Design Web e Arquitetura da Informação

Prof. Romerito Campos

Plano de Aula

- **Objetivo:** Compreender o uso da propriedade `position`
- **Conteúdos**:
  - Position: definição
  - position: static, absolute, relative, sticky e fixed

position

position

- A propriedade `position` indica como podemos posicionar um elemento para sua renderização.
- O valor padrão desta propriedade é `position: static`.
  - Não é necessário indicar este valor;
  - Todos os elementos por definição já vêm com esta propriedade definida para `static`.
- Há 5 positions: `static`, `relative`, `absolute`, `fixed` e `sticky`.

position

- Um conceito importante sobre position é a ideia de **Containing Block**.
- Basicamente, ele indica qual é o box que contém um elemento.
- Por exemplo, a tag `<html>` é o **containing block** da tag `<body>`.
- Ou seja, o **containing block** é o elemento pai em relação aos elementos filhos.

Position

- Exemplo de Containing Block

```
<!-- trecho de uma página -->
<body>
    <div class="container">
    </div>
</body>
```

- A tag `<body>` é o **containing block** do elemento `<div>`.

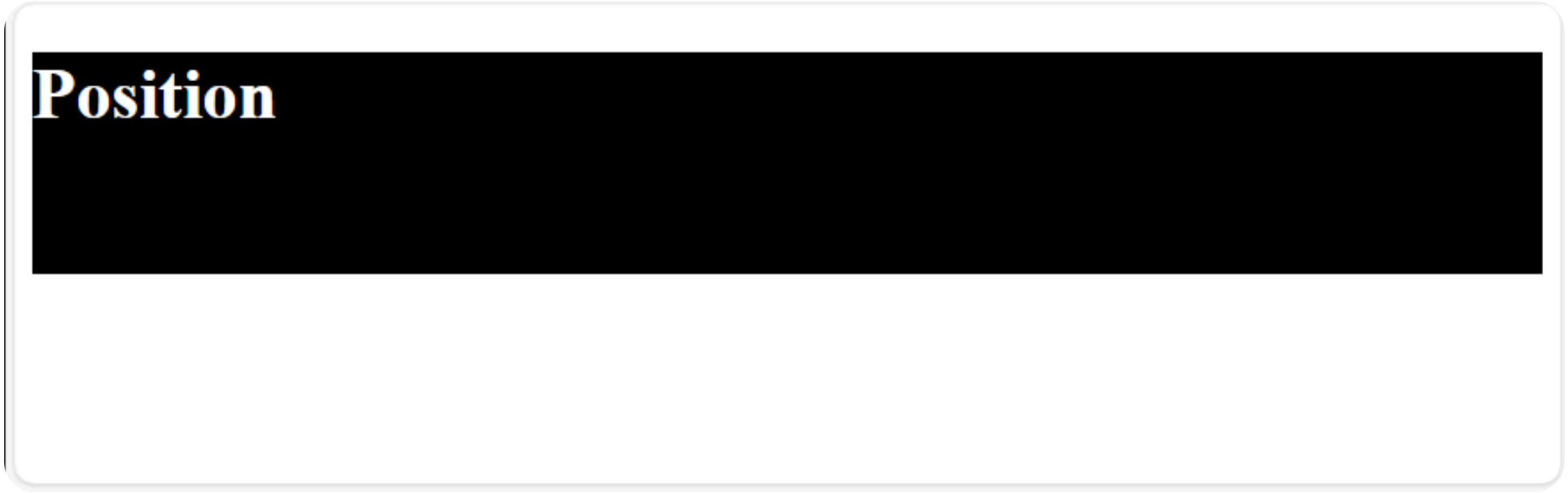
Position

- É importante considerar o **Containing block** para saber onde o elemento começa a ser desenhando.
- Vimos que os elementos são desenhados na tela considerando o **fluxo normal de renderização**.
  - Em idiomas cuja escrita é da esquerda para direita: português, inglês entre outros;
  - Portanto, o fluxo normal é o mesmo da escrita.
- Além disso, considere os níveis de `block` ou `inline`.

