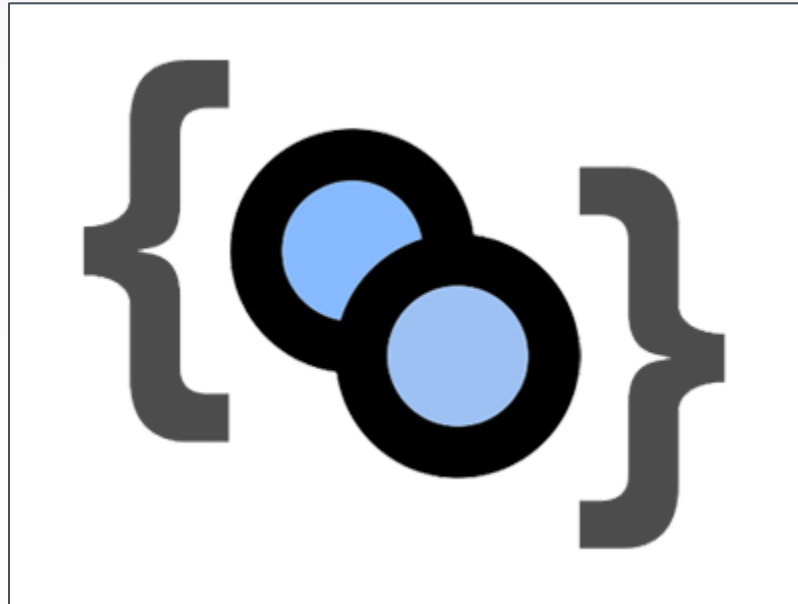


Desenvolvimento para Web

Módulo Básico





POSICIONAMENTO CSS

- **Posicionamento**

- As **CSS** preveem várias funcionalidades destinadas a **manipular** e **definir** o posicionamento dos **boxes** da **marcação HTML**
- **Distribuir boxes** na página significa **criar o layout** da página (**definir** como os **diferentes boxes** se **distribuem visualmente** na **tela do usuário**)
- Por padrão, os **elementos HTML** **nível** de **bloco** distribuem-se **verticalmente** um após o outro na **ordem** em que **aparecem na marcação HTML** e os **elementos inline** posicionam-se em **linha** na **horizontal**

- **Posicionamento**

- *Os mecanismos de posicionamento CSS permitem ao desenvolvedor **alterar** o **comportamento** padrão, **não** só **alterando** a **ordem** como também **posicionando elementos nível** de **bloco** um ao lado do outro*
- *Essas alterações são feitas com regras CSS definindo valores para as propriedades CSS destinadas a **manipular** a **posição** dos **boxes** da página*



- **Posicionamento**

- *Estilizamos o **topo do site do Geraldo** e **visualizamos** sua renderização no **topo** da tela do navegador*
- *É perfeitamente **factível**, com **uso** de **algumas declarações CSS**, fazer com que o **topo do site** seja **renderizado** no **final** da tela, **alterar a navegação** de **horizontal** para **vertical** e **posicioná-la** no **lado direito***



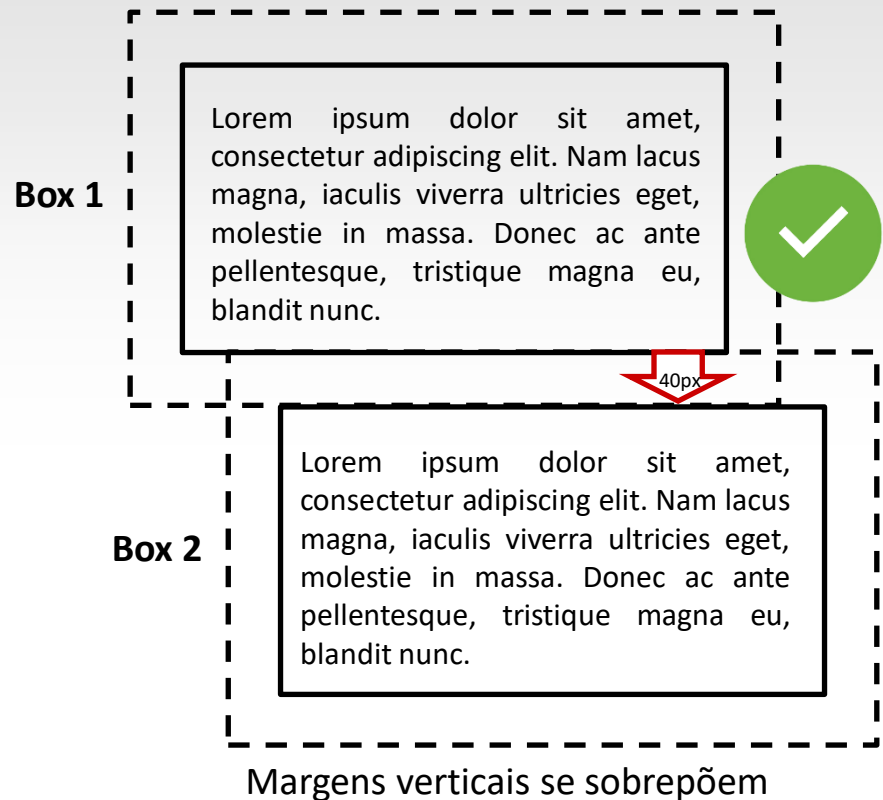
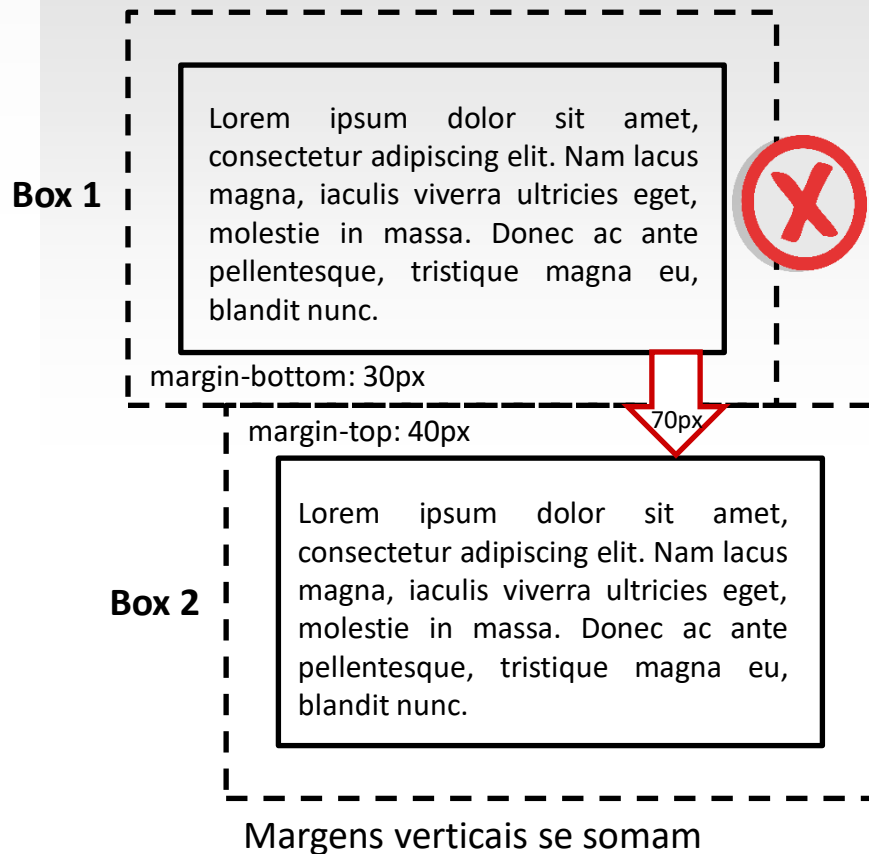
- **Esquemas de Posicionamento**

- As CSS preveem **três esquemas** de posicionamento: **normal flow** (fluxo normal), **float** (flutuado) e **absolute** (absoluto)

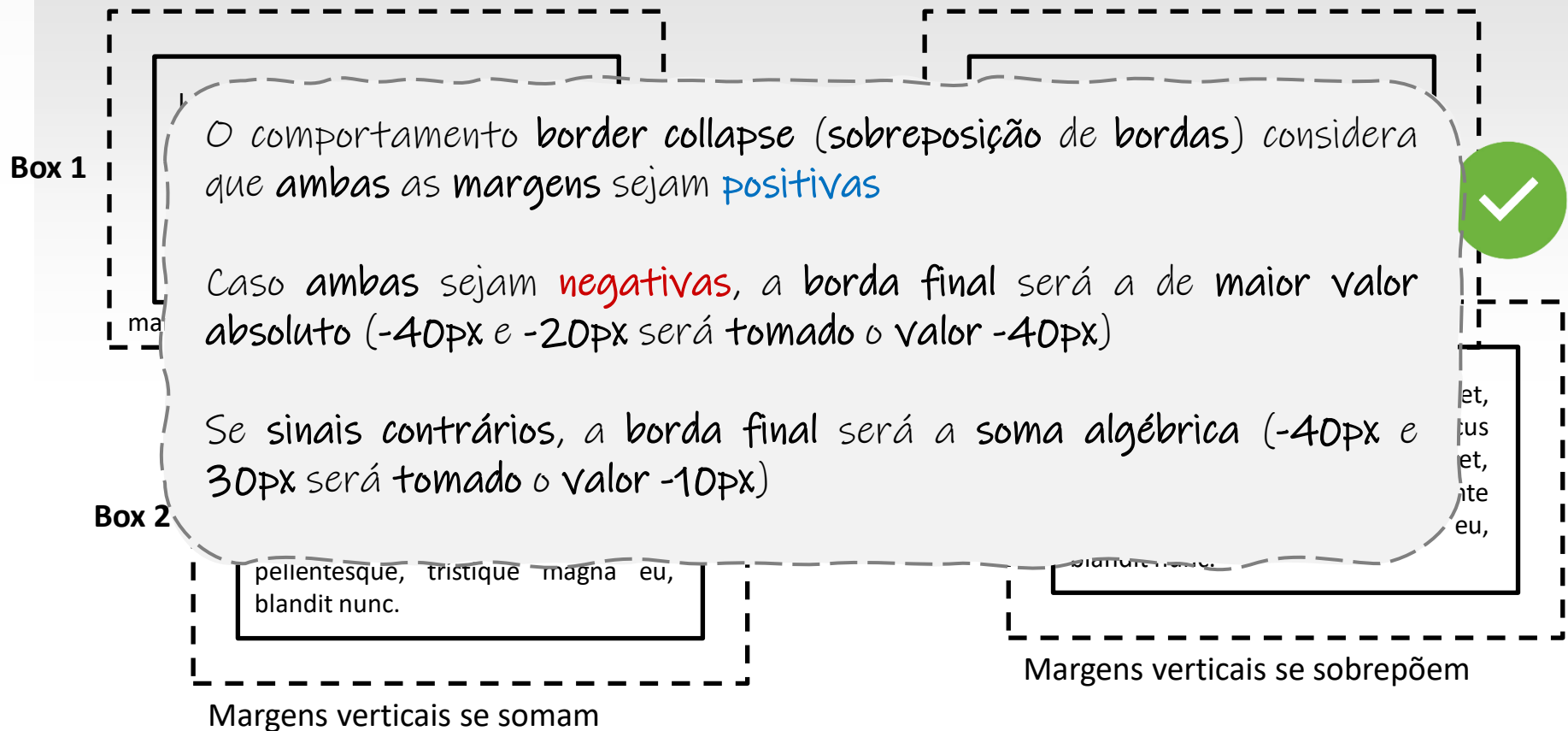
- **Fluxo Normal:** é o posicionamento padrão dos boxes, na vertical ou horizontal
 - **Flutuado:** quando o box continua no fluxo normal e é posicionado à esquerda ou à direita, como se estivesse “**flutuando**” no conteúdo (texto) do box seguinte, que se desenvolve ao lado do box flutuado
 - **Absoluto:** quando o box é retirado do fluxo normal e posicionado segundo um sistema de coordenadas

- **Esquema Normal (Padrão)**
 - **Elementos Nível de Bloco**
 - A **distância vertical** entre elementos **nível** de **bloco** que se seguem na **marcação HTML** (**fluxo do documento**) é **determinada** pela **propriedade margin**
 - **Não** tendo sido **definida** por **regras** de **estilo** uma **margem vertical** para o **bloco**, será **tomada** a **margem inicial padrão**, própria da **folha de estilo nativa** do navegador
 - **Margens verticais** entre **blocos** que se seguem sempre se **sobrepõem** (**collapse**)

- **Esquema Normal (Padrão)**
 - **Elementos Nível de Bloco**



- **Esquema Normal (Padrão)**
 - **Elementos Nível de Bloco**



- **Esquema Normal (Padrão)**
 - **Elementos inline**
 - A **formatação dá-se em linha** (*horizontal*), e dentro do **bloco** que contém o **box inline**
 - Elementos **inline** admitem somente **margens horizontais** (*margin-left* e *margin-right*) – **margens verticais** (*margin-bottom* e *margin-top*) são **ignoradas**



- **Esquema Normal (Padrão)**
 - **Elementos inline**
 - Na *marcação a seguir* identificamos os seguintes boxes: **um bloco container** formado pelo elemento parágrafo **p** e **três boxes inline**
 - "**Este parágrafo contém**" e "**elementos inline**", os quais são denominados **boxes inline anônimos**, e "**três**" que é um **box inline contido no elemento inline strong**

`<p> Este parágrafo contém três elementos inline. </p>`

- ***Esquema Normal (Padrão)***
 - ***Elementos inline***
 - ***Considere o parágrafo anterior inserido entre outros dois parágrafos e com a classe **especial** nele declarada, conforme a marcação HTML***

`<p> Este é o primeiro parágrafo. </p>`

`<p class="especial"> Este parágrafo contém três elementos inline. </p>`

`<p> Este é o terceiro parágrafo. </p>`

- **Esquema Normal (Padrão)**
 - **Elementos inline**
 - Aplicaremos uma **margem** no **box inline strong** do **segundo parágrafo** e uma **borda** em **todos** os **parágrafos** para **facilitar a visualização**

```
p { border: 1px solid black; }  
.especial > strong {  
    margin-top: 3800px;      /* será ignorada */  
    margin-bottom: 5400px;  /* será ignorada */  
    margin-left: 150px;     /* será aplicada */  
    margin-right: 280px;    /* será aplicada */  
}
```

