Universidade Federal
de Mato Grosso
Campus Rondonópolis

Programação em Ambiente Web I

Aula 3 – Posicionamento de elementos com CSS

Prof. Msc Cleyton Slaviero

cslaviero@gmail.com

Agradecimentos ao Prof. Jivago Medeiros (UFMT-Cba)
Fabiano Taguchi (UFMT-Roo)

O fluxo HTML

- Cada elemento da página aparece em uma determinada posição
- Quem determina essa posição?
  - Nós!
- Cada elemento é posicionado com base em sua colocação no documento HTML
- Hoje veremos que podemos alterar esse fluxo, com CSS!

Universidade Federal
de Mato Grosso
Campus Rondonópolis

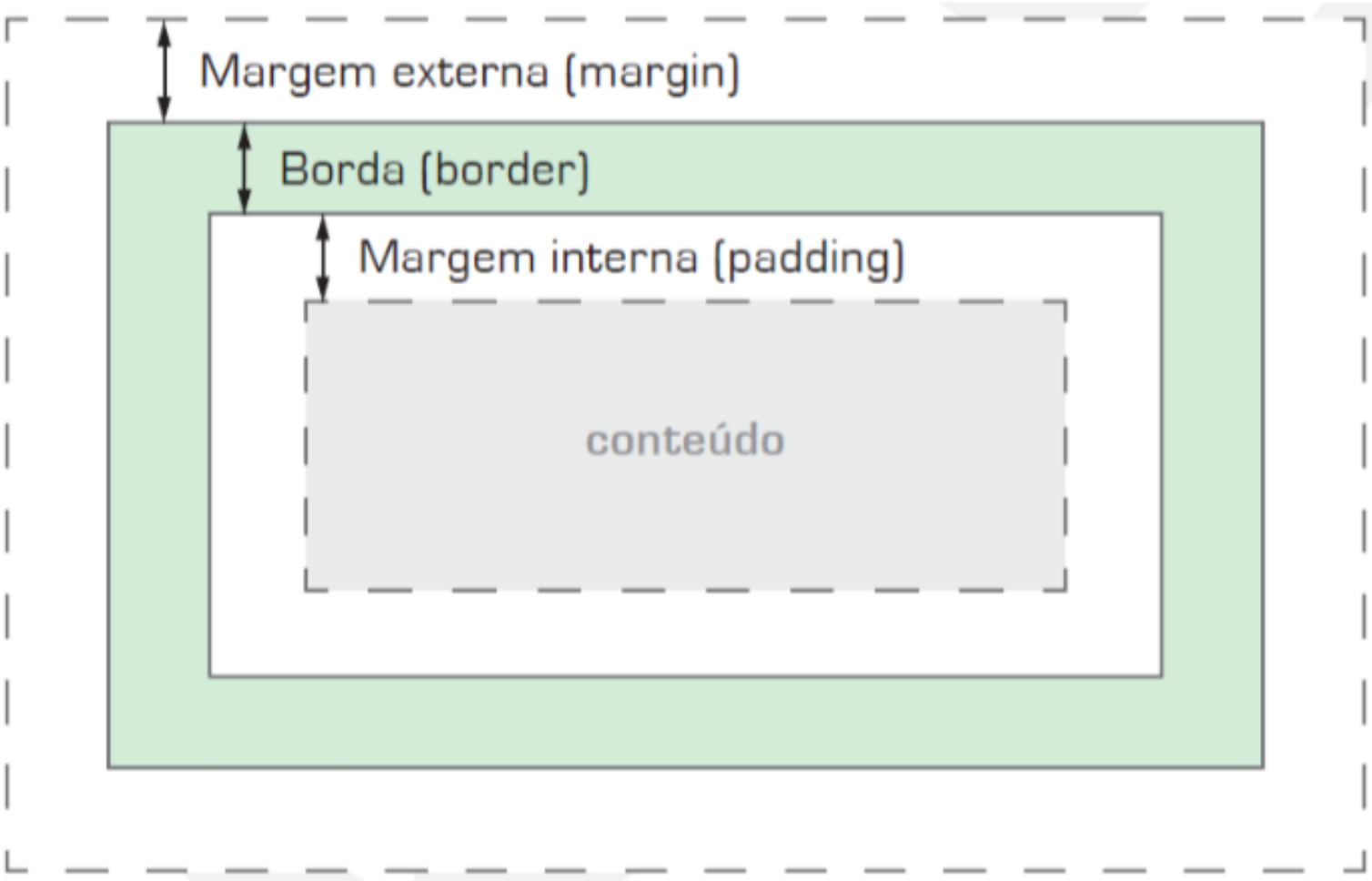
Box model

- Uma regra CSS considera e tratam todos os elementos HTML como se fossem caixas.
  - Nessas caixas algumas propriedades podem ser alteradas:
    - **Largura e altura;**
    - **Bordas;**
    - **Margens e espaçamentos.**

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Posicionamento de elementos

- Diferentes propriedades são utilizadas em CSS para definirmos o posicionamento dos elementos, a citar:
  - `margin`,
  - `padding`,
  - `position`,
  - `float`,
  - `top`,
  - `bottom`,
  - `left e`
  - `right`



Propriedade margin

- A propriedade `margin` define a distância do elemento em relação ao elemento pai.
- Pode ser definida de forma composta, exemplo: margin: *8px 2px 4px 8px*;
- Nesse caso, estaríamos definindo 8px para a **margem superior,** 2px para a margem **direita**, 4px para a **margem inferior** e 8px para a margem **esquerda.**

Propriedade margin

- Ou, também, definida individualmente para cada margem do elemento

```
margin-top: 8px;
margin-right: 2px;
margin-bottom: 4px;
margin-left: 8px;
```

Propriedade margin

- Além da definição composta da propriedade `margin` apresentada no **slide 5**, a propriedade também aceita as seguintes definições:

```
margin: 8px;
margin: 8px 4px;
margin: 8px 4px 2px;
```

Primeiro caso: 8px é atribuído para **todas** as margens
Segundo caso: 8px é atribuído as margens **superiores e inferiores** e 4px as margens **direita e esquerda**.
Terceiro caso: 8px é atribuído a margem superior, 4px as margens **direita e esquerda** e 2px a margem inferior.