Position

- Por que preciso saber sobre *containing block* e nível de bloco|inline?
- Para alterar o posisionamento de um elemento é necessário ter noção de onde ele inicia o seu desenho.
- Veja a imagem no próximo slide:

- Vejamos este exemplo:



- Ele contém uma `div` com um `h1` dentro conforme o código a seguir.

- Código que produziu a imagem anterior

```
<body>
    <div>
        <h1>Position</h1>
    </div>
</body>
```

- O **containing block** da `div` é o `body`.
- Ele está no fluxo normal de renderização: esquerda para direita.
- O ponto inicial da div é `top: 0` e `left: 0`.
    - canto superior esquerdo do `body`

- Essa noçãoé importante porque a propriedade `position` se relaciona com as properidades `left`, `top`, `bottom` e `right`.
- `top: 0` e `left:0` indica o canto superior esquerdo
- `top: 0` e `right: 0` indica o canto superior direito.
- Em essência utilizaremos `position` e essas propriedades citadas para:

Retirar os elementos do fluxo normal de renderização

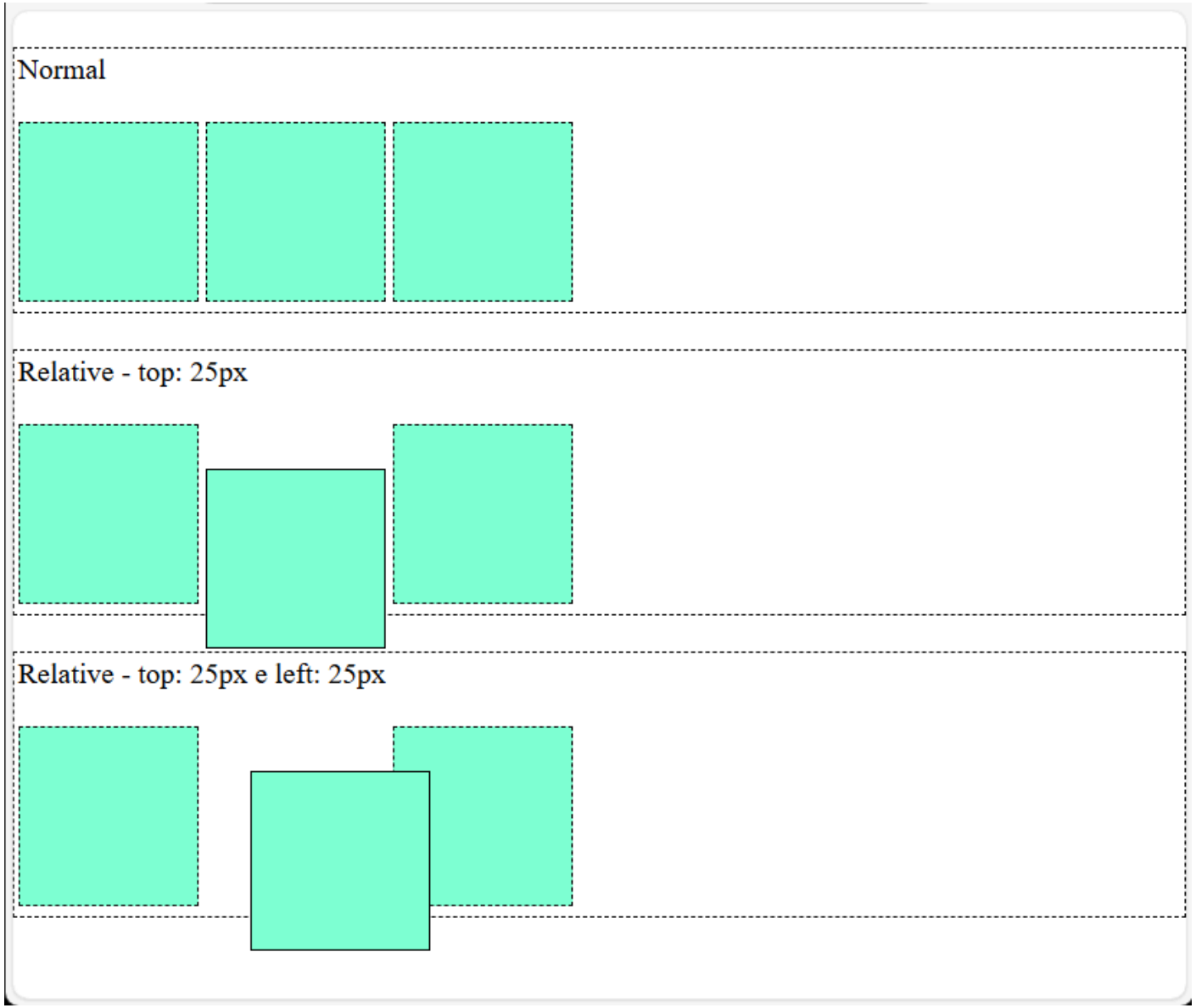
position: relative

position: relative

- O posicionamento relativo permite que se retire o elemento do fluxo normal em relação ao seu posicionamento original.
- Ao sair do local que seria sua posição original, o espaço que o elemento deveria ocupar é mantido.
- Vejamos um exemplo para entender essas definições: ex02.html

- Na imagem temos:
  - Fluxo Normal;
  - `position: relative` com deslocamento do elemento a partir do topo (`top: 25px`)
  - `position: relative` com deslocamento no topo e na esquerda:

```
top: 25px;
left:25px
```



- No primeiro conjunto de quadrados não temos alteração no fluxo normal de renderização.
- No segundo caso, perceba que o elmento foi deslocado para abaixo.
  - Ele saí de sua posição original;
  - Ultrapassa o limite da `div` que o contém;
  - Mas seu espaço original é mantido.
- No terceiro caso, acontece o mesmo que no caso dois. Além disso, temos um deslocamento da esquerda para direita.
  - Observe que o elemento sobrepós a terceira caixinha ao ser deslocamento.

- Este é o comportamento relativo.
- Segue o código que modifica os elementos apresentados.

```
.top {
    position: relative;
    top: 25px;
    border: 1px solid;
}
```

```
.left {
    position: relative;
    left: 25px;
    border: 1px solid;
}
```