- ***Esquema Normal (Padrão)***
 - ***Elementos inline***

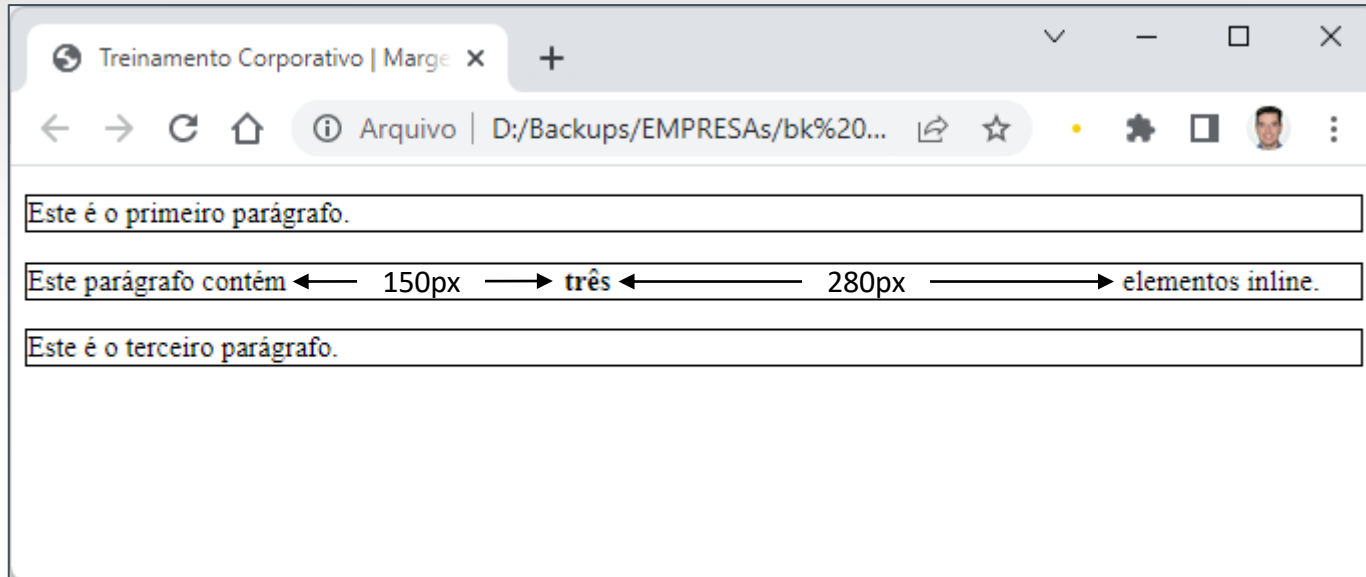


Figura 1 – Margens em boxes inline

- **Esquema Relativo**

- O esquema de **posicionamento relativo** é regido pela propriedade CSS **position** e seu **valor relative**
- Esse **posicionamento não** retira o elemento posicionado do **fluxo do documento**, **preservando o espaço** que ele **ocupava antes de ser posicionado**



- **Esquema Relativo**

- **Posicionamento Relativo**

- A declaração CSS **position: relative** sozinha **não** causa nenhum efeito no posicionamento de um box
 - Quando usada em conjunto com as propriedades **left**, **top**, **right** e **bottom**, movimenta o **bloco** da sua **posição inicial** a uma **distância definida** pelos **valores declarados** nessas **propriedades**
 - As **propriedades left** e **right** definem o **quanto** o **bloco** deve ser **deslocado** para a **direita** ou à **esquerda**
 - As **propriedades top** e **bottom** definem o **deslocamento** para **baixo** e para **cima**

- **Esquema Relativo**

- **Posicionamento Relativo**

- Para exemplificar o **posicionamento relativo**, considere **três div** (**div.um**, **div.dois** e **div.tres**) em **sequência** na **marcação HTML** e **uma margem** entre eles
 - Para o **segundo div** (**div.dois**) foi **declarado** um **posicionamento relativo** com **definição** das **propriedades left** e **top** e **margem superior** e **inferior**

```
<div class="um"> Este é o elemento DIV 1 </div>  
<div class="dois"> Este é o elemento DIV 2 </div>  
<div class="tres"> Este é o elemento DIV 3 </div>
```

- ***Esquema Relativo***
 - ***Posicionamento Relativo***

```
div { width: 170px;  
      height: 40px;  
      border: 1px solid black; }
```

```
div.dois { position: relative;  
           left: 60px;  
           top: 15px;  
           margin: 20px 0; }
```

CSS

- ***Esquema Relativo***
 - ***Posicionamento Relativo***

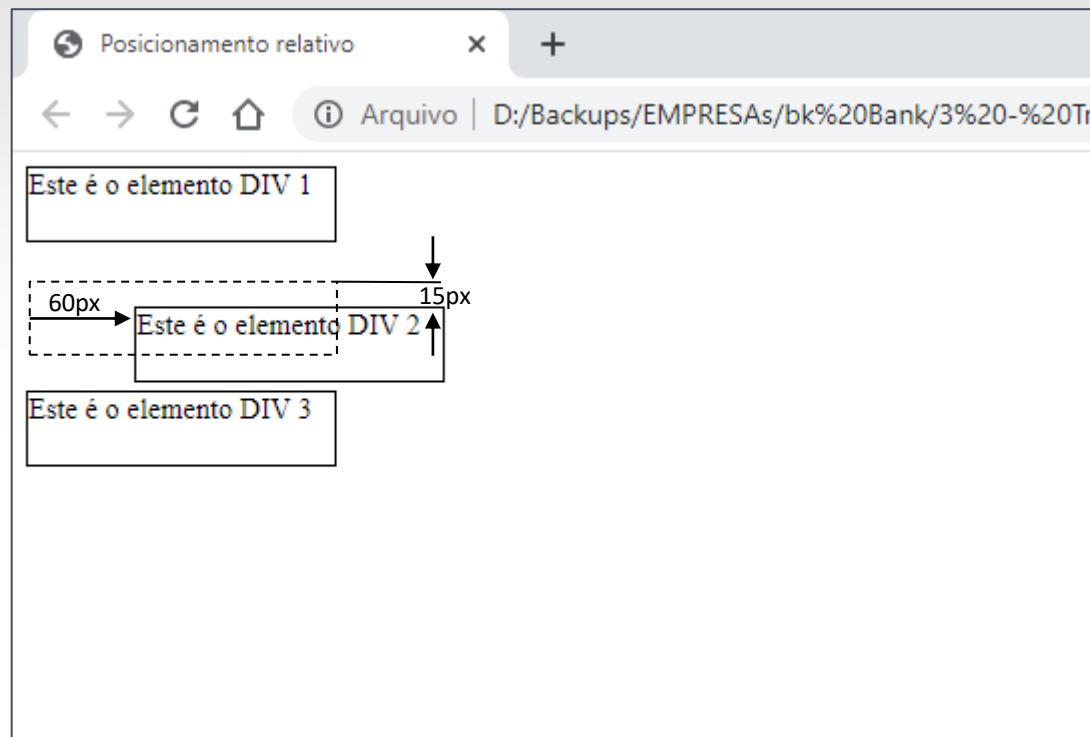


Figura 2 – Posicionamento relativo

- **Esquema Relativo**
 - **Posicionamento Relativo**

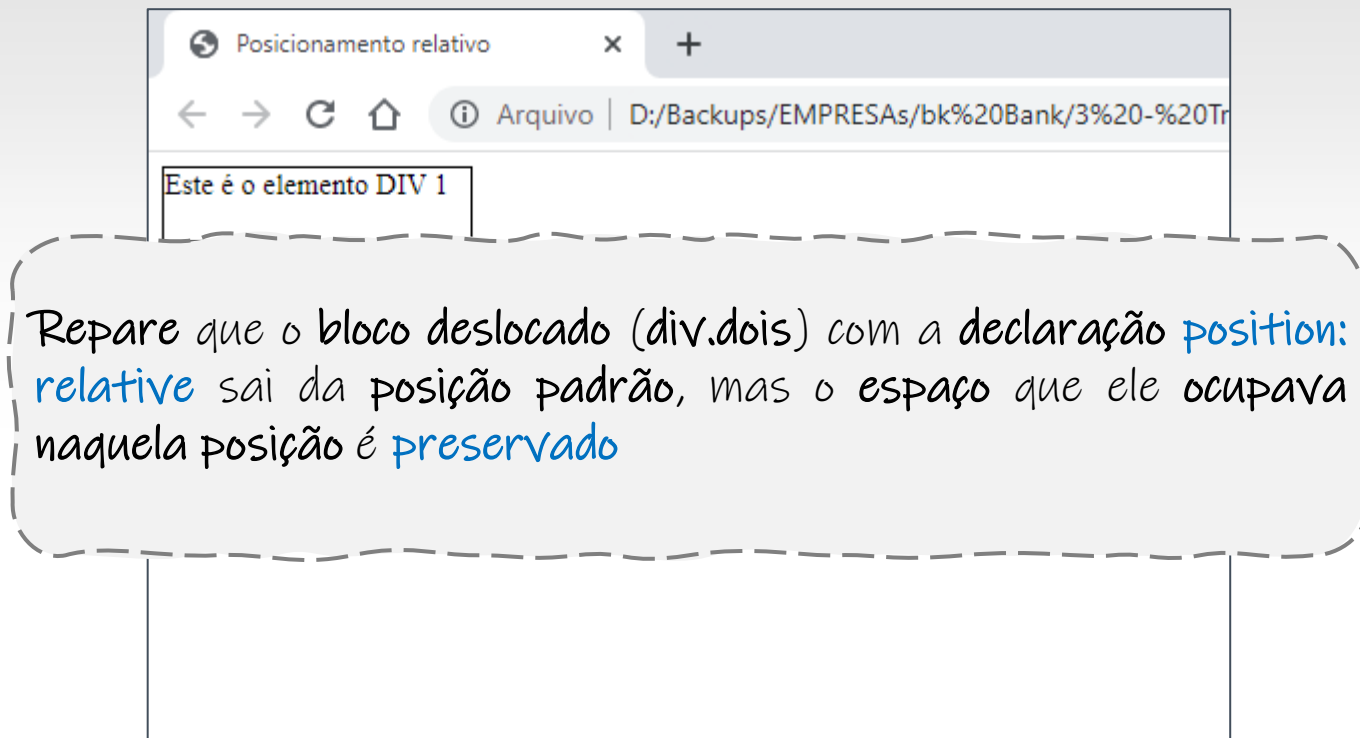


Figura 2 – Posicionamento relativo

- **Esquema Relativo**

- **Posicionamento Relativo**

- Elementos **inline** também podem ser **deslocados** da sua **posição padrão** com o uso **dessa declaração**

HTML

```
<h2> Neste cabeçalho <span> estas palavras </span>  
foram deslocadas de sua posição padrão. </h2>
```

CSS

```
span {  
  position: relative;  
  left: 60px;  
  top: 40px;  
  border: 1px solid black;  
}
```