Universidade Federal de Mato Grosso
Campus Rondonópolis

Colapso de margens

Fonte: https://www.w3.org/TR/CSS2/box.html#collapsing-margins

- Quando blocos estão empilhados, a margem inferior do bloco de cima pode se combinar com a margem superior do bloco de baixo
- Margens verticais colapsam, exceto:
  - Margens do elemento raiz da caixa não colapsam
  - Se as margens superior e inferior de um elemento com clear forem adjacentes, suas margens colapsam com as margens adjacentes dos irmaos seguintes, mas a margem resultante não colapsa com a margem inferior do bloco pai
- Margens horizontais nunca colapsam

Colapso de margens

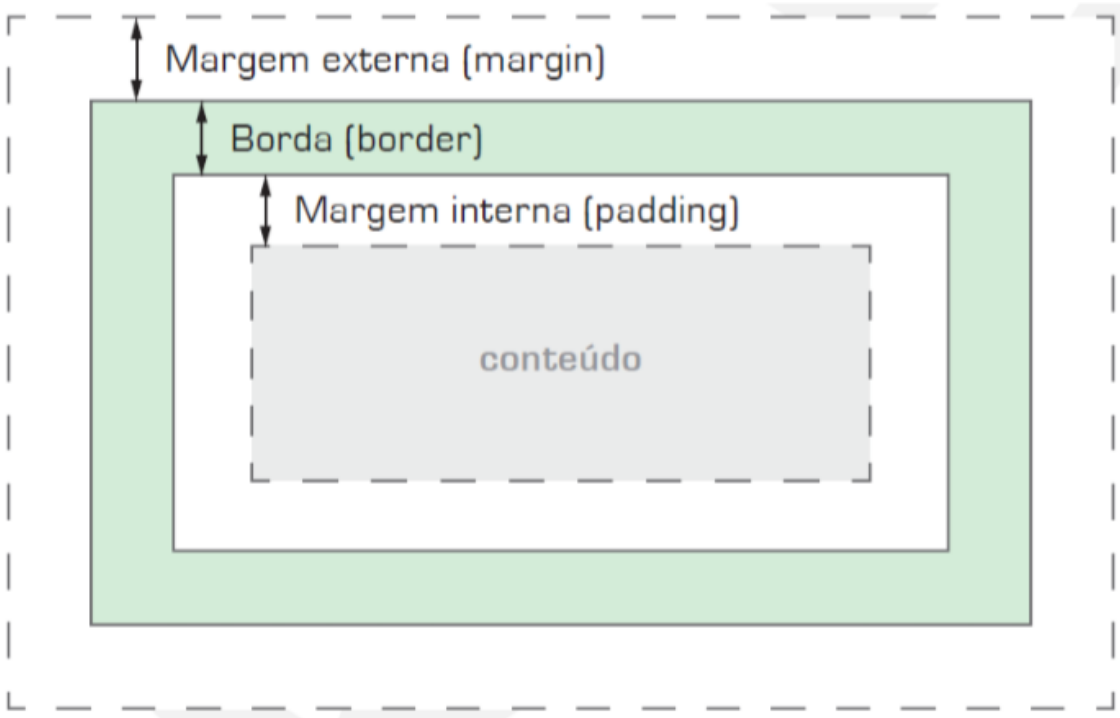
Fonte: https://www.w3.org/TR/CSS2/box.html#collapsing-margins

- Margens entre um box com float e qualquer outro box não colapsam (mesmo entre um *float* e seus descendentes que estiverem renderizados "no fluxo"/*in-flow).
- Margens de elementos que estabelecem novos contextos de bloco (como floats e elements com qualquer valor de 'overflow' exceto 'visible') não colapsam com seus descendentes "no fluxo".
- Margens de elementos com posição absoluta não colapsam (nem mesmo com seus descendentes "no fluxo").
- Margens de boxes *inline-block* não colapsam (nem mesmo com seus descendentes "no fluxo").
- A margem inferior de um bloco "no fluxo" sempre colapsa com a margem superior do próximo irmão que também esteja no fluxo e seja um bloco, exceto se esse irmão possuir 'clear' definido com uma valor que não seja 'none'.
- A margem inferior de um bloco no fluxo com 'height' com valor 'auto' e 'min-height' com valor zero colapsa com a margem inferior de seu último descendente imediato que seja um bloco e esteja no fluxo, se esse bloco não possuir padding inferior nem borda inferior, e se a margem desse elemento não colapsar com alguma outra margem superior que possua 'clear' definido.
- As margens de um box colapsam entre si se a propriedade 'min-height' tiver valor zero, e se ela não possuir bordas nem padding superiores ou inferiores, tiver valor 0 ou 'auto' para 'height', e não contiver um box de linha, contanto que todos seus descendentes diretos no fluxo colapsem.

Quiz

- O que acontece se fazemos o seguinte em um elemento?

```
.classe {
      width: 50px;
      height: 50px;
      border: 1px solid gray;
      padding: 10px 20px;
}
```



Resposta

- Largura
  - 50 + 20 + 20 + 1+ 1 = **92px**
- Altura
  - 50+10+10+1+1 = **72px**

- Porque isso acontece?
- Podemos mudar isso?

Universidade Federal
de Mato Grosso
Campus Rondonópolis

box-sizing

- Sim!
  - Propriedade `box-sizing`
  - Permite definir a partir de onde será calculado a largura e altura
    - `content-box:` o box model considera apenas o conteúdo par cálculo da largura e altura
    - `border-box:` O box-model considera o padding e o border como parte da largura e altura

Propriedade position

- Define o **tipo** de posicionamento que o elemento terá.
- Aplica-se a **praticamente** todos os objetos HTML
- O valor padrão para todos os elementos é **static**
- Os demais valores possíveis são: `absolute, relative, fixed.`

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Propriedade position

- `position:absolute`
  - O elemento é posicionado de maneira “absoluta” (fora do fluxo de elementos) por meio das propriedades: `top, right, bottom, left.`
  - Estas propriedades definem as distâncias superior, direita, inferior e esquerda em **relação ao elemento pai.**