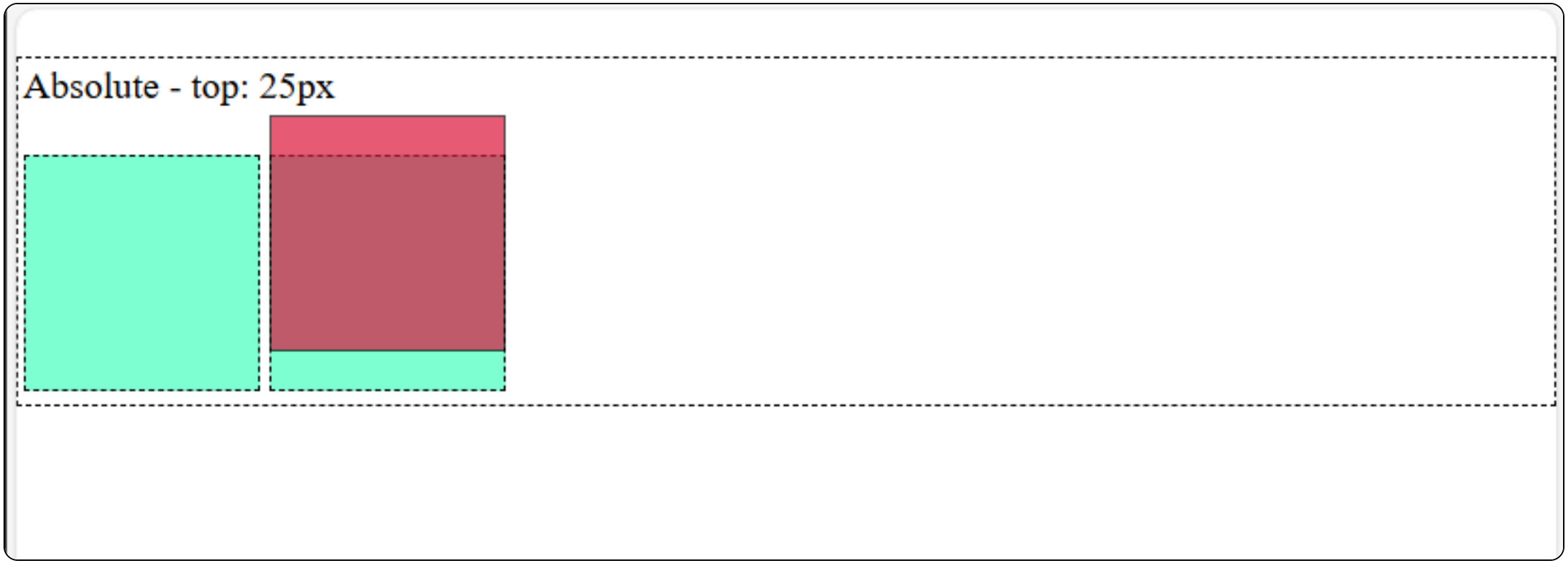
- Como desafio, crie o exemplo completo.

position: absolute

position: absolute

- O posicionamento absoluto indica o local em específico onde o elemento é desenhado.
- Além disso, ele não preserva o espaço original do elemento como acontece com `position: relative`.
- Com o `position: absolute` é muito fácil compreender a ideia de **containing block** apresentada no início.
- Observe a imagem a seguir:

- Exemplo de `position: absolute`:



- Vejamos os detalhes.

- É notável que uma caixa esta sobrepondo a outra.
- No caso, a caixa com tom avermelhado está deslocada de sua posição.
- Além disso, o elemento seguinte que é uma caixa ocupou seu lugar.
- Neste caso, o deslocamento aplicado (`top: 25px`) não funciona da mesma maneira que o `position: relative`.
- Observe que não definimos `left`.
- O position absolute vai usar o **containing block** `<body>` ao invés de usar a `div`.