- **Esquema Relativo**
 - **Posicionamento Relativo**

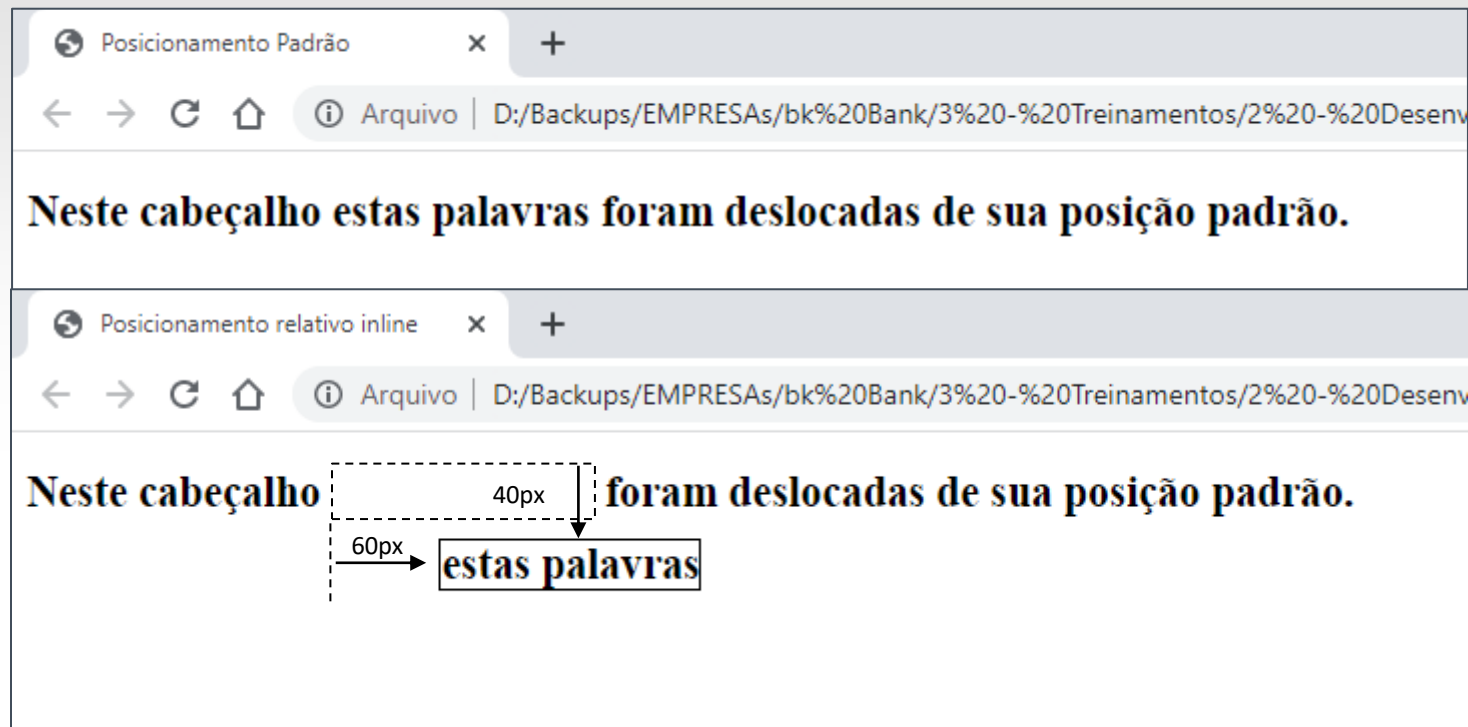


Figura 3 – Posicionamento relativo inline

- **Esquema Relativo**
 - **Posicionamento Relativo**

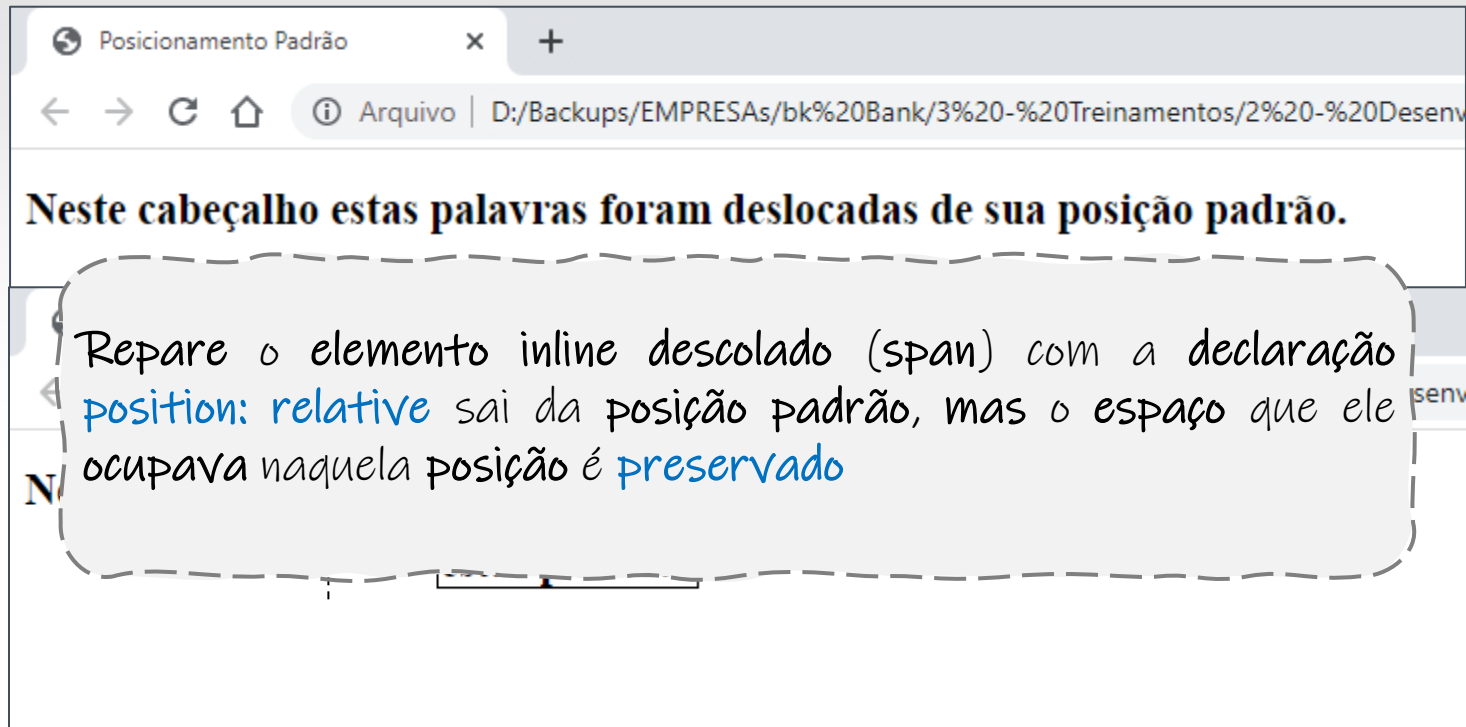


Figura 3 – Posicionamento relativo inline

- **Esquema com Float**

- **Posicionamento** com **float** ou **posicionamento flutuado** é definido pela propriedade CSS **float** e seus valores **left**, **right**, **none** e **inherit** (**herdado**)
- Outra propriedade CSS usada nesse esquema de posicionamento é a propriedade **clear** e seus valores **none**, **left**, **right**, **both** e **inherit**
- O **box** é retirado de sua posição no fluxo do documento e flutuado para a **direita** ou para a **esquerda**

- ***Esquema com Float***

- ***Ao contrário do **posicionamento relativo**, nesse esquema de posicionamento, o **espaço original ocupado pelo box não** será **deixado livre**, mas **ocupado pelo elemento que se segue no fluxo do documento*****



- *Esquema com Float*
 - *Flutuando elementos nível de bloco*
 - Considere **vários blocos dentro** de um mesmo **elemento container**
 - O **primeiro bloco flutuado** desloca-se lateralmente, para a esquerda (valor: **left**) ou para a direita (valor: **right**), até tocar a borda lateral (**esquerda** ou **direita**) e superior do **elemento container**
 - O **segundo bloco flutuado** desloca-se igualmente até tocar a borda do primeiro, e assim sucessivamente, enquanto houver espaço **horizontal** no **elemento container** (faltando espaço, o próximo bloco ocupará uma linha abaixo, e assim por diante)

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - Considere a **marcação HTML** para seis **elementos div** e um **container** e as **regras de estilo**

#HTML

```
<div class="container">  
  <div class="um"> DIV 1 </div>  
  <div class="dois"> DIV 2 </div>  
  <div class="tres"> DIV 3 </div>  
  <div class="quatro"> DIV 4 </div>  
  <div class="cinco"> DIV 5 </div>  
  <div class="seis"> DIV 6 </div>  
</div>
```

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - Considere a **marcação HTML** para seis **elementos div** e um **container** e as **regras de estilo**

CSS

```
.container {  
    border: 1px solid black;  
    width: 300px;  
    height: 280px;  
}
```

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - Considere a **marcação HTML** para seis **elementos div** e um **container** e as **regras de estilo**

CSS

```
div { border: 2px dotted black; }  
.container > div:nth-child(2n) { background: #ccc; }  
.um { width: 100px; height: 50px; }  
.dois { width: 65px; height: 40px; }  
.tres { width: 50px; height: 45px; }  
.quatro { width: 115px; height: 60px; }  
.cinco { width: 45px; height: 20px; }  
.seis { width: 50px; height: 30px; }
```

- ***Esquema com Float***
 - ***Flutuando elementos nível de bloco***

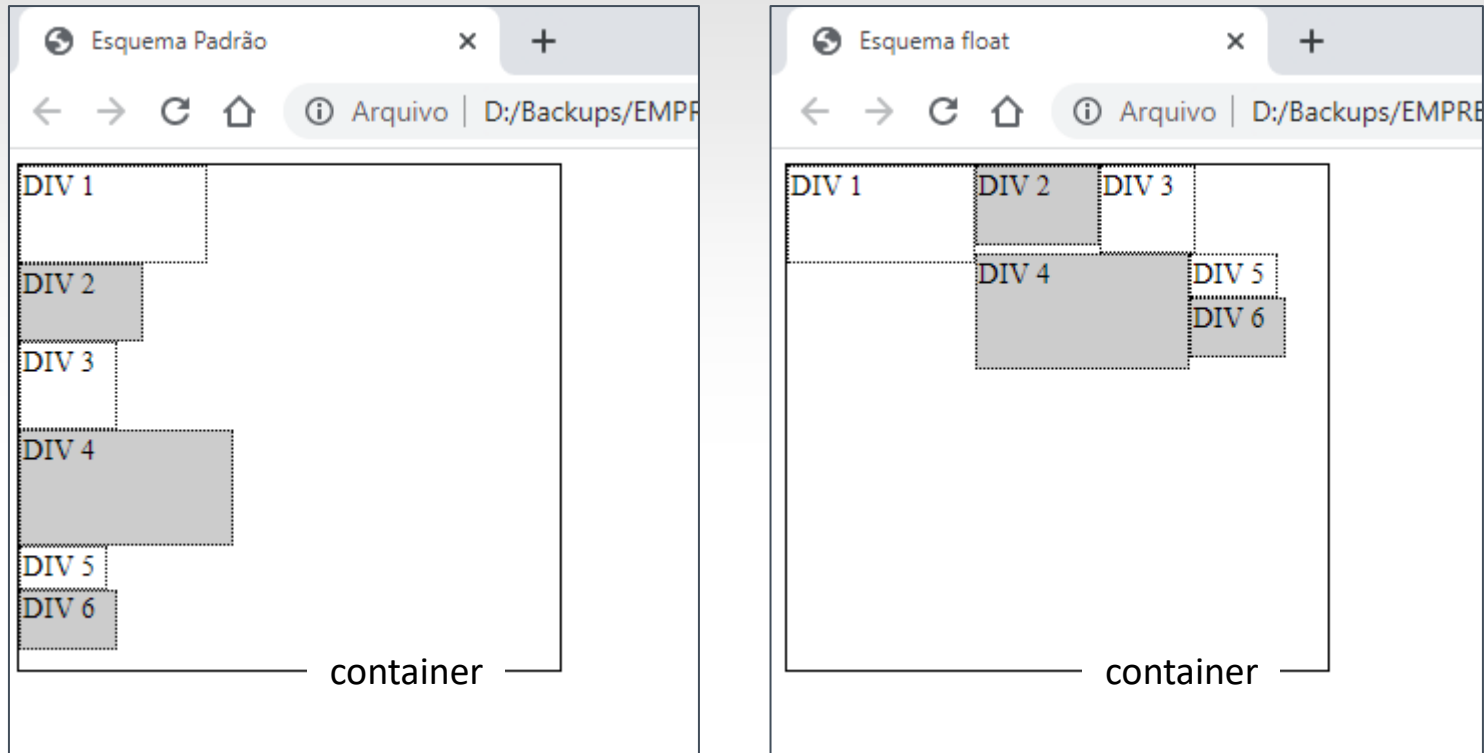


Figura 4 – Posicionamento float

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**

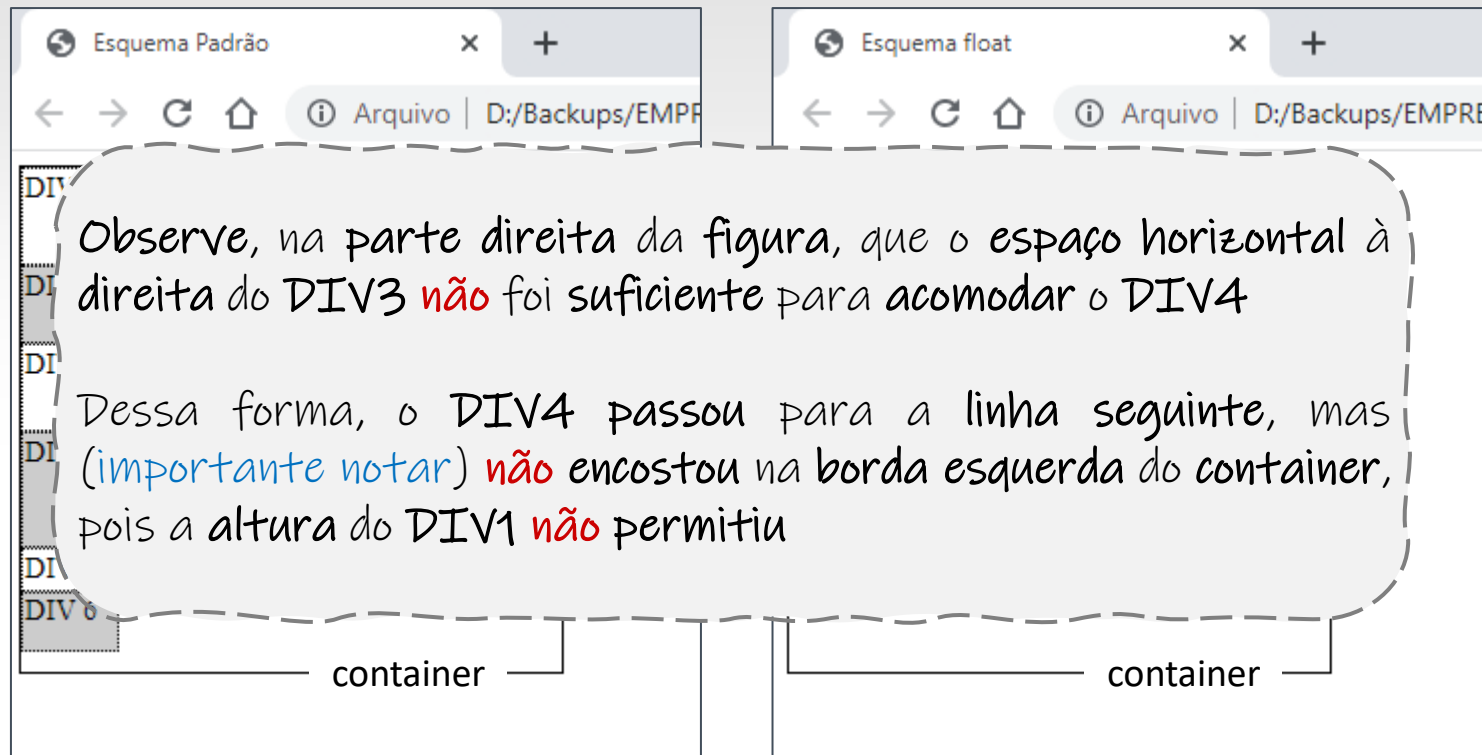


Figura 4 – Posicionamento float

- **Esquema com Float**

- **Flutuando elementos nível de bloco**

- Uma **aplicação prática** de uso desse tipo de posicionamento para uma sequência de blocos é na construção de uma **barra** de **navegação**
 - **Barras de navegação**, em geral, são marcadas com o elemento HTML **lista**, nas quais cada **item** é um **link**

```
<ul>
  <li><a href="home.html">Home</a></li>
  <li><a href="quemsomos.html">Quem Somos</a></li>
  <li><a href="portfolio.html">Portfólio</a></li>
  <li><a href="contato.html">Contato</a></li>
</ul>
```


- ***Esquema com Float***
 - ***Flutuando elementos nível de bloco***
 - Em uma **lista não ordenada**, o elemento **ul** é o **container** para os **elementos li**, que são os **blocos a serem flutuados**
 - Se **definirmos** uma **largura** para o elemento **ul** que seja **suficiente** para **acomodar a soma das larguras dos elementos li** e **flutuarmos à esquerda tais elementos**, **obteremos uma lista na horizontal**

- *Esquema com Float*
 - *Flutuando elementos nível de bloco*
 - A declaração **clear** com seus valores: **none**, **left**, **right**, **both** e **inherit** tem a finalidade de “**limpar**” o que está abaixo de boxes flutuados
 - Se a flutuação for à esquerda, use **clear: left**; se à direita, **clear: right**
 - Ou faça como a maioria dos desenvolvedores e use sempre **clear: both** que se aplica a **ambos** os casos de flutuação e funcionará se no futuro você resolver inverter o esquema de flutuação