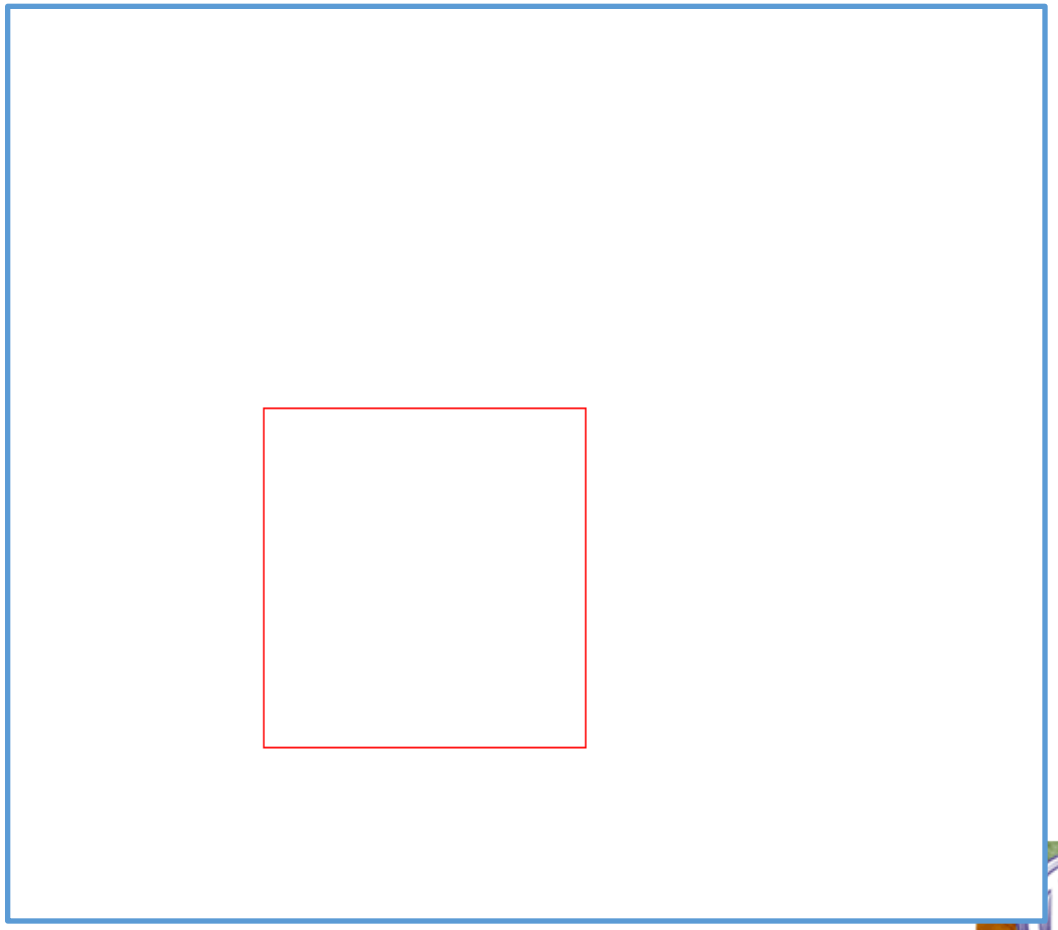
position: absolute

- Exemplo

```
<style>
        div#um {
              height: 200px;
              width: 200px;
              border: 1px solid #FF0000;
         position: absolute;
         left: 16px;
         top: 240px;
        }
</style>
<div id="um">
     
</div>
```

position: absolute

- Resultado esperado



position: absolute

- A primeira vista pode-se não notar a diferença em relação ao uso da propriedade margin, porém, é importante saber que:
  - As propriedades `top, right, bottom, left` **não possuem efeito** em elementos com `positon: static` (valor padrão)
  - **E principalmente:** Um elemento com posição absoluta se "desprende" do fluxo normal de elementos, sobressaindo sobre os demais.

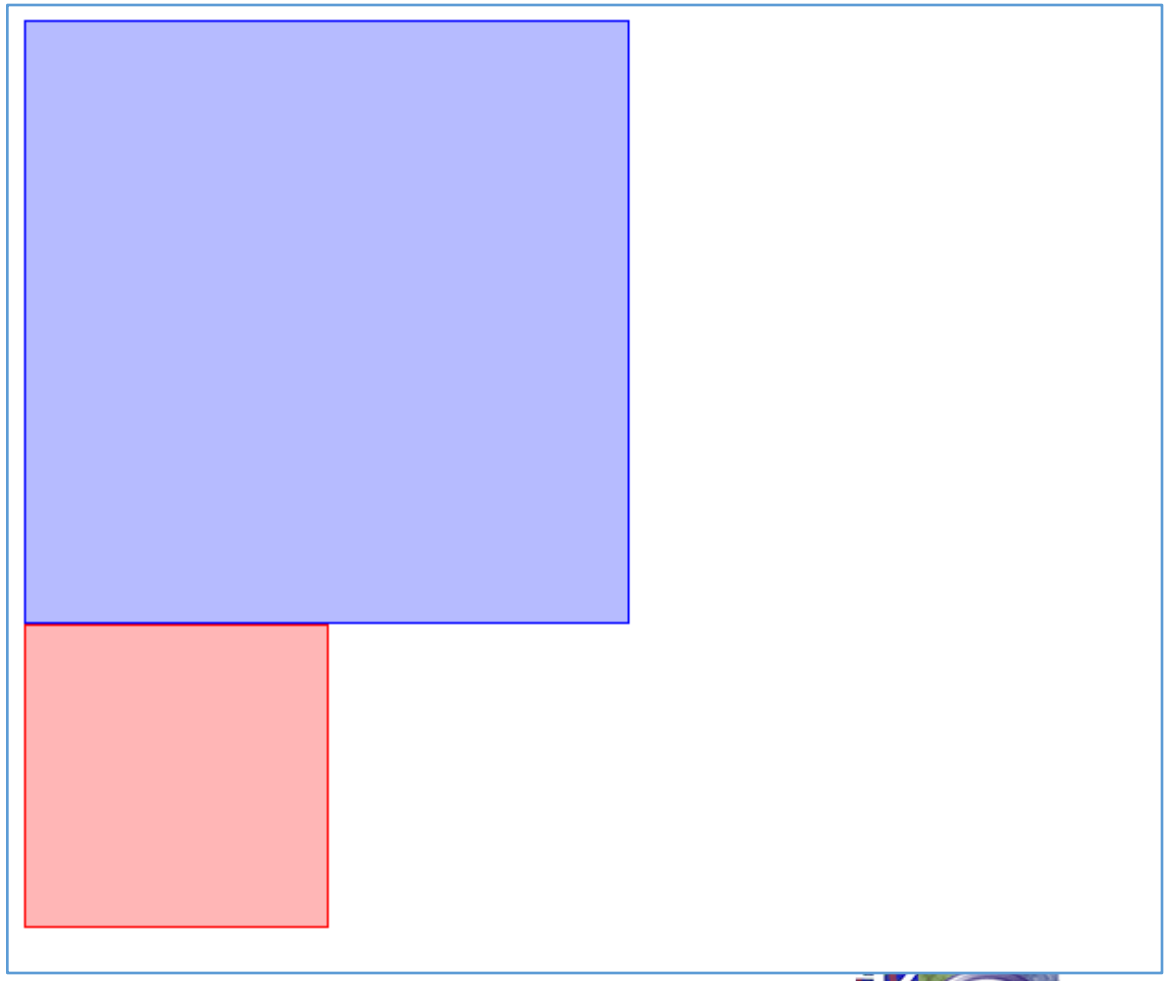
position: absolute

- Exemplo

```
<style>
    div#um {
        background-color: #b6bbff;
        height: 300px;
        width: 300px;
        border: 1px solid #0000FF;
        position: static;
    }
    div#dois {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: static;
        left: 80px;
        top: 120px;
    }
</style>
<div id="um"> </div>
<div id="dois"> </div>
```

