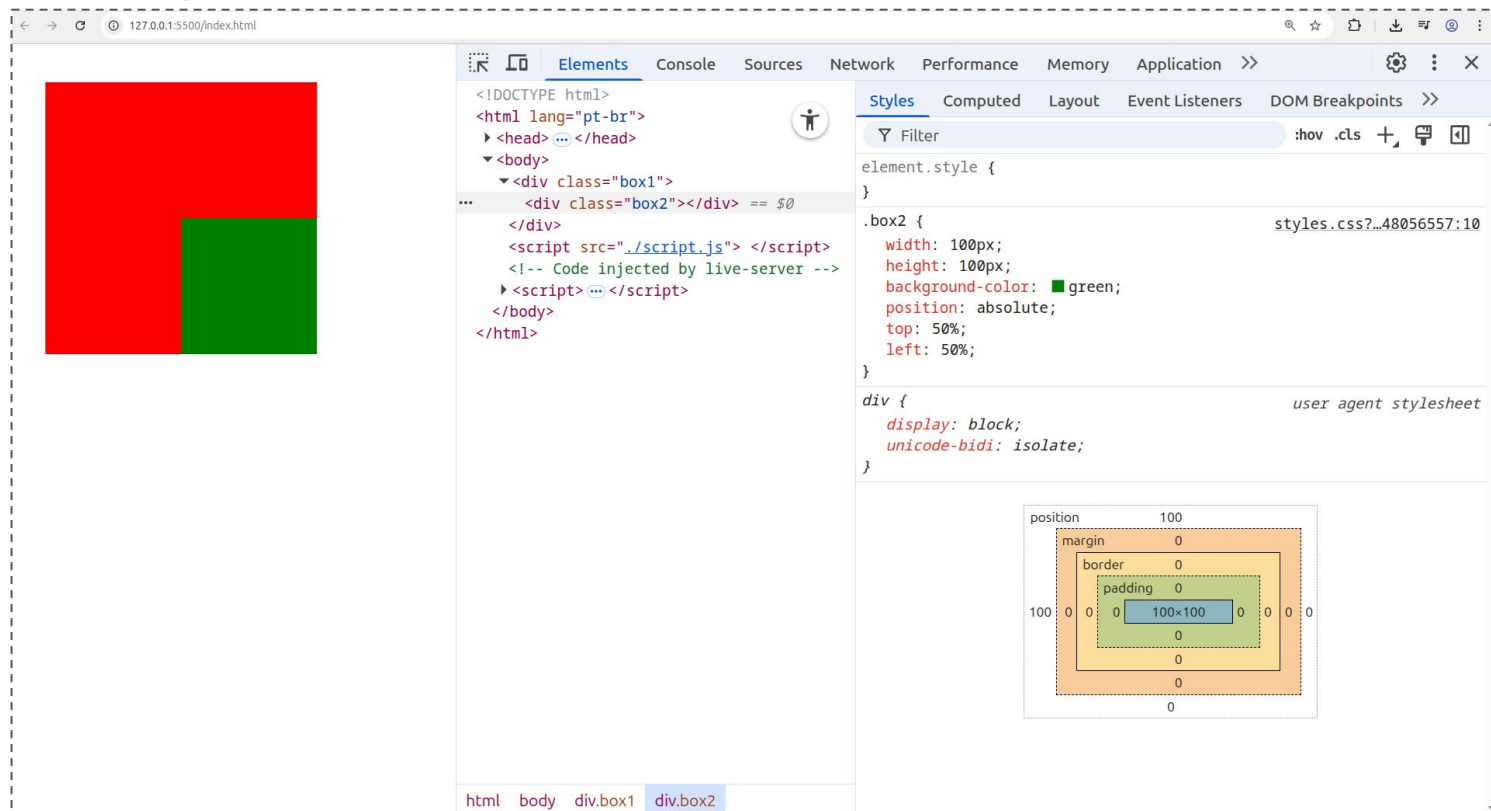
The total width and height of the box are 100px, and the total margin is 0px.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Tipos de position no css

```
.box1 {  
  width: 200px;  
  height: 200px;  
  background-color: red;  
  position: relative;  
  top: 20px;  
  left: 20px;  
}  
  
.box2 {  
  width: 100px;  
  height: 100px;  
  background-color: green;  
  position: absolute;  
  top: 50%;  
  left: 50%;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

Fixed: fixo na tela, mesmo com scroll.

- O elemento também sai do fluxo normal.
- Ele é posicionado em relação à janela do navegador (viewport).
- Mesmo se você rolar a página, ele continua no mesmo lugar.
- Muito usado em menus fixos ou botão “voltar ao topo”.

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

## Tipos de position no css