- ***Esquema com Float***
 - ***Flutuando elementos nível de bloco***

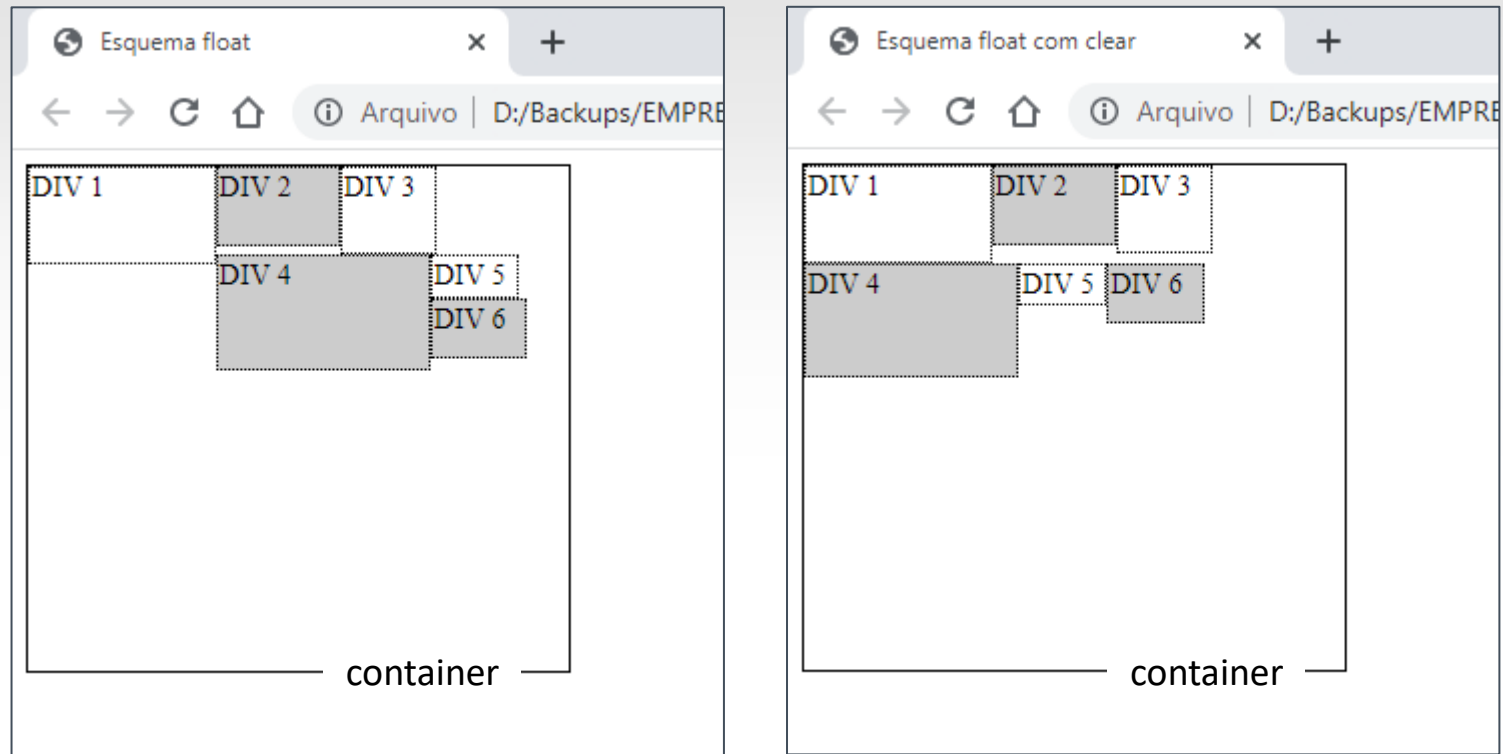


Figura 5 – Posicionamento float com clear

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**

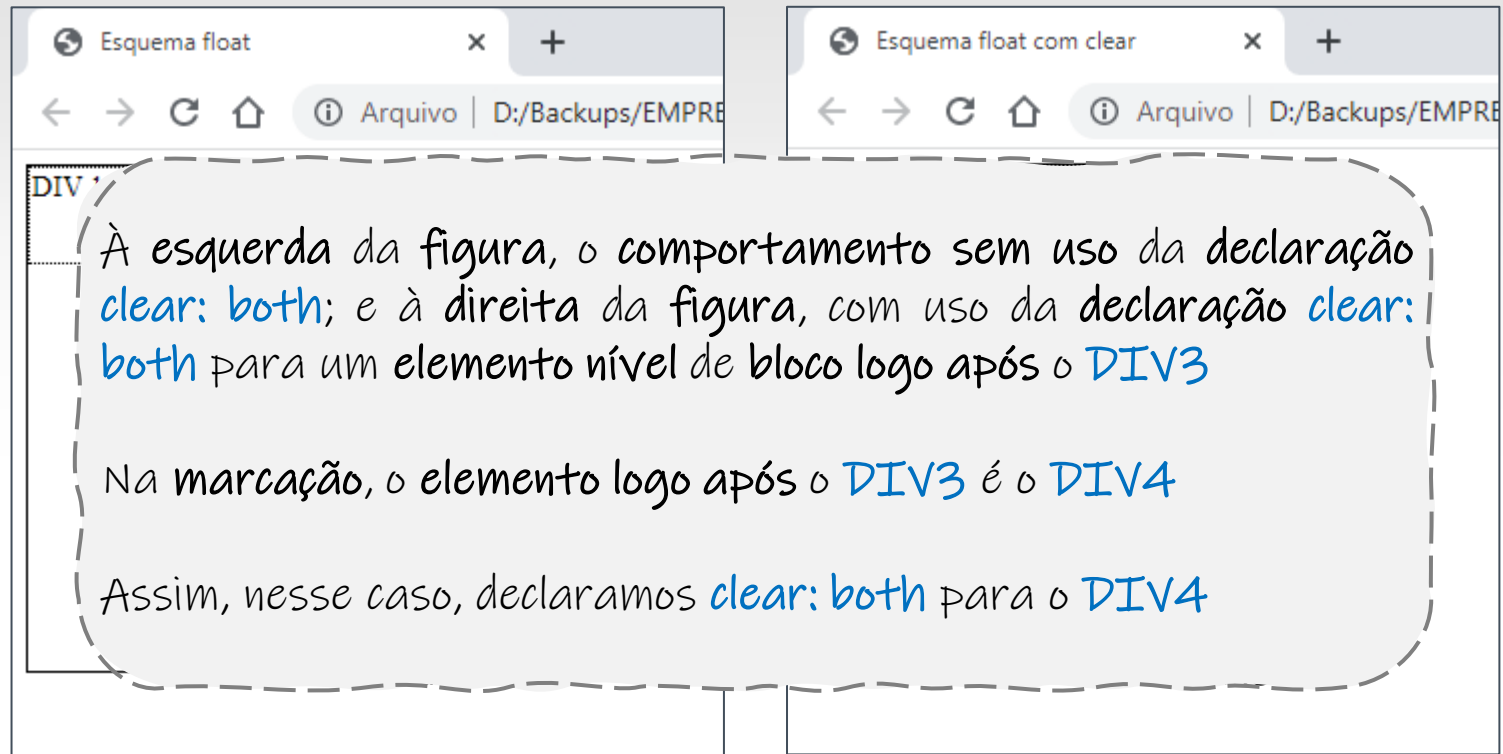


Figura 5 – Posicionamento float com clear

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - Outro comportamento que deve ser considerado quando usamos o **posicionamento flutuado** diz respeito ao **elemento container** dos **boxes flutuados**
 - Se **não** for especificada uma **altura**, o **container** de um ou mais elementos, por padrão, expande-se para poder conter os elementos dentro dele
 - Uma vez que **elementos flutuados** são retirados do **fluxo normal** do documento, tudo se passa como se eles deixassem de forçar a expansão da altura do container, que, se **não** tiver sido especificado (**altura**), torna-se zero

- **Esquema com Float**

- **Flutuando elementos nível de bloco**

- Nos **dois exemplos apresentados**, repare que a **soma total** das **alturas** dos **seis elementos div** é igual a $50\text{px} + 40\text{px} + 45\text{px} + 60\text{px} + 20\text{px} + 30\text{px} = 245\text{px}$ e que foi **declarada** uma **altura igual a 280px** para o **container**
 - Dessa forma, **há uma folga de 35px embaixo** do **container** como se pode **observar** na parte **esquerda** das **figuras** dos **exemplos**
 - Repare ainda que, **após a flutuação** dos **elementos div**, o **container manteve-se** com sua **altura declarada igual a 280px**

- ***Esquema com Float***
 - ***Flutuando elementos nível de bloco***

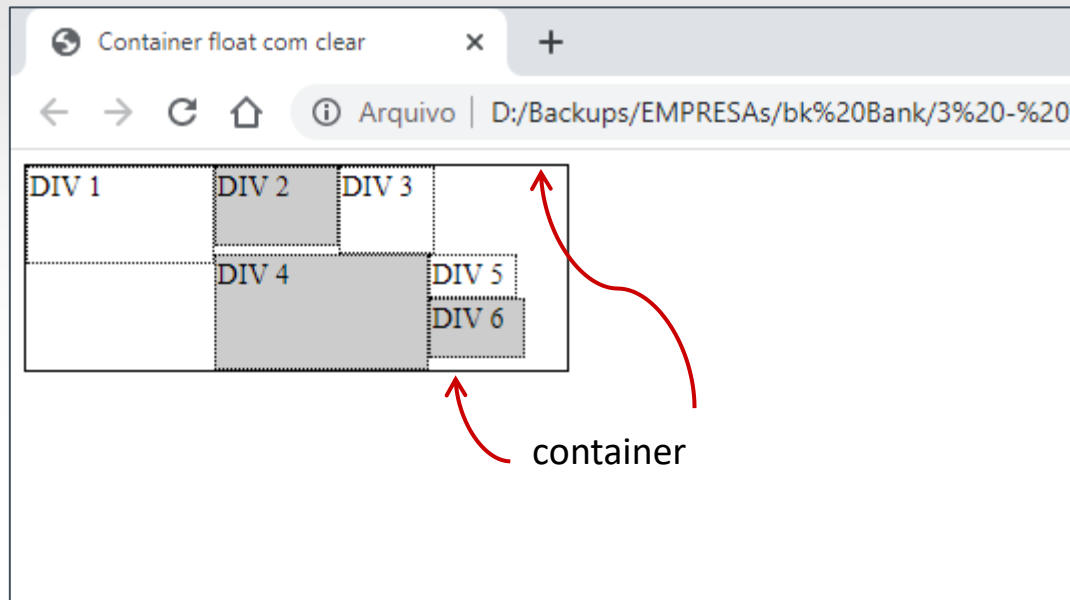


Figura 6 – Altura do container de boxes flutuados

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**

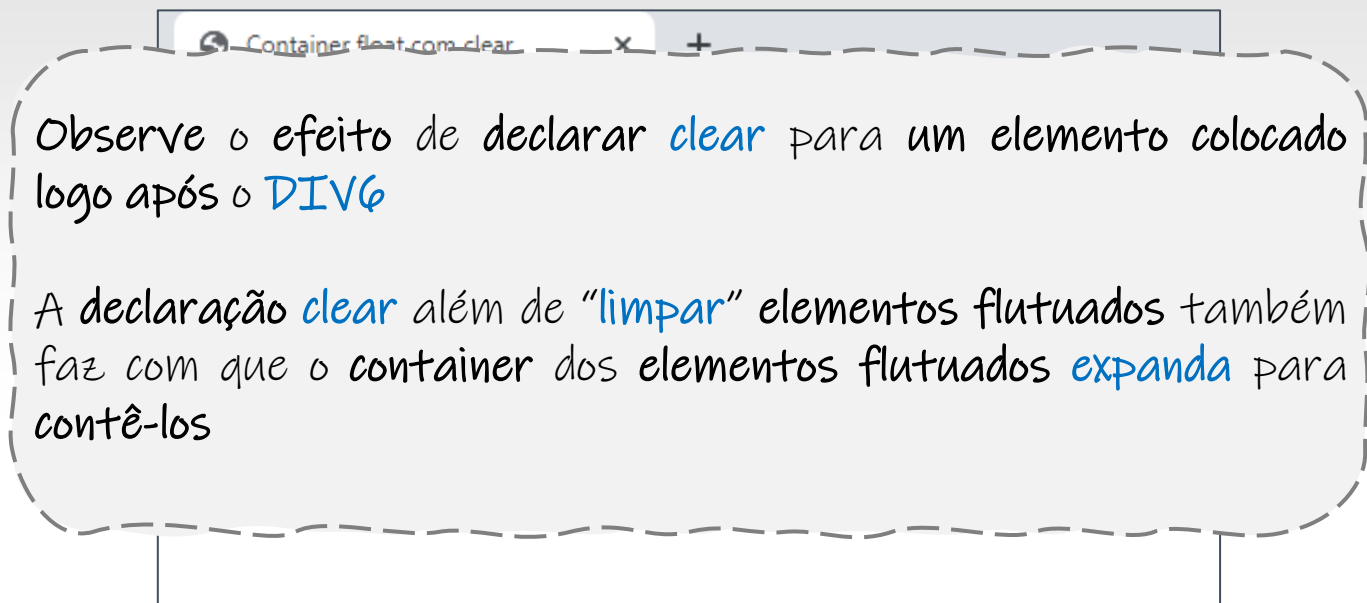


Figura 6 – Altura do container de boxes flutuados

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - A técnica para “**limpar**” elementos flutuados produz **dois efeitos**:
 - **(1)** Restabelece o fluxo normal dos elementos, isto é, cada elemento nível de bloco será renderizado na sequência em que se encontra na marcação
 - **(2)** Faz com que o container dos elementos flutuados se expanda para contê-los
 - **Regra Geral**: toda vez que se usa posicionamento com **flutuação** deve-se “**limpar**” os elementos flutuados

- **Esquema com Float**
 - **Flutuando elementos nível de bloco**
 - A técnica mais antiga para “**limpar**” elementos flutuados **consiste** em **marcar um elemento vazio** (**sem conteúdos**) logo após a marcação dos elementos flutuados e declarar para eles **clear: both**
 - Crie um **div vazio** e atribua a ele a classe **clear**
 - Nas **CSS** declare **clear: both** para a classe **clear**

```
<div class="clear"></div>
```

HTML

```
.clear { clear: both; }
```

CSS

- ***Esquema com Float***
 - ***Flutuando elementos inline***
 - ***Como exemplo consideraremos o caso de uma imagem dentro de um parágrafo***
 - O **elemento container** é o parágrafo, e o **elemento inline** a flutuar é a imagem
 - **Considere dois parágrafos no fluxo do documento e a imagem no primeiro parágrafo**

```
<p>
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
```

```
<p> Suspendisse potenti. ...</p>
```

```
<p> Morbi varius, ...</p>
```

- ***Esquema com Float***
 - ***Flutuando elementos inline***

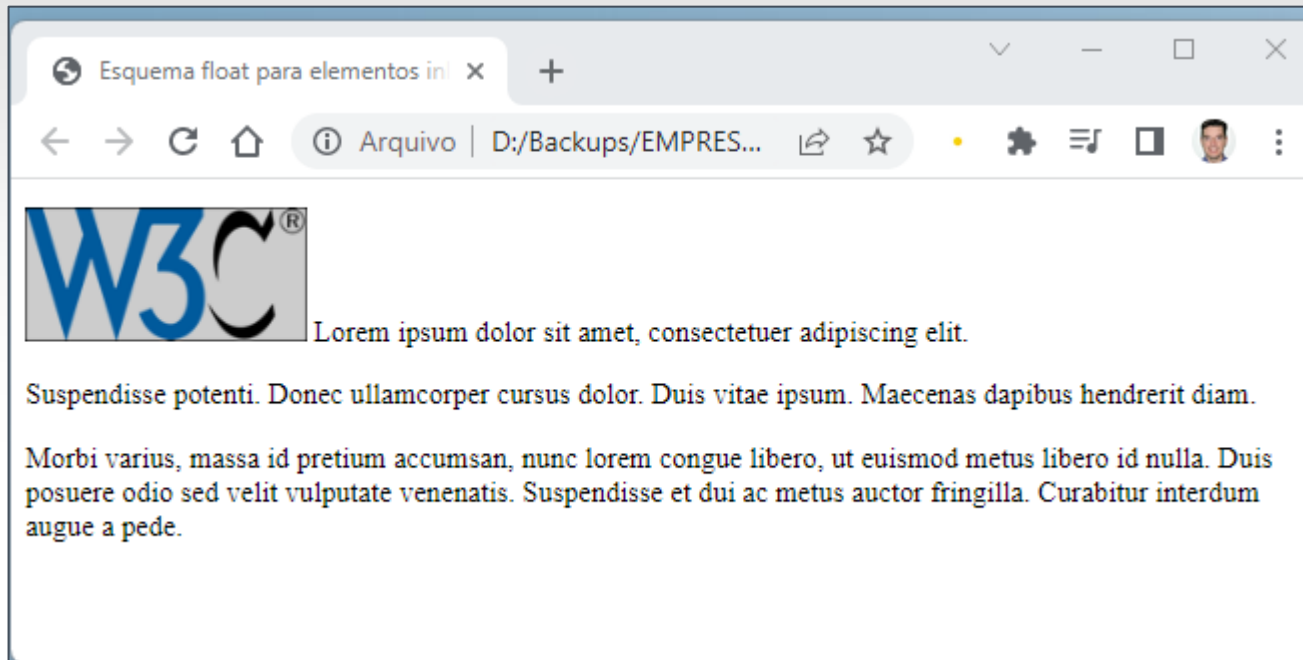


Figura 7 – Imagem inline em parágrafo

- ***Esquema com Float***
 - ***Flutuando elementos inline***
 - ***Para flutuar a imagem à esquerda ou à direita, aplicamos a seguinte regra CSS***

```
img { float: left; }
```