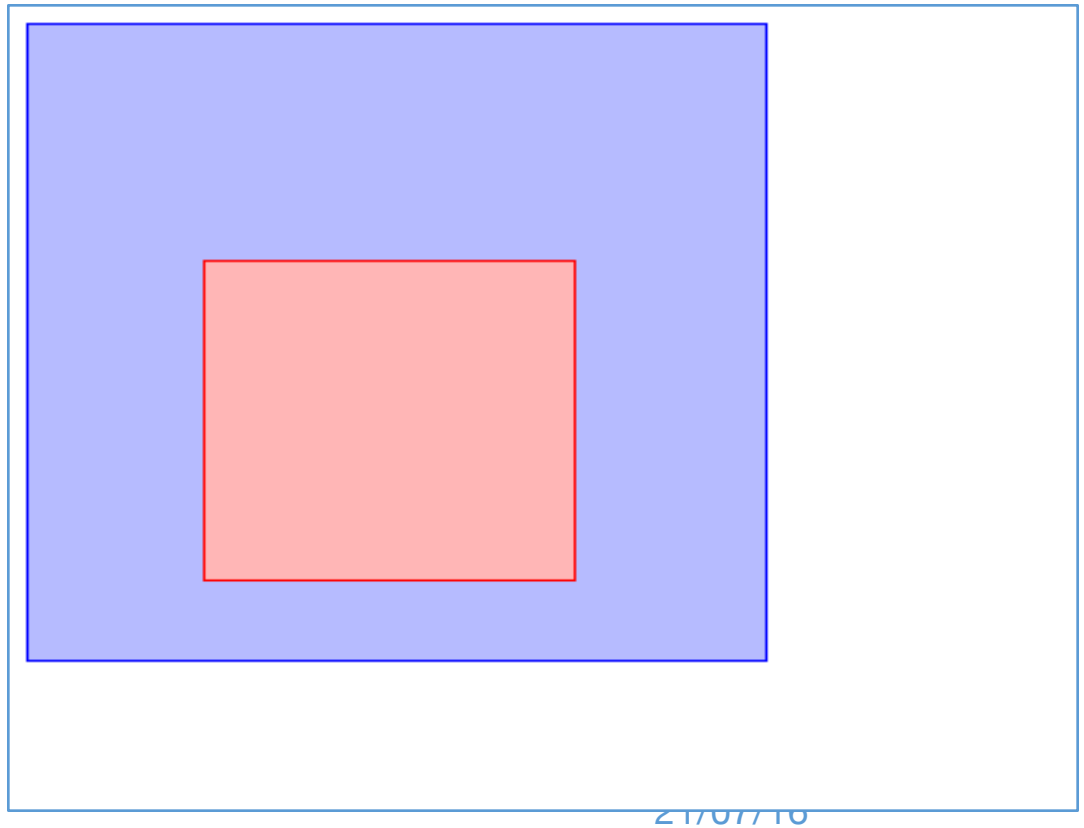
position: absolute

- Resultado esperado



position: absolute

Alterando a position do `div.dois` para `absolute`, o resultado é consideravelmente diferente:



Importante: Na realidade, o `div.dois` não está “dentro” do `div.um` e sim “sobre”.

Z-index

- A propriedade `z-index` é utilizada para definir a disposição de um elemento sobre outro: Qual estará a frente, e qual estará atrás.

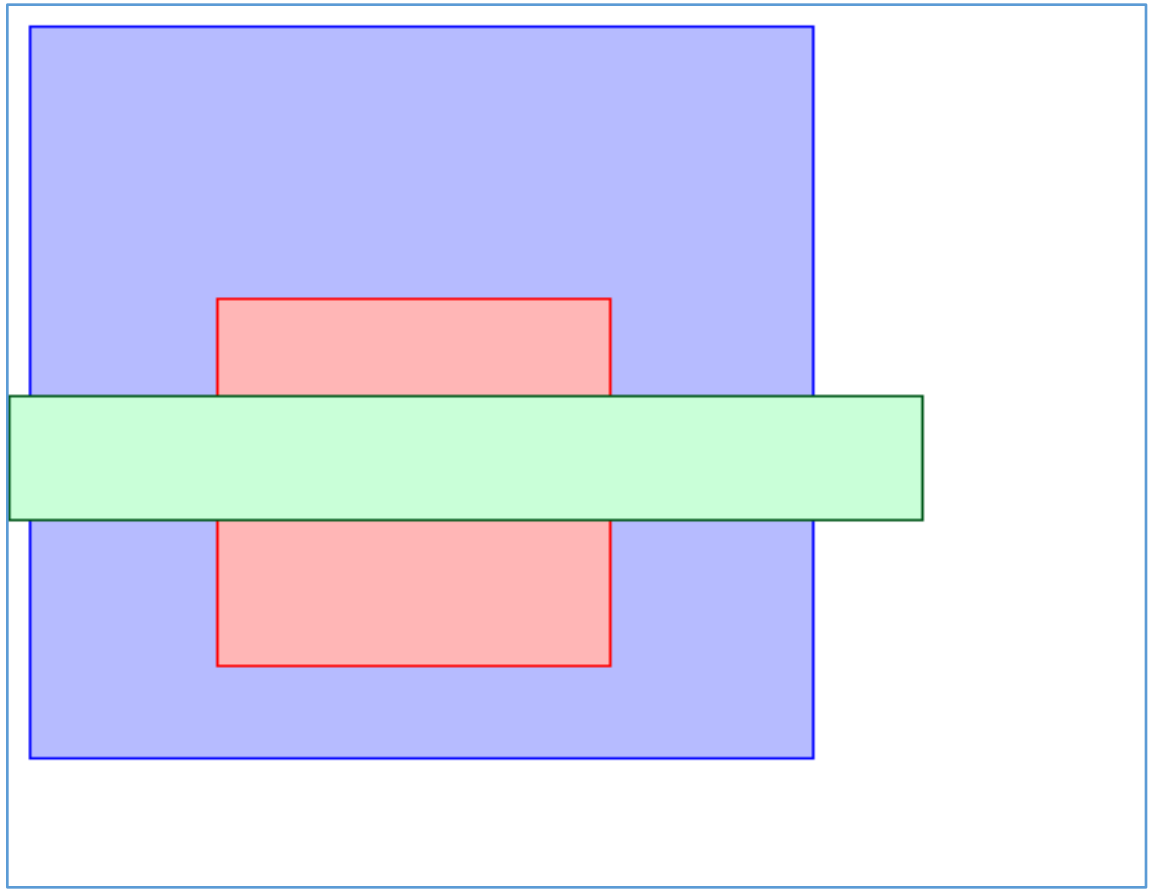
z-index

- Vamos adicionar mais um elemento ao nosso exemplo

```
<style>
    div#tres {
        background-color: #c9ffd8;
        height: 50px;
        width: 350px;
        border: 1px solid #005719;
        position: absolute;
        left: 0px;
        top: 160px;
    }
</style>
<div id="tres">
    
</div>
```