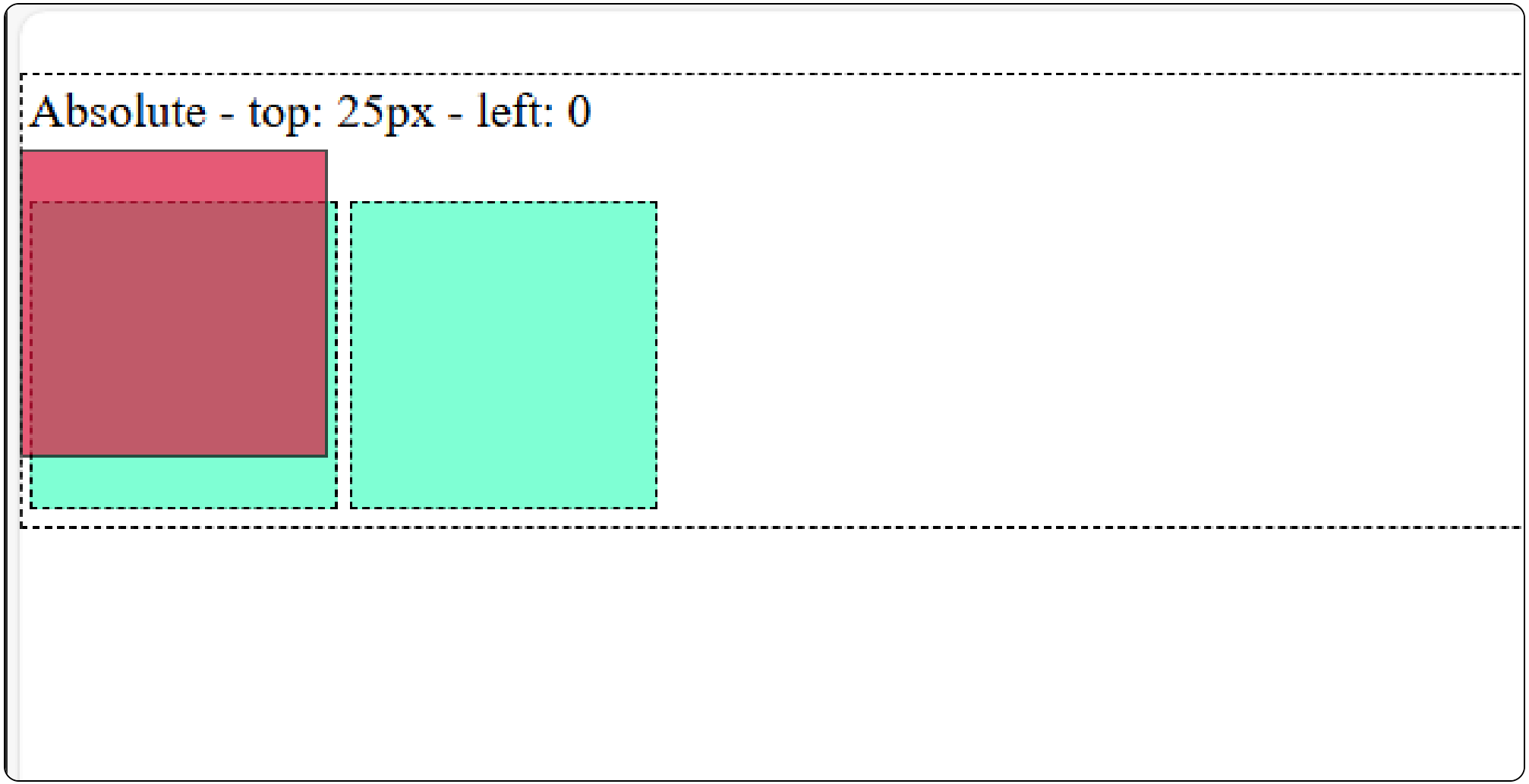
- Portanto, o elemento considera o `top: 0` como sendo o limite do `body`. Ele não considera a `div`.
- A posição em relação a `left` permanece a posição que deveria ser a original no fluxo normal. Veja a regra CSS:

```css
.top {
    position: absolute;
    top: 25px;
    border: 1px solid;
    background-color: crimson;
    opacity: 70%;
}
```

- Agora considere que vamos aplicar a seguinte regra CSS:

```css
.top {
    position: absolute;
    top: 25px;
    left: 0px;
    border: 1px solid;
    background-color: crimson;
    opacity: 70%;
}
```