ou

```
img { float: right; }
```

- ***Esquema com Float***
 - ***Flutuando elementos inline***

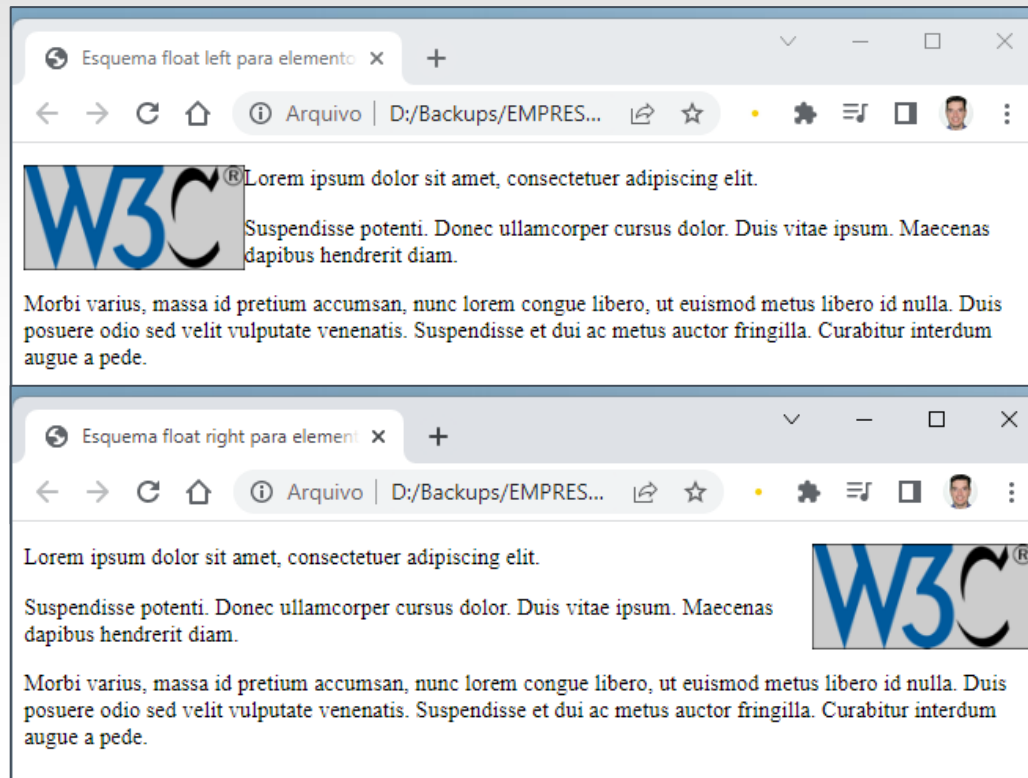


Figura 8 – Imagem float left e right

- ***Esquema com Float***
 - ***Flutuando elementos inline***

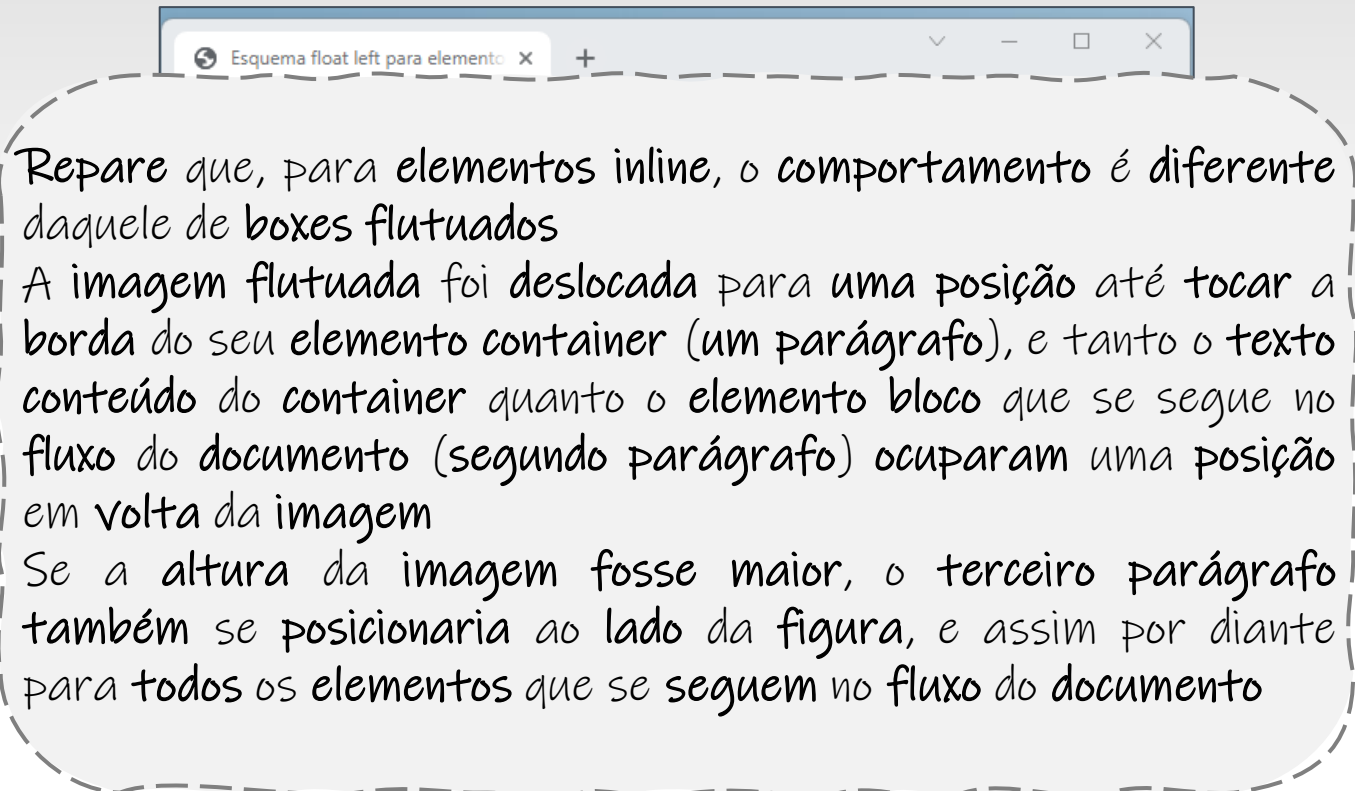


Figura 8 – Imagem float left e right

- **Esquema com Float**

- **Flutuando elementos inline**

- **Elementos flutuados são retirados do *fluxo normal* do documento, e foi por essa razão que o *segundo parágrafo* “*subiu*” e se *posicionou ao lado da imagem***
 - **Sabemos também que para *evitar* a “*subida*” do *segundo parágrafo*, basta “*limpar*” o *elemento flutuado acrescentando a seguinte marcação***

```
<p>
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
```

```
<div class="clear"></div>
```

```
<p> Suspendisse potenti. ...</p>
```

```
<p> Morbi varius, ...</p>
```

```
.clear { clear: both; }
```

HTML

- ***Esquema com Float***
 - ***Flutuando elementos inline***

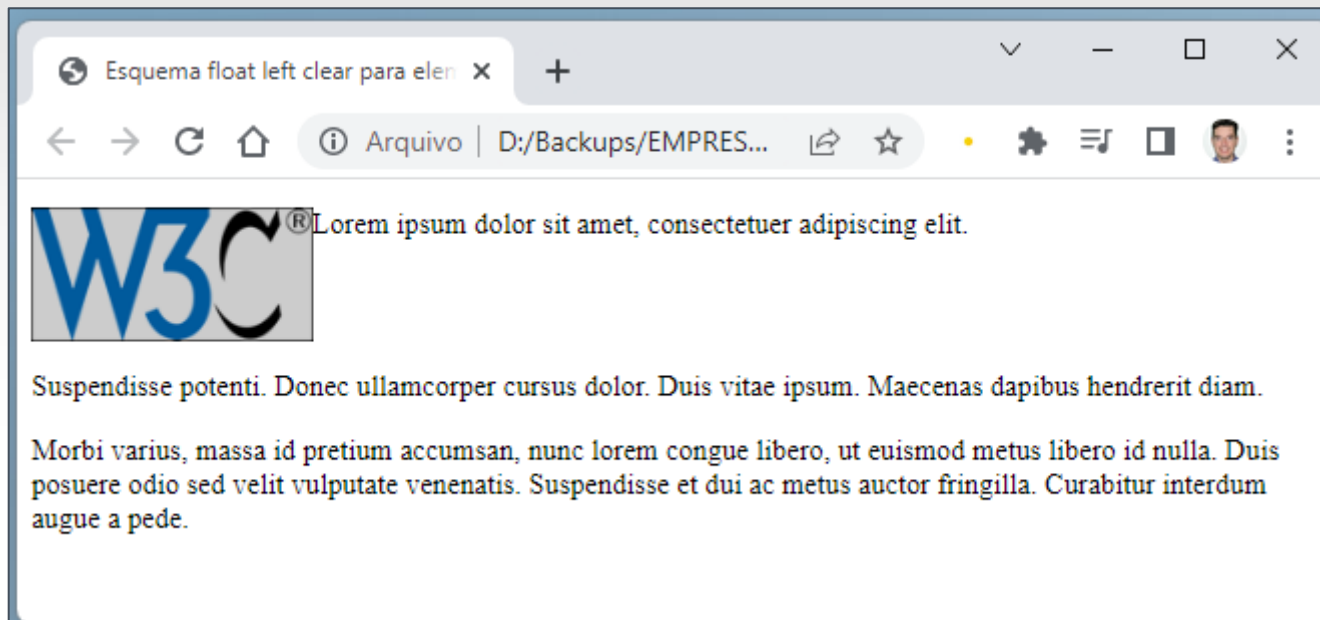


Figura 9 – Imagem float left com clear

- **Esquema Absoluto**

- É regido pela propriedade CSS **position** e seus **valores absolute** e **fixed**
- Ao contrário do posicionamento com o **valor relative**, esse posicionamento retira o elemento posicionado do fluxo do documento, fazendo com que o elemento que se segue ocupe seu lugar



- **Esquema Absoluto**

- **Posicionamento Absoluto**

- A declaração CSS **position: absolute;** sozinha **não** causa qualquer efeito no posicionamento do box, mas faz com que o **elemento seguinte** na marcação se **desloque** para a **posição** que ele ocupava antes de ser posicionado
 - Um **box deslocado** pode ou **não** ser **obscurecido** pelo **box** com o qual **compartilha a mesma posição** (**z-index/posicionamento de boxes sobrepostos**)

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- A declaração **position: absolute**; quando usada em conjunto com as propriedades **left**, **top**, **right** e **bottom**, movimenta o bloco da sua **posição inicial** de uma **distância** definida pelos valores declarados em tais propriedades
 - Medidas de comprimento CSS são empregadas para definir as **coordenadas** do **deslocamento**
 - As propriedades **left** e **right**: definem o quanto o bloco deve ser **deslocado** para a **direita** ou à **esquerda**, respectivamente

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- As propriedades **top** e **bottom**: definem o quanto o bloco deve ser deslocado para **baixo** e para **cima**, respectivamente
 - Entretanto, ao **contrário** do que ocorre com o **posicionamento relativo**, no qual a referência para o deslocamento (**origem** das **coordenadas**) é o próprio box a ser posicionado, quando se trata de **posicionamento absoluto**, a referência **não** é sempre a mesma e depende do que chamamos de **contexto** de **posicionamento**

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- **Regra (Contexto de Posicionamento):**

- Ao **posicionar** um **box** de forma **absoluta**, o **contexto** de **posicionamento** (origem das coordenadas) é o **elemento ancestral** mais próximo para o qual se tenha declarado a propriedade **position** com **valor** **relative**, **absolute** ou **fixed**
 - Se **não** houver **elemento ancestral** posicionado, o **contexto** de **posicionamento** é a **viewport** (janela do navegador), e, nesse caso, chamamos de **contexto de padrão**

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- Para **exemplificar os diferentes casos de posicionamento absoluto**, usaremos a **marcação HTML**:

```
<p>Lorem ipsum dolor sit amet ... </p>  
<div class="container">  
  <p>Morbi varius, massa ... </p>  
  <p class="tres">Suspendisse potenti. Donec ...</p>  
  <p>Curabitur hendrerit ...</p>  
</div>
```

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- Na **CSS** definimos **bordas** para os **elementos da marcação** e **uma cor de fundo** para **fins de clareza nas figuras**, **apresentando os posicionamentos**

```
div { width: 360px;  
      padding: 10px 20px;  
      border: 2px solid black; }  
p {   border: 2px dotted black;  
      padding: 10px 20px; }  
.tres { background: #ddd; }
```


- ***Esquema Absoluto***
 - ***Posicionamento Absoluto***

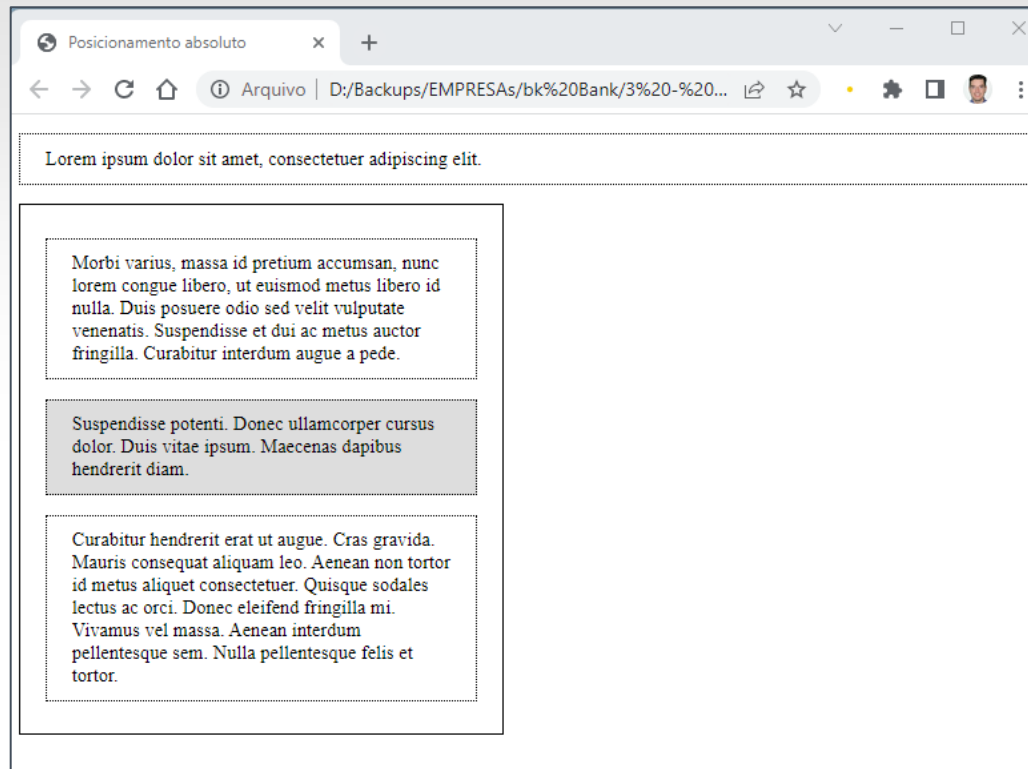


Figura 10 – Renderização Padrão

- **Esquema Absoluto**
 - **Posicionamento Absoluto**
 - Declarar **posicionamento absoluto** para o parágrafo **p.tres** acrescentando na *folha de estilo*

```
.tres {  
    position: absolute;  
    background: #ddd;  
}
```