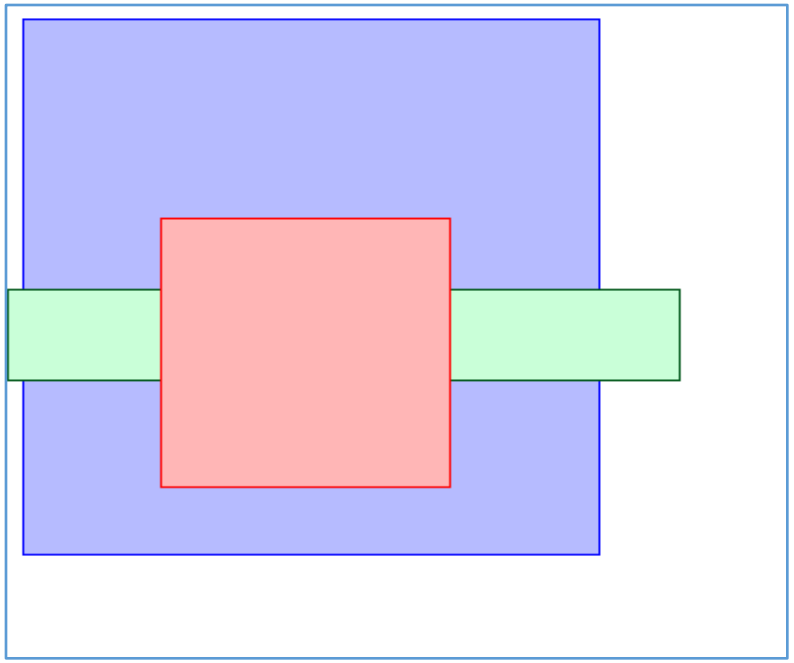
Z-index

- Resultado



Z-index

- Se adicionarmos a propriedade `z-index` para o `div.tres` com o **valor 1** e para o `div.dois` com o **valor 2**, o resultado será:



Position:absolute

- É possível aninhar elementos que possuam o valor de position como **`absolute`** (e **`relative`** também).

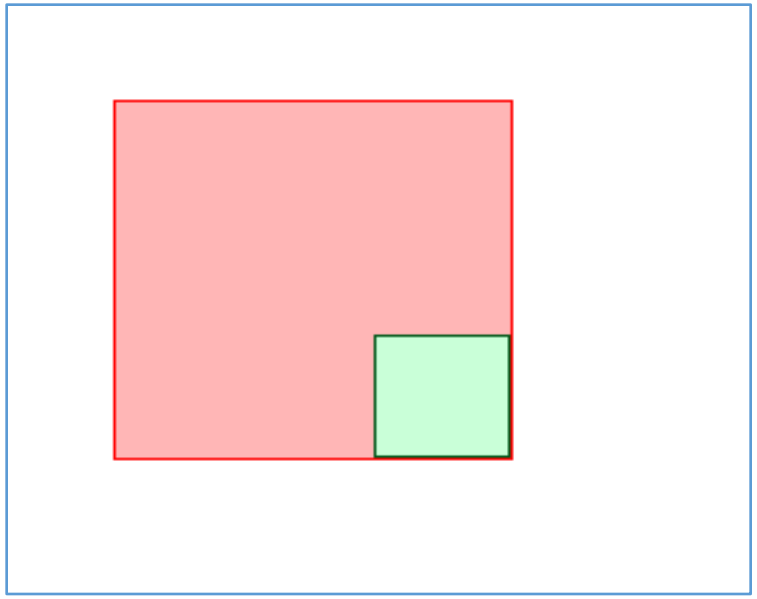
position: absolute

- Exemplo de elementos com **`position:`** `absolute` aninhados:

```
<style>
    div#um {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: absolute;
        left: 40px;
        top: 40px;
    }
    div#dois {
        background-color: #c9ffd8;
        height: 50px;
        width: 50px;
        border: 1px solid #005719;
        position: absolute;
        right: 0px;
        bottom: 0px;

    }
</style>
<div id="um">
    <div id="dois">
         
    </div>
</div>
```

position: absolute

- Resultado



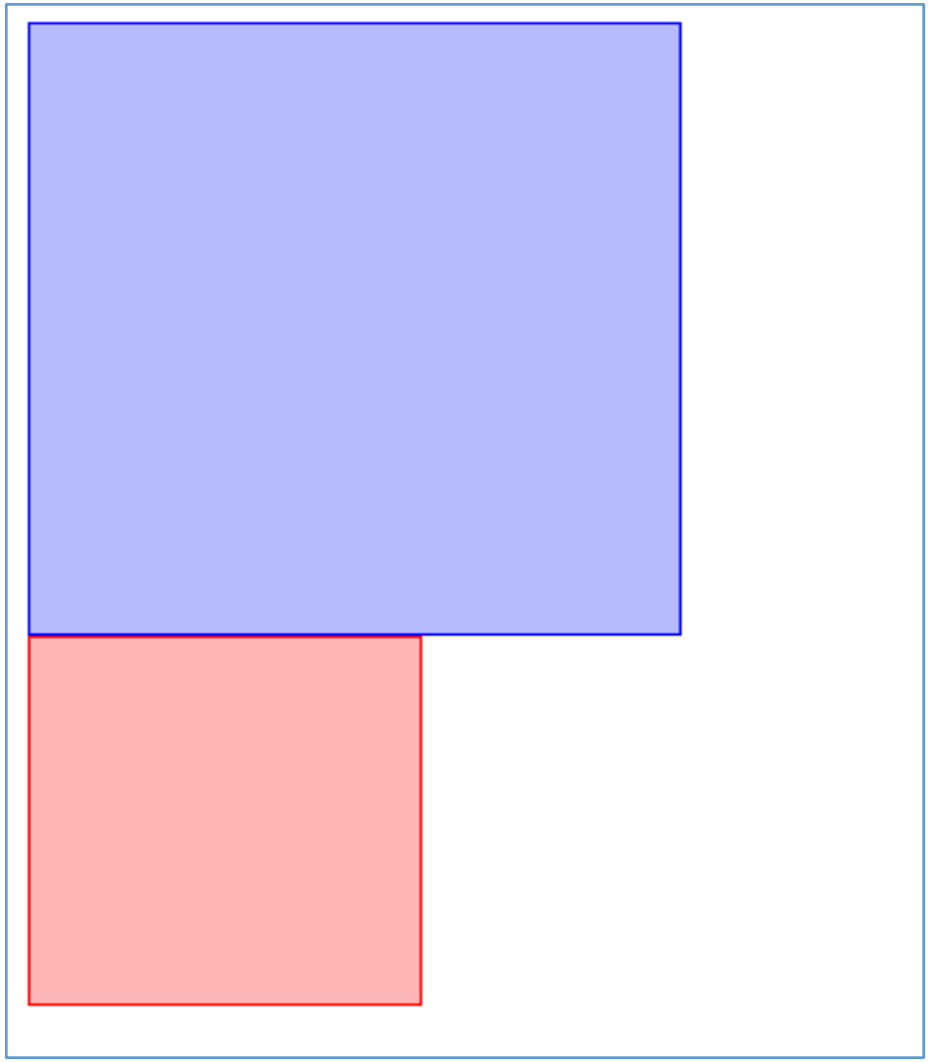
A “lógica” ainda é a mesma: As `propriedades top, right, bottom, left` são em relação aos elementos pai. No caso do `div.dois` o elemento pai é o `div.um` e no caso do div.um o elemento pai é o body.