```
html {  
  padding-bottom: 1000px;  
}  
  
.fixed {  
  height: 50px;  
  width: 200px;  
  background-color: cadetblue;  
  position: fixed;  
  top: 10px;  
  right: 10px;  
}
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

What is Lorem Ipsum? Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it? It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Where can I get some? There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc.

What is Lorem Ipsum? Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it? It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Where can I get some? There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

Sticky: mistura relative e fixed.

- É uma mistura de relative e fixed.
- O elemento age como relative até que a página role a um certo ponto, e então passa a agir como fixed.
- Muito usado em headers que grudam no topo ao rolar.

# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

Tipos de position no css

Sticky: mistura relative e fixed.

```
div.sticky {  
  position: sticky;  
  top: 0;  
  padding: 5px;  
  background-color: #cae8ca;  
  border: 2px solid #4CAF50;  
}
```



# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

```
<h2>Using position: sticky;</h2>
```

```
<p>Try to scroll this page to understand how sticky works.</p>
```

```
<p>Here, when the sticky element reach the top of the page (top: 0), it sticks to this position!</p>
```

```
<div class="sticky">I have position sticky!</div>
```

```
<div style="padding-bottom:2000px">
```

```
  <p>Scroll back up to remove the stickyness!</p>
```

```
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.
```

```
  </p>
```

```
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.
```

```
  </p>
```

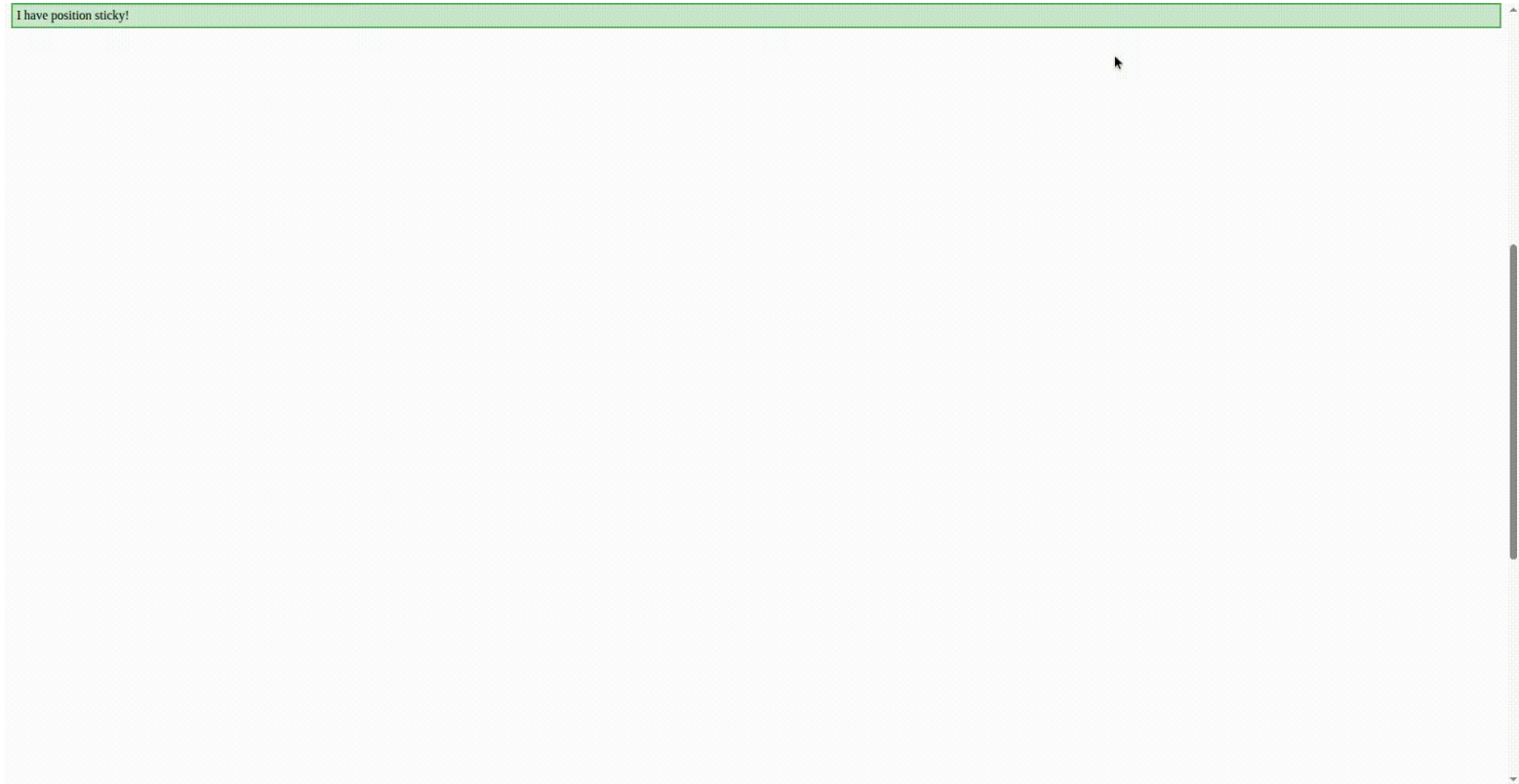
```
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.
```

```
  </p>
```

```
</div>
```