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- **Parágrafo p.tres**
A simples declaração de `position: absolute;` sem definir as coordenadas do deslocamento, fez com que o **parágrafo três** permanecesse no mesmo lugar, liberando espaço que ocupava e causando a subida do parágrafo quatro que segue no fluxo do documento

No contexto do documento, o **parágrafo três** obscureceu a parte do **parágrafo quatro** que ficou atrás dele, no **eixo z**

Repare ainda que, ao sair do fluxo normal do documento, o **parágrafo três** "**libertou-se**" do seu container e assumiu uma largura equivalente a 100%

- ***Esquema Absoluto***
 - ***Posicionamento Absoluto***

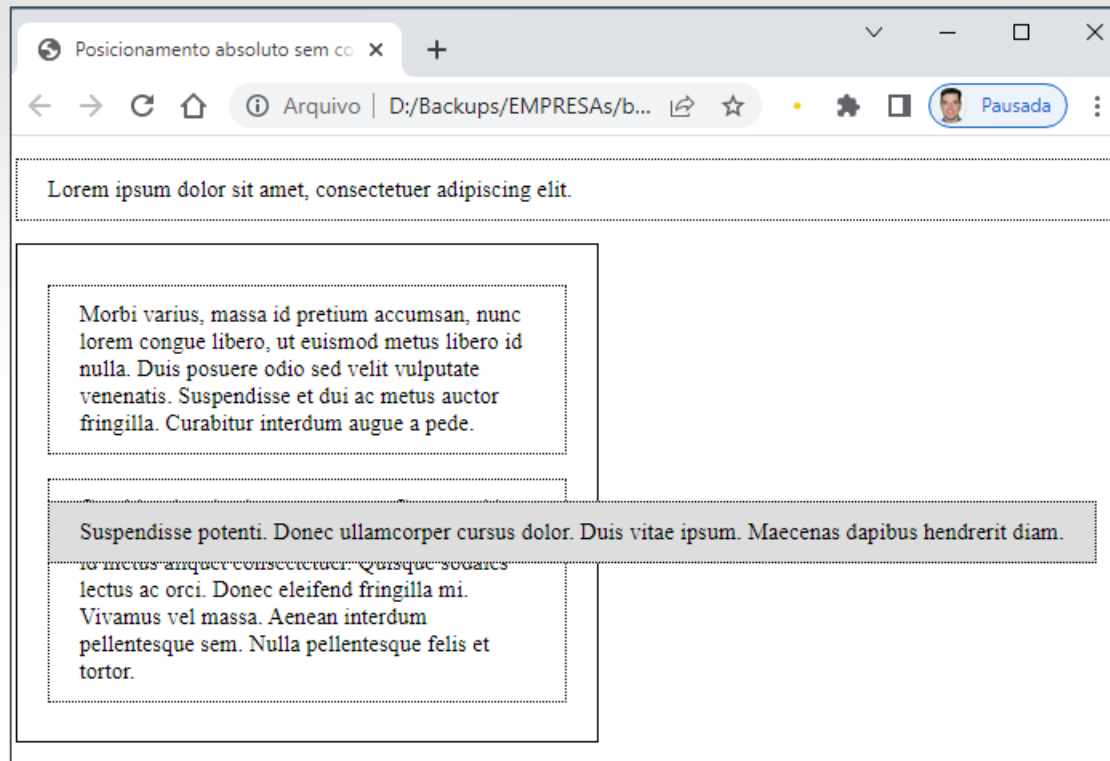


Figura 11 – Posicionamento absoluto sem coordenadas

- **Esquema Absoluto**
 - **Posicionamento Absoluto**
 - **Definir coordenadas para o posicionamento do *parágrafo três*, acrescentando as seguintes declarações na CSS**

```
.tres {  
    position: absolute;  
    background: #ddd;  
    left: 90px;  
    top: 125px;  
}
```

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- A origem padrão das coordenadas para o **posicionamento absoluto** (**contexto padrão**) é o **canto superior esquerdo** da área de renderização na janela do navegador (**viewport**)
 - Essa origem pode mudar dependendo do conceito conhecido como **contexto** de **posicionamento**
 - A regra CSS aplicada ao **parágrafo três** está definida no contexto de **posicionamento padrão**

- **Esquema Absoluto**
 - **Posicionamento Absoluto**

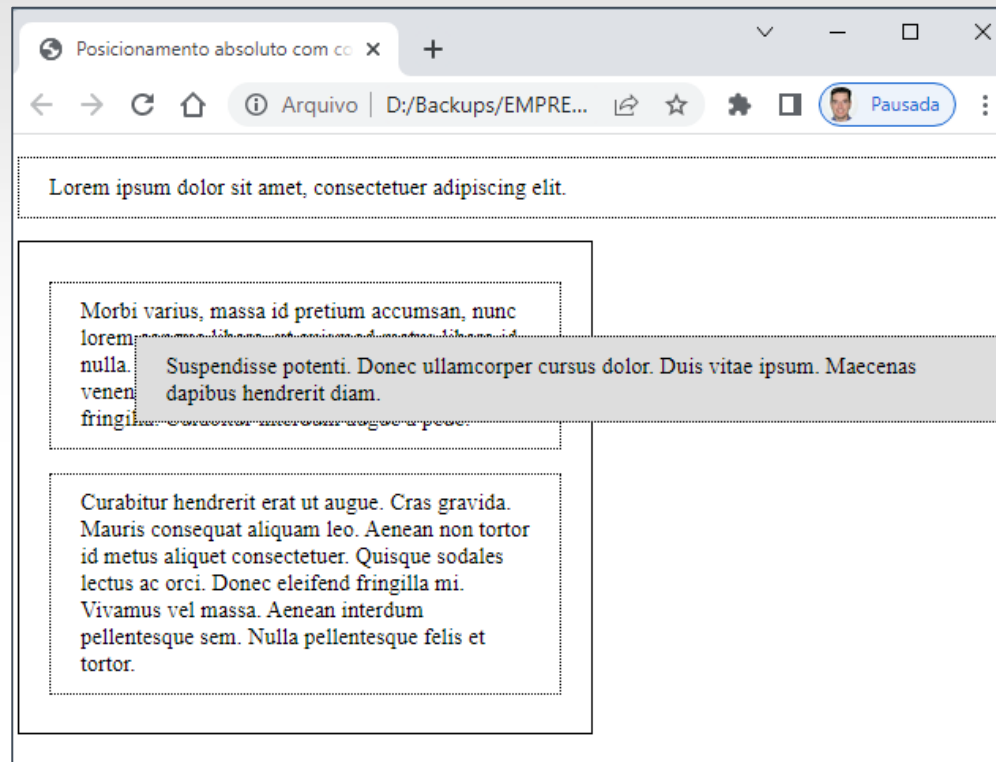


Figura 12 – Posicionamento absoluto com coordenadas

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- O **contexto** de **posicionamento** é que **estabelece a origem do sistema de coordenadas para posicionamento absoluto**
 - A **regra** que **rege o posicionamento** estabelece que **elementos posicionados com a declaração **position: absolute** serão deslocados, tomando como base para determinação das coordenadas o ancestral mais próximo posicionado**, ou seja, no qual tenham sido declarados os valores **relative**, **absolute** ou **fixed** para a **propriedade position** (**não** havendo **ancestral posicionado**, o **contexto de posicionamento** será a **viewport**)

- **Esquema Absoluto**

- **Posicionamento Absoluto**

- Para elucidar o conceito de **contexto** de **posicionamento**, considere o mesmo cenário do exemplo anterior e, adicionalmente, posicionar o elemento **div.container** que contém o **parágrafo três**
 - Declare **position: relative** para aquele **div** sem declarar coordenadas, mantendo-a em seu lugar no posicionamento para o **parágrafo três**, pois agora o elemento-pai do parágrafo foi posicionado, e a origem do sistema de coordenadas será tomada a partir dele

- ***Esquema Absoluto***
 - ***Posicionamento Absoluto***
 - ***Adicione a regra CSS na folha de estilo***

```
.container { position: relative; }
```

- **Esquema Absoluto**
 - **Posicionamento Absoluto**

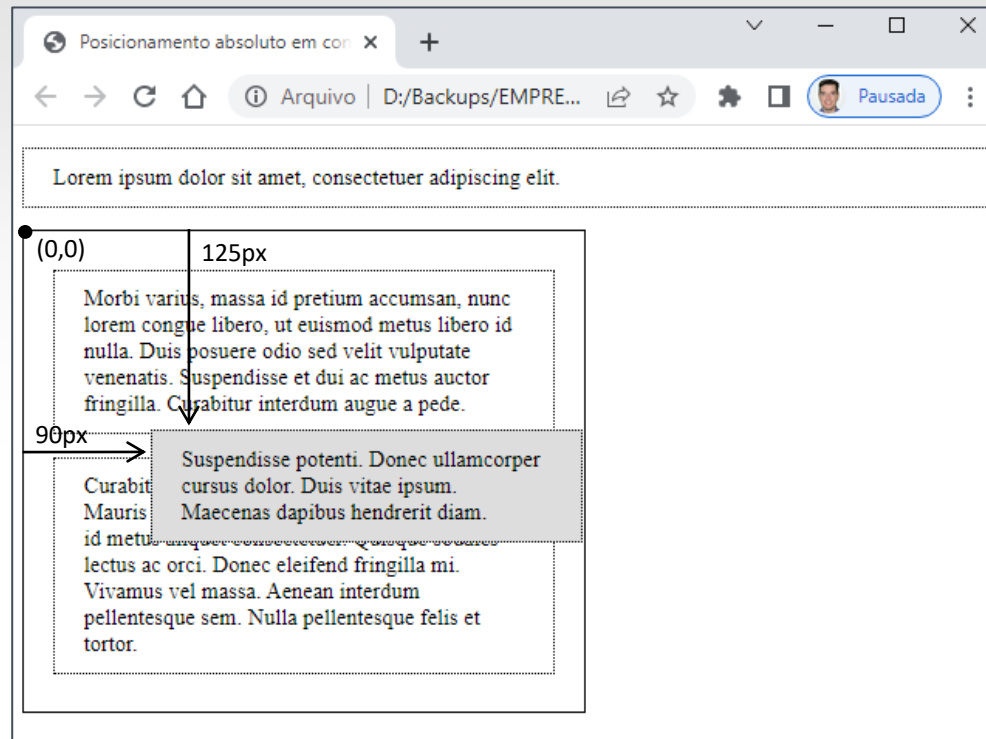
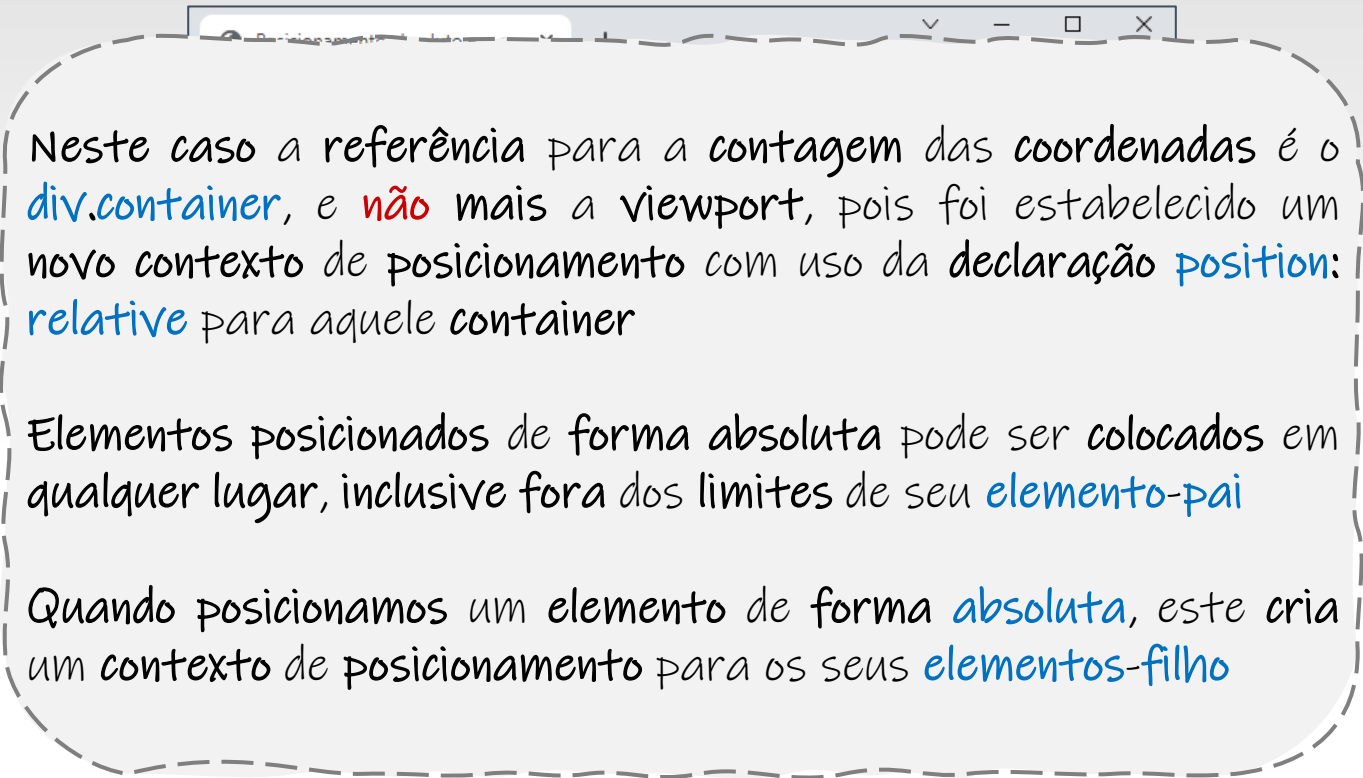


Figura 13 – Posicionamento absoluto em contexto

- **Esquema Absoluto**
 - **Posicionamento Absoluto**

A diagram showing a browser window with a dashed border. Inside, there is a rounded rectangle with a dashed border. The text inside this rectangle explains that the reference for absolute coordinates is the 'div.container' (highlighted in blue), not the viewport, because a new positioning context was established with the 'position: relative' declaration (highlighted in blue) for that container.