Importante: No caso de elementos aninhados, a propriedade z-index não tem funcionalidade.

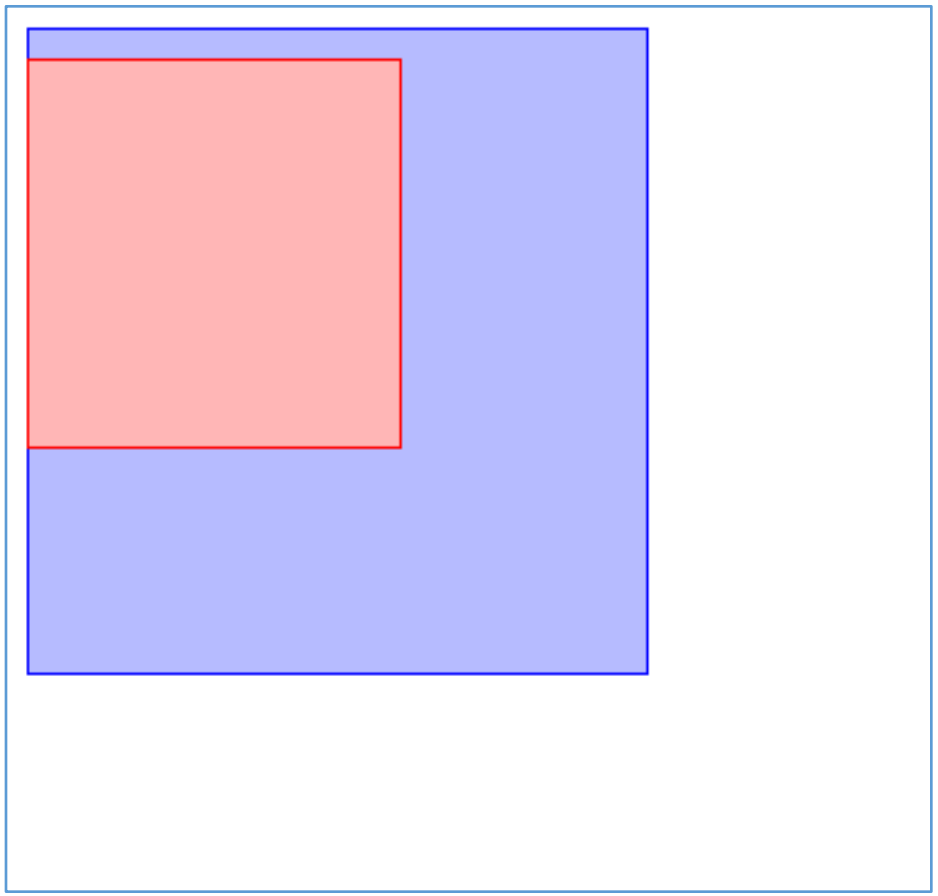
Position:relative

- O valor **relative** para a propriedade position adiciona um comportamento ao elemento análogo aos elementos com position **absolute**.
  - Todavia, as propriedades referentes a posicionamento (top, right, bottom, left) são **relativas** a posição do elemento no fluxo normal de elementos na página.

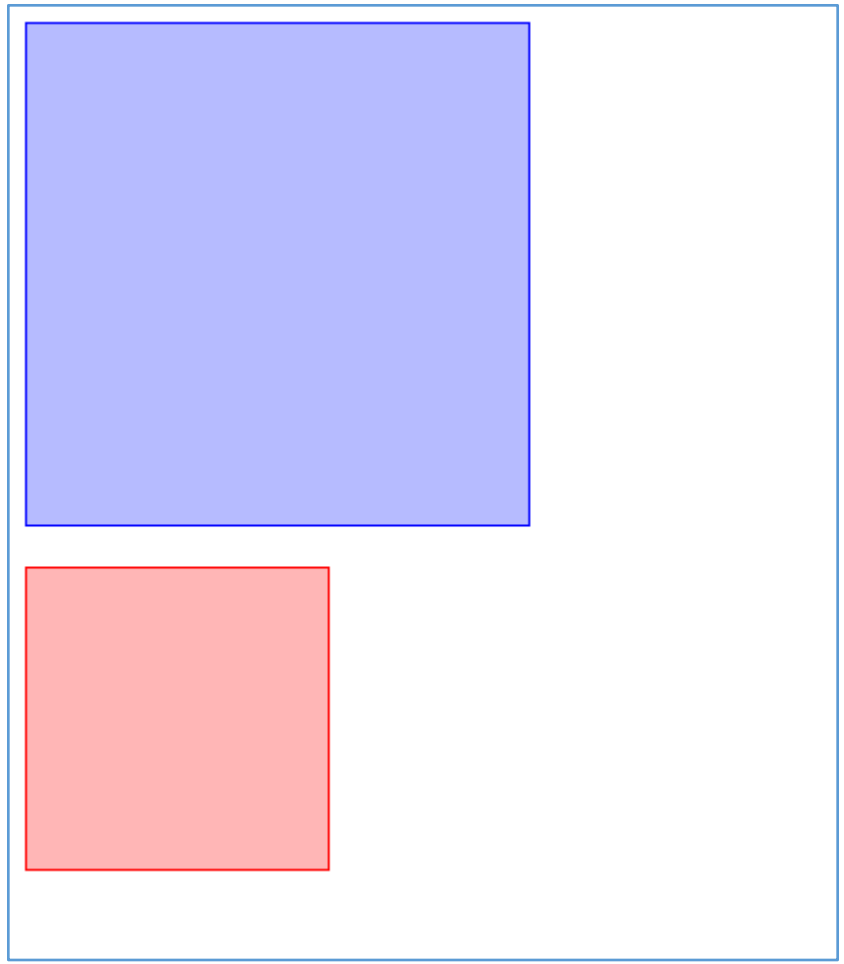
```
<style>
    div#um {
        background-color: #b6bbff;
        height: 250px;
        width: 250px;
        border: 1px solid #0000FF;
        position: static;
    }
    div#dois {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: static;
    }
</style>
<div id="um">
     
</div>
<div id="dois">
     
</div>
```



```
<style>
    div#um {
        background-color: #b6bbff;
        height: 250px;
        width: 250px;
        border: 1px solid #0000FF;
        position: static;
    }
    div#dois {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: absolute;
        top: 20px;
    }
</style>
<div id="um">
     
</div>
<div id="dois">
     
</div>
```



```
<style>
    div#um {
        background-color: #b6bbff;
        height: 250px;
        width: 250px;
        border: 1px solid #0000FF;
        position: static;
    }
    div#dois {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: relative;
        top: 20px;
    }
</style>
<div id="um">
     
</div>
<div id="dois">
     
</div>
```