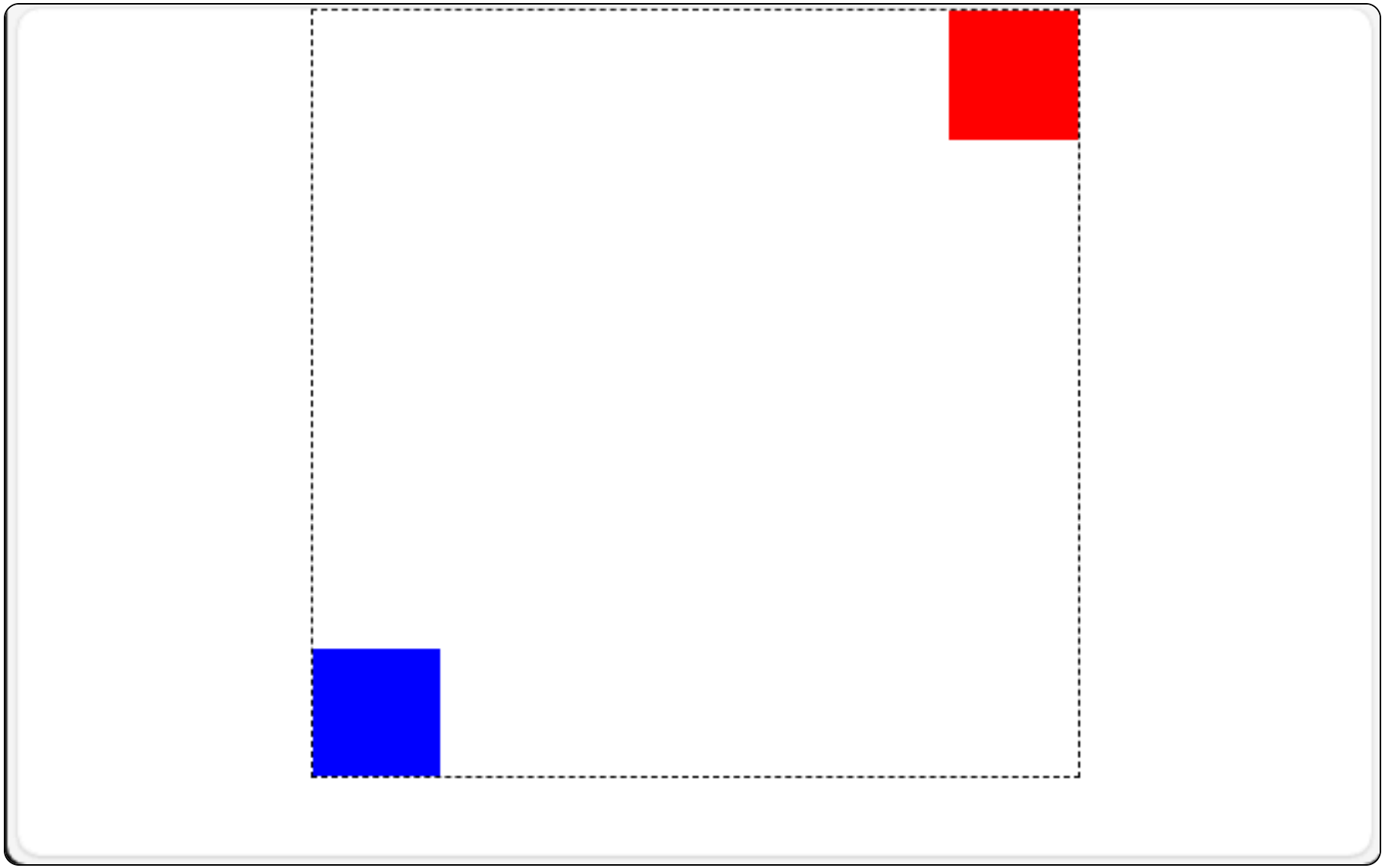
- Mantenha em mente que o **containing block** é o `body`
- O que será que vai acontecer? Vejamos no slide seguinte.