Neste caso a referência para a contagem das coordenadas é o **div.container**, e **não** mais a viewport, pois foi estabelecido um novo contexto de posicionamento com uso da declaração **position: relative** para aquele container

Elementos posicionados de forma absoluta pode ser colocados em qualquer lugar, inclusive fora dos limites de seu **elemento-pai**

Quando posicionamos um elemento de forma **absoluta**, este cria um contexto de posicionamento para os seus **elementos-filho**

Figura 13 – Posicionamento absoluto em contexto

- **Esquema Estático**

- Os demais valores possíveis para a propriedade CSS **position** são: **static**, **fixed** e **inherit**
- O valor **static** é o valor **inicial** ou **padrão** de **posicionamento**
- Os **boxes** permanecem como no **fluxo normal** do **documento**
- O uso dessa declaração é feito com a finalidade de **sobrescrever** um **posicionamento declarado anteriormente**
- Os valores de **posicionamento** **left**, **top**, **right** e **bottom** **não** se **aplicam** para esse **posicionamento**

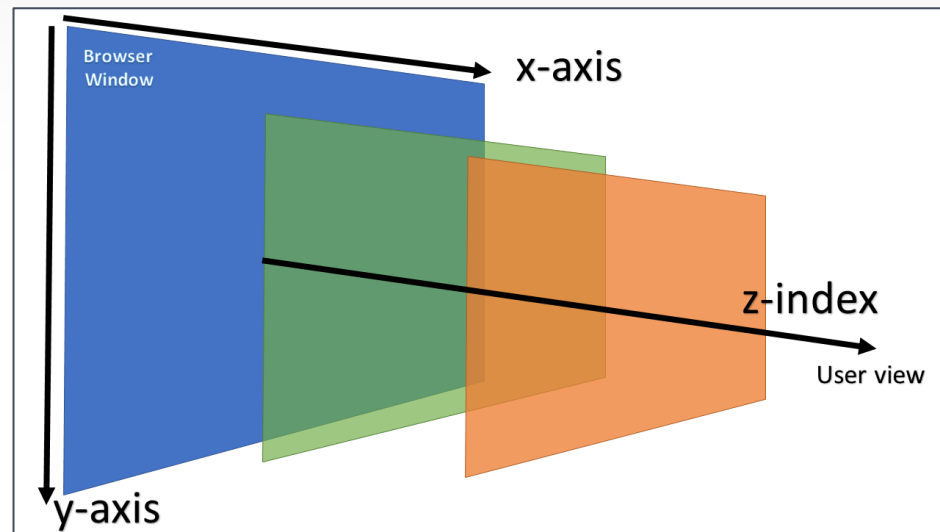
- **Posicionamento Fixo**

- **Aplicado** com a declaração **position: fixed** (**caso especial** de **posicionamento absoluto**)
- O **box posicionado** permanece **fixo** em relação a uma referência
- O contexto de posicionamento é sempre a **área** de **renderização** na **viewport**
- **Mídia visual**: o box fica **fixo** em relação à **viewport** e **não** se movimenta quando há rolagem da página
- **Mídia impressa**: o box será **impresso** em cada uma das folhas

- **Posicionamento com z-index**
 - Utilizado para o **empilhamento** segundo o **eixo z**
 - O **eixo z** é **perpendicular** à **tela** do **monitor** e tudo se passa como em um **sistema tridimensional** com **coordenadas horizontal, vertical** e **coordenada z** ou de **profundidade**
 - A **propriedade z-index** determina qual **box** fica à **frente**
 - Quando retiramos um ou mais **boxes** do **fluxo normal** do **documento** com uso da **propriedade position**, pode **ocorrer sobreposição** de **boxes** (**exemplos anteriores** - quando um **parágrafo** foi **colocado sobre outro**)

- **Posicionamento com z-index**

- Havendo **sobreposição** de **boxes** posicionados, entra em cena a **propriedade z-index**, responsável por **regular a ordem** de **empilhamento** no **eixo z** (**qual elemento está mais próximo do usuário e como eles se distribuem em profundidade**)



- **Posicionamento com z-index**

- *Considere a marcação abaixo, na qual **três elementos div** se seguem no **fluxo do documento***

```
<div class="um"> DIV UM </div>  
<div class="dois"> DIV DOIS </div>  
<div class="tres"> DIV TRÊS </div>
```

- *A **folha de estilo** posicionando os **três div***

```
div { border: 2px solid #000;  
      position: absolute;  
}
```

- **Posicionamento com z-index**
 - *A folha de estilo posicionando os **três div***

```
.um { width: 200px;  
      height: 80px;  
      background: #cff;  
}  
.dois { width: 110px;  
        height: 120px;  
        left: 120px;  
        top: 20px;  
        background: #6f9;  
}
```

- **Posicionamento com z-index**
 - A *folha de estilo* posicionando os **três div**

```
.tres {  
    width: 90px;  
    height: 60px;  
    left: 70px;  
    top: 50px;  
    background: #fcf;  
}
```

- **Posicionamento com z-index**
 - A *folha de estilo* posicionando os **três div**

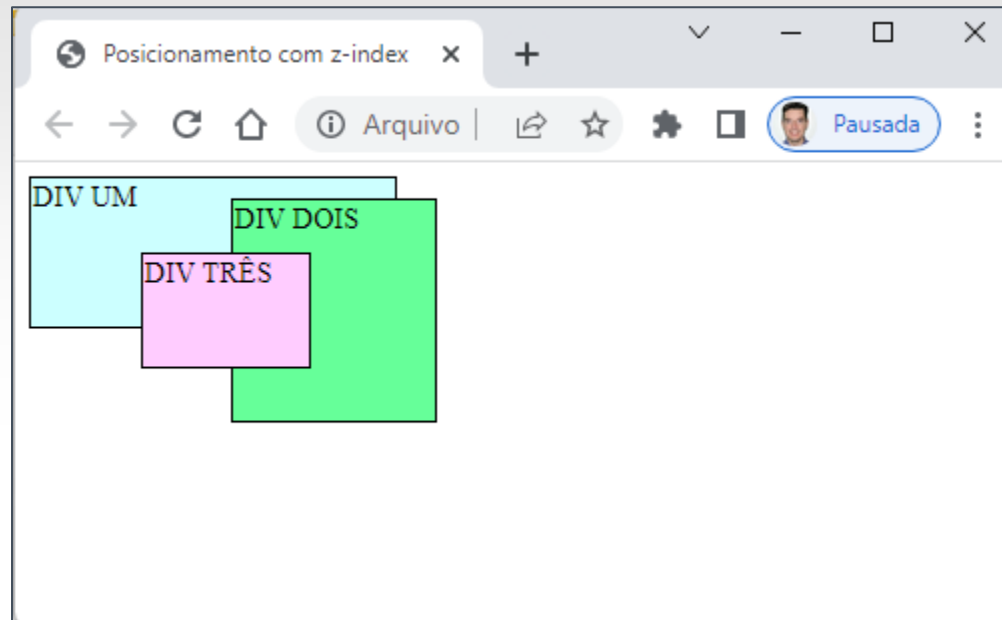


Figura 14 – Eixo z de profundidade

- **Posicionamento com z-index**
 - Quando **não** há declaração **explícita** de **z-index**, cabe ao agente de usuário determinar a ordem de empilhamento
 - A maioria dos navegadores adota como critério posicionar mais à frente os elementos que aparecem por último na marcação
 - Assim, o elemento **div** mais próximo do usuário é o **três**, sendo, em consequência, **posicionado** mais à frente, seguindo-se o **dois**, por último, o **um**



- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo do exemplo anterior***

```
div {  
    border: 2px solid #000;  
    position: absolute; }  
  
.um {  
    width: 200px;  
    height: 80px;  
    background: #cff;  
    z-index: 3; }
```

- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo do exemplo anterior***

```
.dois {  
    width: 110px;  
    height: 120px;  
    left: 120px;  
    top: 20px;  
    background: #6f9;  
    z-index: 2;  
}
```

- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo do exemplo anterior***

```
.tres {  
    width: 90px;  
    height: 60px;  
    left: 70px;  
    top: 50px;  
    background: #fcb;  
    z-index: 1;  
}
```


- ***Posicionamento com z-index***
 - ***Efeito causado pela nova folha de estilo***

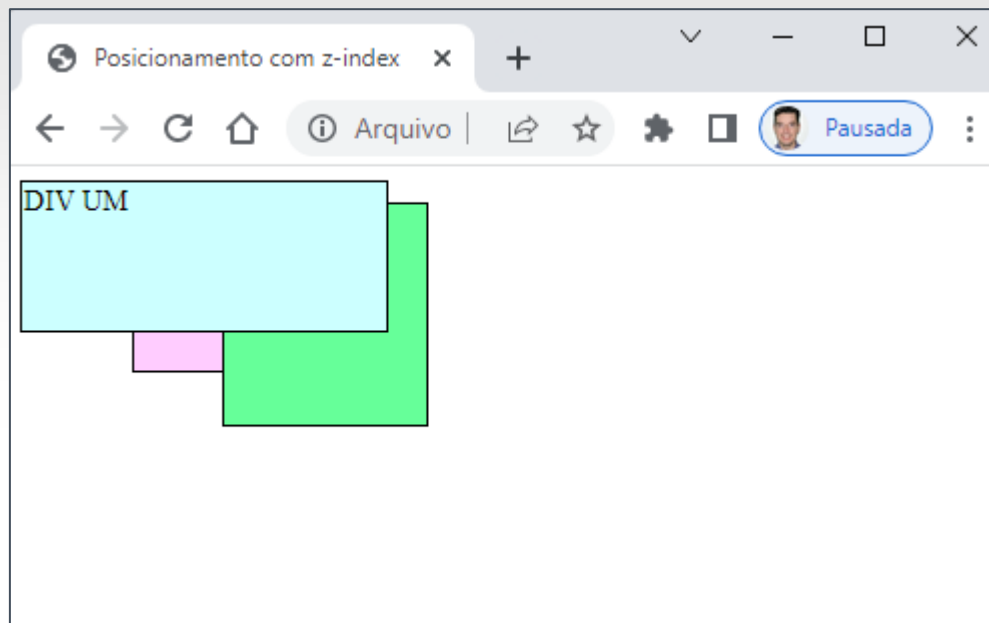


Figura 15 – Declaração z-index

- **Posicionamento com z-index**
 - **Efeito causado pela nova folha de estilo**

A **propriedade z-index** controla o **posicionamento** dos elementos **sobrepostos** no sentido da **profundidade**

Repare que, com a **declaração** dessa propriedade, **invertemos** a ordem-padrão de renderização apresentada anteriormente

Podemos estabelecer **qualquer ordem** de posicionamento para atender às necessidades do **layout**

Figura 15 – Declaração z-index

- **Posicionamento com z-index**
 - Os **valores possíveis** para a propriedade **z-index** são: **auto**, **integer** e **inherit**
 - O **valor inicial** ou **default** é **auto**
 - O **valor integer** (inteiro) é um **número inteiro positivo** ou **negativo**
 - **Não** existe obrigatoriedade quanto à sequência de números a adotar para declarar os valores inteiros de **z-index**
 - A única regra é, quanto maior o valor, mais próximo do usuário o box será posicionado (valores **1, 2, 3, 4, 5** tem o **mesmo efeito** de **36, 120, 204, 99**)

- **Posicionamento com z-index**

- Alguns agentes de usuário tratam **valores negativos** de **z-index** de forma **não prevista** nas recomendações do W3C para as CSS
- Ao invés de posicionar os elementos definidos com aqueles valores na **ordem natural** de crescimento do **eixo z**, colocam esses elementos **atrás** da tela, fazendo com que **desapareçam da vista do usuário**



- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - O **comportamento** de **renderização** para **posicionamento** em **profundidade** **não** é tão simples assim, havendo **outros fatores a considerar**
 - As **recomendações** do **W3C** para as **CSS** **preveem** um **contexto de empilhamento** segundo o **eixo** dos **z**
 - No **exemplo anterior**, os **três elementos** **div** foram **posicionados** de forma **absoluta**, tendo como **origem** das **coordenadas** **horizontal** e **vertical** o canto superior da **área de renderização** do navegador



Você se lembra do
conceito "contexto
de posicionamento"?



- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - **Todo elemento posicionado com o valor *relative*, *absolute* ou *fixed* cria um contexto de posicionamento, estabelecendo uma origem para contagem das coordenadas de posição *horizontal* e *vertical***
 - **De modo similar, um elemento posicionado da forma citada anteriormente cria um contexto de empilhamento, estabelecendo uma origem para contagem da coordenada *z* de profundidade de seus elementos-filho**

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Para **exemplificar** esse **conceito**, vamos **considerar** a seguinte **marcação**

```
<div id="a">  
  DIV A  
    <div class="a-um"> DIV A-um </div>  
    <div class="a-dois"> DIV A-dois </div>  
    <div class="a-tres"> DIV A-tres </div>  
</div>
```


- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Para **exemplificar** esse **conceito**, vamos **considerar** a seguinte **marcação**

```
<div id="b">  
  DIV B  
    <div class="b-um"> DIV B-um </div>  
    <div class="b-dois"> DIV B-dois </div>  
</div>
```

- **Posicionamento com z-index**

- **Contexto de Empilhamento**

- Na *marcação*, identificamos o **div A** que contém como **elementos-filho** os **div a-um**, **div a-dois** e **div a-tres**, e, em seguida, no **fluxo do documento**, o **div B** com os **div b-um** e **div b-dois**
 - A **folha de estilo** com **aplicação de bordas**, **cor de fundo** e **dimensões** para **fins de clareza do exemplo**

```
div {  
    border: 2px solid #000;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - *A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo*

```
div#a, div#b {  
    width: 300px;  
    margin-bottom: 20px;  
}  
  
div#a {  
    background: #ddd;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo***

```
div.a-um {  
    width: 200px;  
    height: 80px;  
    background: #cff;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo***

```
div.a-dois {  
    top: 70px;  
    width: 110px;  
    height: 120px;  
    background: #6f9;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo***

```
div.a-tres {  
    top: 90px;  
    width: 90px;  
    height: 60px;  
    background: #fcf;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo***

```
div.b-um {  
    width: 200px;  
    height: 120px;  
    background: #ffd1d1;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***A folha de estilo com aplicação de bordas, cor de fundo e dimensões para fins de clareza do exemplo***

```
div.b-dois {  
    width: 50px;  
    height: 60px;  
    background: #ccc;  
}
```


- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***

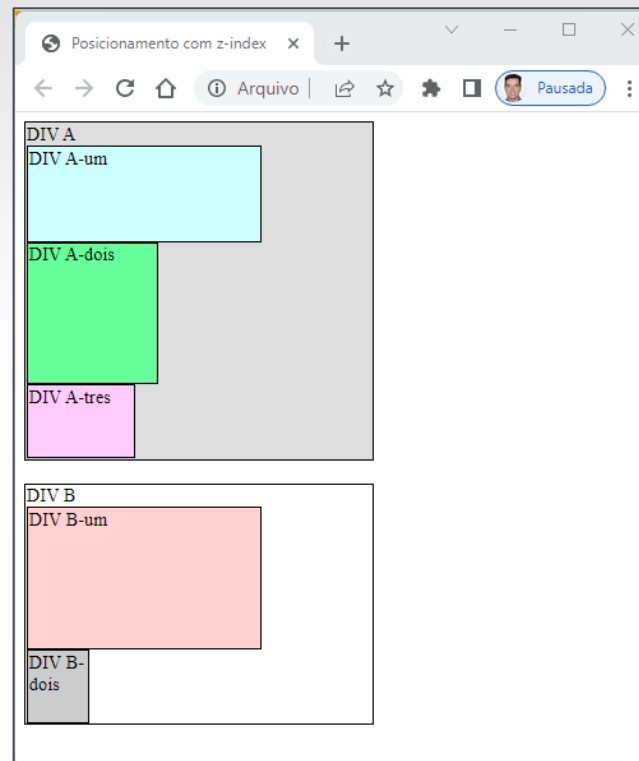


Figura 16 – Marcação básica

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos definir posicionamento **absoluto** para os cinco **elementos-filho** dos **div A** e **div B**, fazendo com que eles se sobreponham
 - Os **div A** e **div B** **não** serão posicionados de modo a **não** estabelecer um novo **contexto** de **empilhamento** para seus **elementos-filho**
 - O **contexto** de **empilhamento** será o **canto superior** da **área** de **renderização**

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div {  
    border: 2px solid #000;  
}  
  
div#a, div#b {  
    width: 300px;  
    margin-bottom: 20px;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***
 - ***Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento*****

```
div#a {  
    background: #ffc;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div.a-um {  
    position: absolute;  
    left: 65px;  
    top: 20px;  
    width: 200px;  
    height: 80px;  
    background: #cff;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div.a-dois {  
    position: absolute;  
    left: 120px;  
    top: 70px;  
    width: 110px;  
    height: 120px;  
    background: #6f9;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div.a-tres {  
    position: absolute;  
    left: 165px;  
    top: 90px;  
    width: 90px;  
    height: 60px;  
    background: #fcf;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div.b-um {  
    position: absolute;  
    left: 145px;  
    top: 130px;  
    width: 200px;  
    height: 120px;  
    background: #ffd1d1;  
}
```


- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Aplicaremos a **folha** de **estilo anterior** com as alterações definindo **posicionamento**

```
div.b-dois {  
    position: absolute;  
    left: 215px;  
    top: 170px;  
    width: 50px;  
    height: 60px;  
    background: #ccc;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***

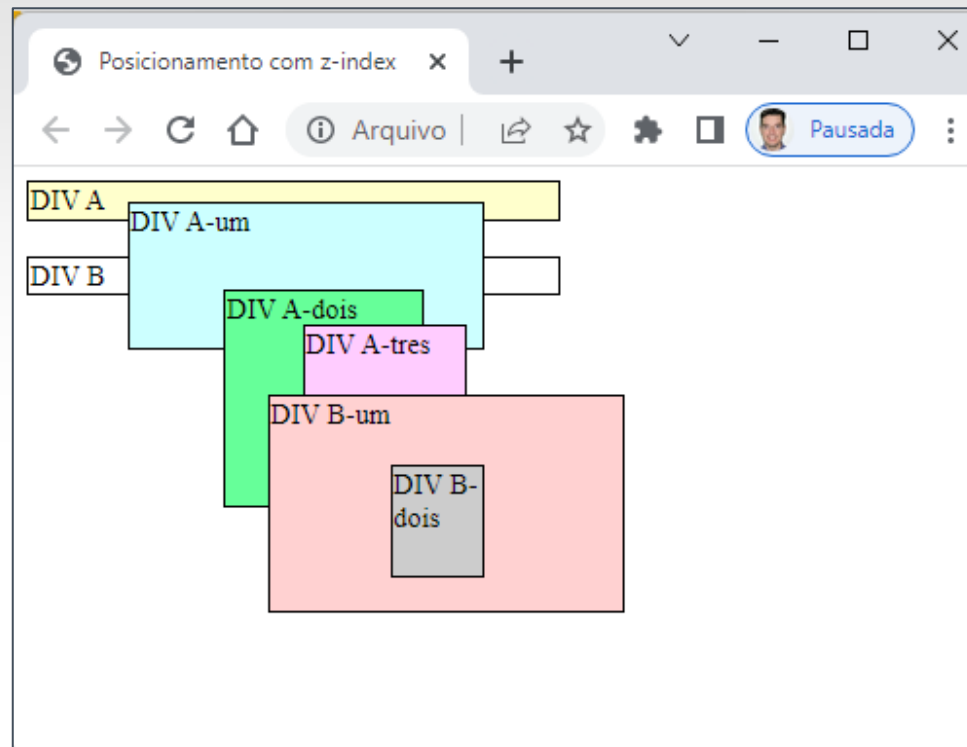


Figura 17 – Contexto de empilhamento-padrão

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Os **div A** e **div B** encolheram até uma **altura suficiente** para conter seus nomes (seus **elementos-filho** foram posicionados de forma **absoluta**, liberando o espaço que ocupavam no **fluxo** do **documento**)
 - A **ordem** de **empilhamento** obedece à ordem em que os **elementos-filho** posicionados aparecem na marcação (**nenhum** dos **elementos-filho** tem um **elemento ancestral** posicionado)

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div {  
    border: 2px solid #000;  
}  
  
div#a, div#b {  
    width: 300px;  
    margin-bottom: 20px;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div#a {  
    position: relative;  
    left: 10px;  
    top: 15px;  
    z-index: 25;  
    width: 350px;  
    height: 200px;  
    background: #ddd;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div#b {  
    position: relative;  
    top: -230px;  
    z-index: 20;  
    height: 200px;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div.a-um {  
    position: absolute;  
    left: 65px;  
    top: 20px;  
    width: 200px;  
    height: 80px;  
    background: #cff;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div.a-dois {  
    position: absolute;  
    left: 120px;  
    top: 70px;  
    width: 110px;  
    height: 120px;  
    background: #6f9;  
}
```


- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div.a-tres {  
    position: absolute;  
    left: 165px;  
    top: 90px;  
    width: 90px;  
    height: 60px;  
    background: #fcf;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div.b-um {  
    position: absolute;  
    left: 145px;  
    top: 130px;  
    width: 200px;  
    height: 120px;  
    background: #ffd1d1;  
}
```

- **Posicionamento com z-index**
 - **Contexto de Empilhamento**
 - Vamos **alterar** o **contexto** de **empilhamento** do exemplo anterior

```
div.b-dois {  
    position: absolute;  
    left: 215px;  
    top: 170px;  
    width: 50px;  
    height: 60px;  
    background: #ccc;  
}
```

- ***Posicionamento com z-index***
 - ***Contexto de Empilhamento***

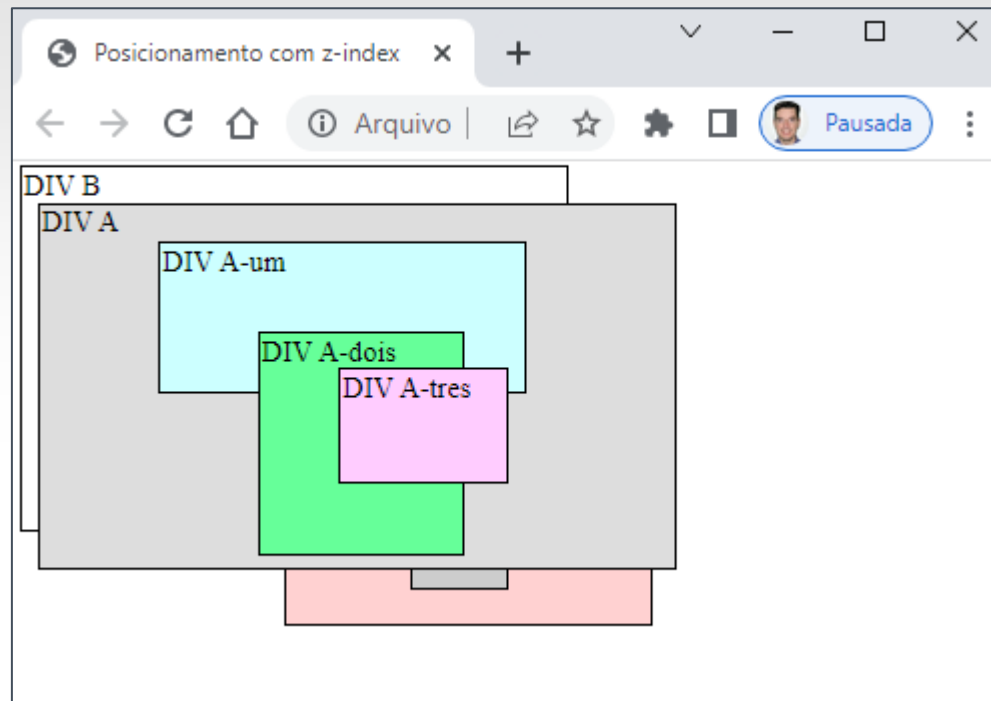


Figura 18 – Contexto de empilhamento z-index

- **Posicionamento com z-index**

- **Observação:**

- Os **div A** e **div B** foram posicionados de **forma relativa** e, em consequência, cada um deles estabeleceu um **contexto de empilhamento**
 - Para os **div a-um**, **div a-dois** e **div a-tres**, a **origem** do **sistema de coordenadas horizontal, vertical** e de **profundidade** passa a ser o **div A**, e **não** mais a **viewport**
 - O mesmo acontece para as **div b-um** e **div b-tres**, que **agora têm como base o sistema o div B** (tanto o **div A** quanto o **div B** estabeleceram um **novo contexto de empilhamento para seus elementos-filho**)

- **Posicionamento com z-index**

- **Observação:**

- Para os **div A** e **div B** definimos **largura**, **altura** e coordenadas **horizontal** e **vertical** com o propósito de tornar mais clara a renderização
 - Definimos **z-index** para os **div A** e **div B** (**conceito** de **contexto** de **empilhamento**), ou seja, definimos um **valor z-index** para o **div A** maior do que para o **div B** (**invertemos o comportamento de sobreposição-padrão**)
 - A **div A** foi colocada à **frente** do **div B** e todos os seus **elementos-filho** ficarão sempre à **frente** dos **elementos-filho** do **div B** independentemente do **valor** de **z-index**

- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo***

```
div {  
    border: 2px solid #000;  
}
```

```
div#a {  
    ...  
    z-index: 25;  
    ... }
```

- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo***

```
div#b {  
  ...  
  z-index: 20;  
  ... }
```

```
div.a-um {  
  ...  
  z-index: 10;  
}
```


- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo***

```
div.a-dois {  
    ...  
    z-index: 5;  
}
```

```
div.a-tres {  
    ...  
    z-index: 2;  
}
```

- ***Posicionamento com z-index***
 - ***Considere os seguintes acréscimos na folha de estilo***

```
div.b-um {  
    ...  
    z-index: 9500;  
}
```

```
div.b-dois {  
    ...  
    z-index: 1000;  
}
```

- ***Posicionamento com z-index***

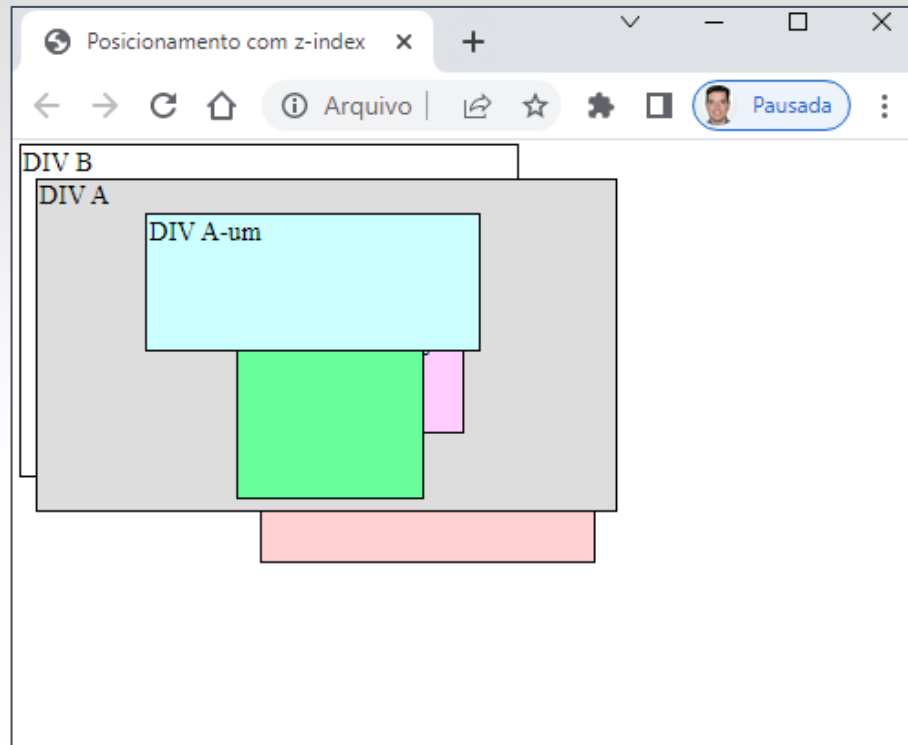
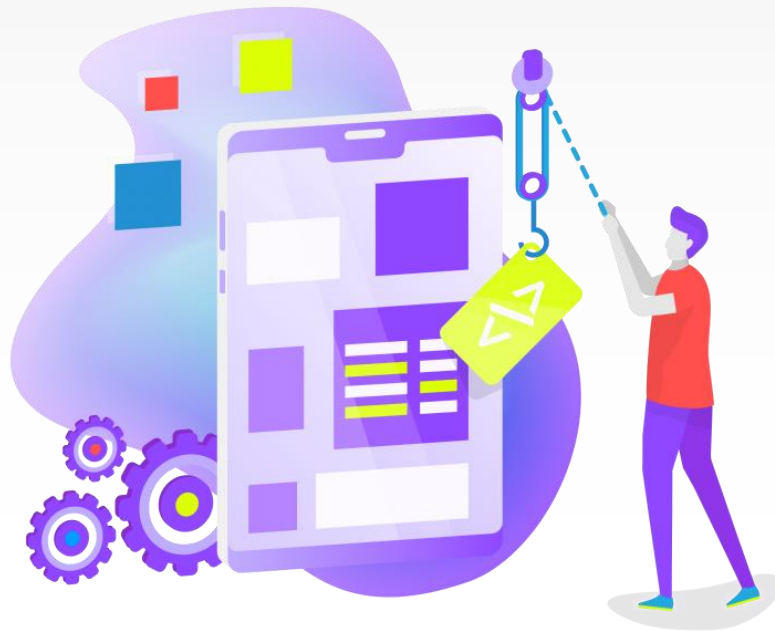


Figura 19 – Empilhamento z-index



EXERCÍCIOS

Referências

Silva, M. S. Fundamentos de HTML5 e CSS3. 1. ed. São Paulo: Novatec, 2015.

Duckett, J. HTML e CSS Projete e Construa Websites. 1. ed. Rio de Janeiro: Alta Books, 2011.

Hyslop, B., Castro, E. HTML and CSS: Visual Quickstart Guide. 8. ed. Barcelona: Peachpit Press, 2013.