position:fixed

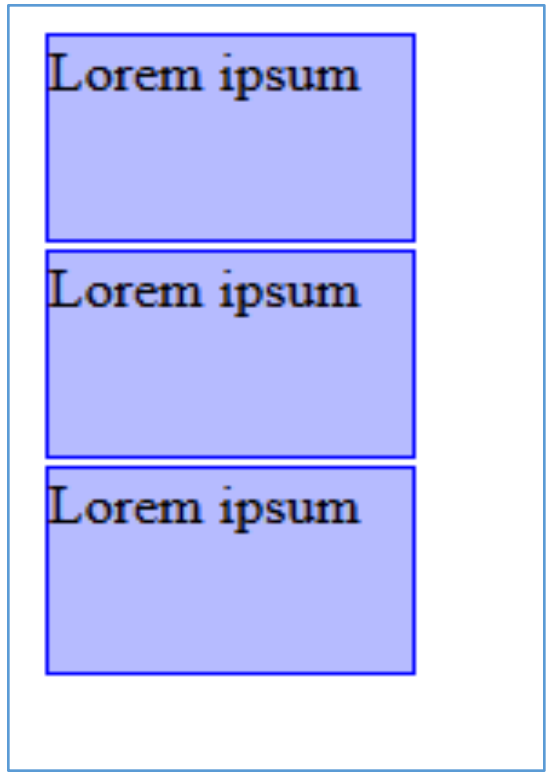
- O valor **fixed** para a propriedade position torna o elemento fixo à uma determinada posição na página mesmo quando a barra de rolagem é acionada.
  - A posição fixa do elemento é dada pelas propriedades top, right, bottom, left, cuja os valores acompanham a rolagem da barra de rolagem.

float

- Faz com que elementos flutuem para direita (right) ou para esquerda (left).
- Essa propriedade é muito útil quando queremos alinhar horizontalmente elementos que são exibidos como **block,** como por exemplo, divs.

float

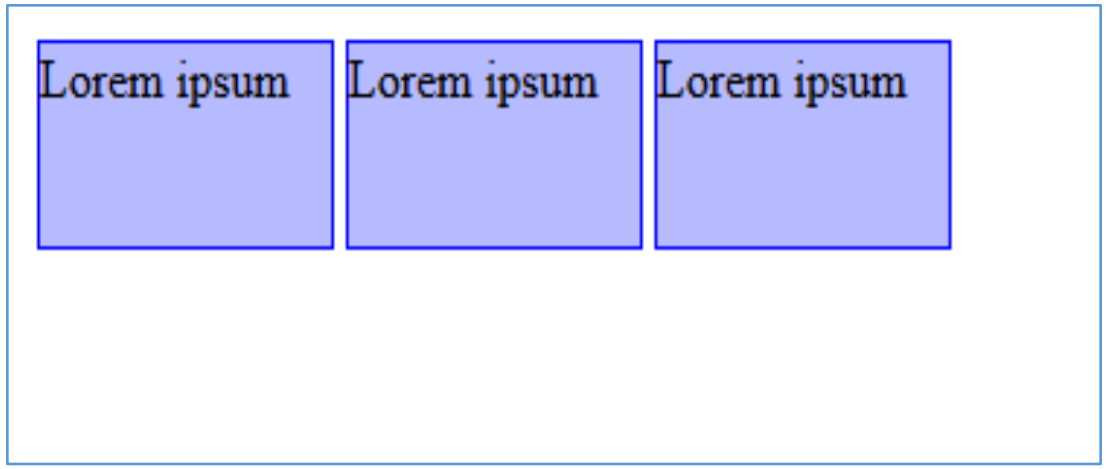
```
<style>
    div.exemplo {
        background-color: #b6bbff;
        border: 1px solid #0000FF;
        height: 60px;
        width: 100px;
        margin: 2px;
    }
</style>

<div class="exemplo">Lorem ipsum</div>
<div class="exemplo">Lorem ipsum</div>
<div class="exemplo">Lorem ipsum</div>
```



*Comportamento tradicional de elementos do tipo **block***

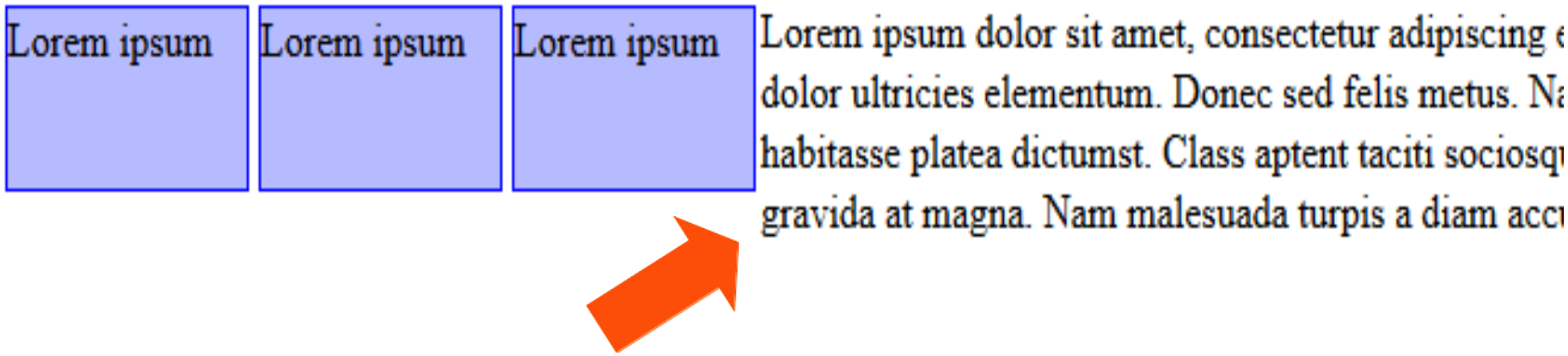
float

```
<style>
    div.exemplo {
        background-color: #b6bbff;
        border: 1px solid #0000FF;
        height: 60px;
        width: 100px;
        margin: 2px;
        float: left;
    }
</style>
<div class="exemplo">Lorem ipsum</div>
<div class="exemplo">Lorem ipsum</div>
<div class="exemplo">Lorem ipsum</div>
```