# INTRODUÇÃO A LAYOUT E POSICIONAMENTO

I have position sticky!



# *INTRODUÇÃO A LAYOUT E POSICIONAMENTO*

## Resumo

- Static (padrão): segue o fluxo normal.
- Relative: ajusta posição em relação a si mesmo.
- Absolute: posicionado em relação ao primeiro ancestral com posição definida.
- Fixed: fixo na tela, mesmo com scroll.
- Sticky: mistura relative e fixed.



***MÃO NA MASSA***

# MÃO NA MASSA



1. Crie um parágrafo em que uma palavra apareça destacada em cor diferente na mesma linha do texto.

Exemplo

Lorem Ipsum is **simply dummy text** of the printing and typesetting industry.

# MÃO NA MASSA



2. Crie três blocos coloridos empilhados, cada um ocupando **toda a largura da tela**.

Exemplo



## MÃO NA MASSA



3. Crie três blocos coloridos empilhados, cada um ocupando **toda a largura da tela**. Insira um bloco dentro do primeiro bloco ao final. Exemplo



## MÃO NA MASSA



4. Crie três blocos coloridos empilhados, cada um ocupando **toda a largura da tela**. Insira um bloco dentro do primeiro bloco ao final. Adiciona mais um bloco dentro do bloco criado no primeiro bloco. Exemplo





## MÃO NA MASSA



5. Desenvolva uma página web simples que use position fixed para colocar uma âncora estilizada como botão com o ícone do whatsapp. Ao clicar na âncora, sua página deverá fazer um redirecionamento para a página do seu whatsapp web. Coloque conteúdo de texto suficiente em seu html para que o scroll seja habilitado automaticamente. Sua âncora deverá ficar visível independente da rolagem do scroll.



- 6. Desenvolva uma página web simples que use position stick para criar uma header flutuante.
- 7. Desenvolva uma página web simples que use position stick para criar uma footer flutuante.

# REFERÊNCIAS

- [https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)
- <https://developer.mozilla.org/pt-BR/docs/Web/CSS/position>
- <https://www.devmedia.com.br/como-usar-a-propriedade-position-css/24451>