float

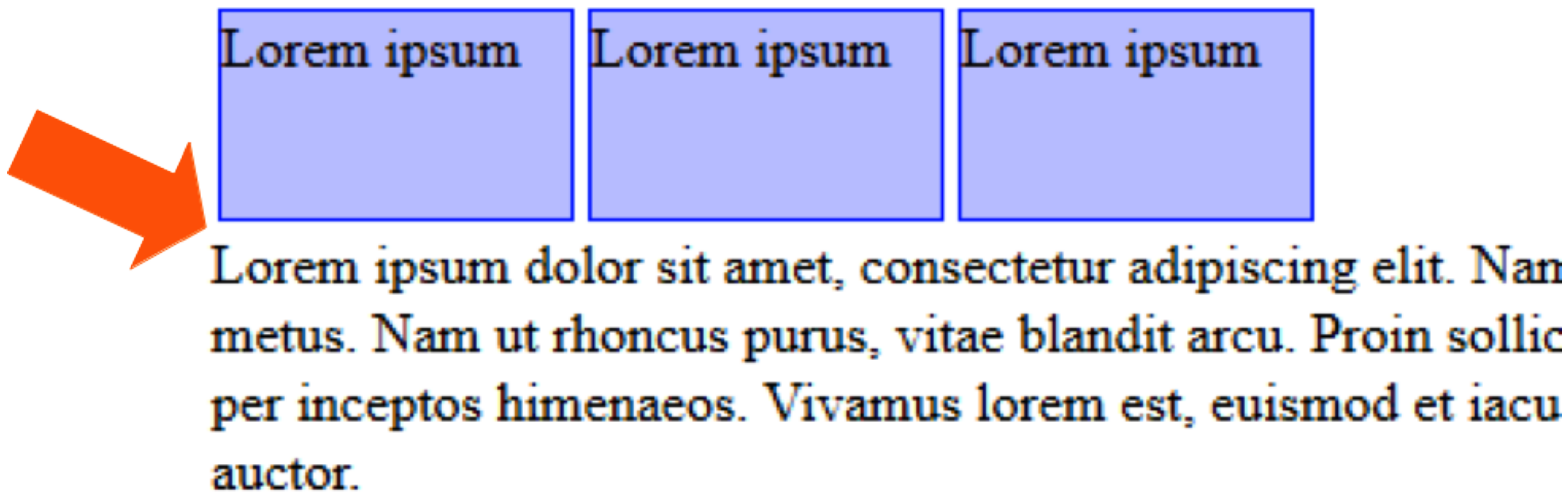
- **Importante:** Quando o navegador encontra um elemento com a propriedade **float**, ele “tenta alinhar” os demais elementos na sequência:



float

- Para que não aconteça, deve-se adicionar a propriedade clear ao elemento subsequente:

```
<p style="clear: both;">...</p>
```



Propriedade display

- Esta propriedade afeta a maneira como tal elemento será posicionado.

- `block`
  - O elemento é posicionado logo abaixo e terá a largura definida pelo atributo width
- `inline`
  - Outros elementos posicionados um ao lado do outro (na mesma linha)
- `inline-block`
  - Outros elementos ao redor são posicionados um do lado do outro. Faz com que existam elementos inline e block

Pseudo classes e pseudoelementos

- CSS nos permite manipular determinadas características da página por meio de seletores especiais
  - Pseudo-classe
    - Especifica estados especiais do elemento selecionado
  - Pseudo-elemento
    - Permite a aplicação de estilos a características não mapeados por elementos

Pseudoelemento

- **FIRST-LETTER**
  - Aplica a formatação do recurso de capitalizar um parágrafo, sua sintaxe é:

```
p:first-letter{
font-size: 300%;
}
```

Pseudoelemento

- **FIRST-LINE**
  - Apenas a primeira linha de um parágrafo ficar com letras maiúsculas, sua sintaxe é:

```
p:first-line{
    font-variant: small-caps;
    color: #0000ff;
}
```

Outros pseudo-elementos

- ::after
- ::before
- ::first-letter
- ::first-line
- ::selection
- ::backdrop

Pseudo-classes

- Personalização de links
  - Permitem a alteração de cores em links

```
a:link{
    color: blue;
}
a:visited{
    color: red;
}
```

Pseudo-classes

- Personalização durante estados:
  - Permitem alterar características quando, por exemplo, ativos ou qundo o mouse está em cima do elemento

```
a:active{
    background-color: blue;
}

a:hover{
    color: orange;
}
```

Mais pseudo-classes

- :active
- :checked
- :default
- :dir()
- :disabled
- :empty
- :enabled
- :first
- :first-child
- :first-of-type
- :fullscreen
- :focus
- :hover
- :indeterminate
- :in-range
- :invalid
- :lang()
- :last-child
- :last-of-type
- :left
- :link
- :not()
- :nth-child()
- :nth-last-child()
- :nth-last-of-type()
- :nth-of-type()
- :only-child
- :only-of-type
- :optional
- :out-of-range
- :read-only
- :read-write
- :required
- :right
- :root
- :scope:
- Target
- :valid
- :visited

Validador CSS

- Para saber sobre eventuais erros em folhas CSS, a W3c criou uma página que retorna um relatório sobre a análise de uma página.

- **Link da ferramenta**
  - http://jigsaw.w3.org/css-validator

Exercício (em sala)

- Você deve criar um *layout* utilizando HTML / CSS o mais fiel possível ao *próximo layout*.
  - **Não deve-se utilizar tabela na criação do layout**
  - Não deve-se utilizar *tags* que atualmente estejam descontinuadas
  - Os elementos principais do layout devem possuir **borda e fundo** para destacarem dos demais
- Ao fim da aula apresentamos os resultados