Absolute - top: 25px - left: 0

- Na imagem do slide anterior, temos o resultado da aplicação de `position: absolute` junto com `top` e `left`.
- O elemento será posicionado a partir da esquerda (`left`) da página na posição correspondende a 0.
  - Isso corresponde a ficar na borda da página.
- Com relação ao topo (`top`), mantém-se o distanciamento da borda superior da página.
- Isso acontece porque o **containing block** padrão para `position: absolute` é o viewport (dimensão da página).
- Veja o exemplo completo aqui

position: relative e position: absolute

- Caso seja necessário alterar o **containing block** para um elemento com `position: absolute`, podemos relacioná-lo com o `position: relative`.
- O próximo exemplo explora a relação entre uma `div` (`position: relative`) com dois elementos filhos (`position: absolute`)
- Veja o código-fonte aqui



- O código HTML é o seguinte:

```
<body>
    <div class="container">
        <div class="caixa top-right"></div>
        <div class="caixa bottom-left"></div>
    </div>
</body>
```

- Neste exemplo temos:
  - `.container`: `position: relative`
  - `.caixa`: `position: absolute`

- Além disso, temos também as classes:

```css
.top-right {
    top: 0;
    right: 0;
    background-color: red;
}
```

```css
.bottom-left{
    bottom: 0;
    left: 0;
    background-color: blue;
}
```

- Observe que aplicamos as propriedades para determinar o posicionamento absoluto para as caixas coloridas.
- A posição `top:0` para a caixa vermelha considera a `div`. Neste caso, não considera a página (`body`) como sendo seu **containing block**.
- Ou seja a referência para o absolute é o elemento pai da caixa.

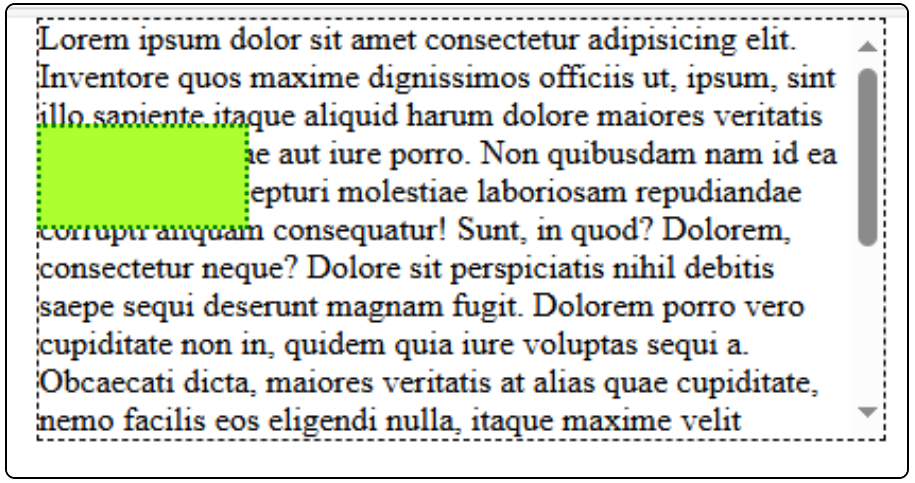
position: fixed

position: fixed

- O posicionamento fixado é semelhante ao absoluto em alguns aspectos:
  - O elemento é retirado do fluxo normal;
  - Seu espaço original é ocupado;
- A diferença crucial é a seguinte: dada uma página que apresenta rolagem na horizontal, o elemento fixado ficará sempre no mesmo local independentemente da movimentação da página.

position: fixed

- Além das diferenças anteriores, o **containing block** para os elementos com `position: absolute` é o `viewport` (a tela inteira).
- Vejamos o exemplo a seguir:



- O elemento de cor verde está fixado
- A `div` com borda tracejada possui conteúdo maior que sua área (**overflow**)
- Logo, podemos controlar esse overflow com rolagem
- A medida que a houver rolagem de página, o elemento verde permanece fixado.

- O código necessário para este exemplo ilustra a ideia do **containing block**:

```css
.container {
    height: 200px;
    width: 400px;
    margin: 0 auto;
    border: 1px dashed black;
    overflow-y: scroll;
}
```

```css
.fixed {
    position: fixed;
    top: 50px;
    right: calc(50% + 100px);
    height: 50px;
    width: 100px;
    border: 2px dotted green;
    background-color: greenyellow;
}
```

- Veja o código ex06.html e o vídeo do exemplo aqui

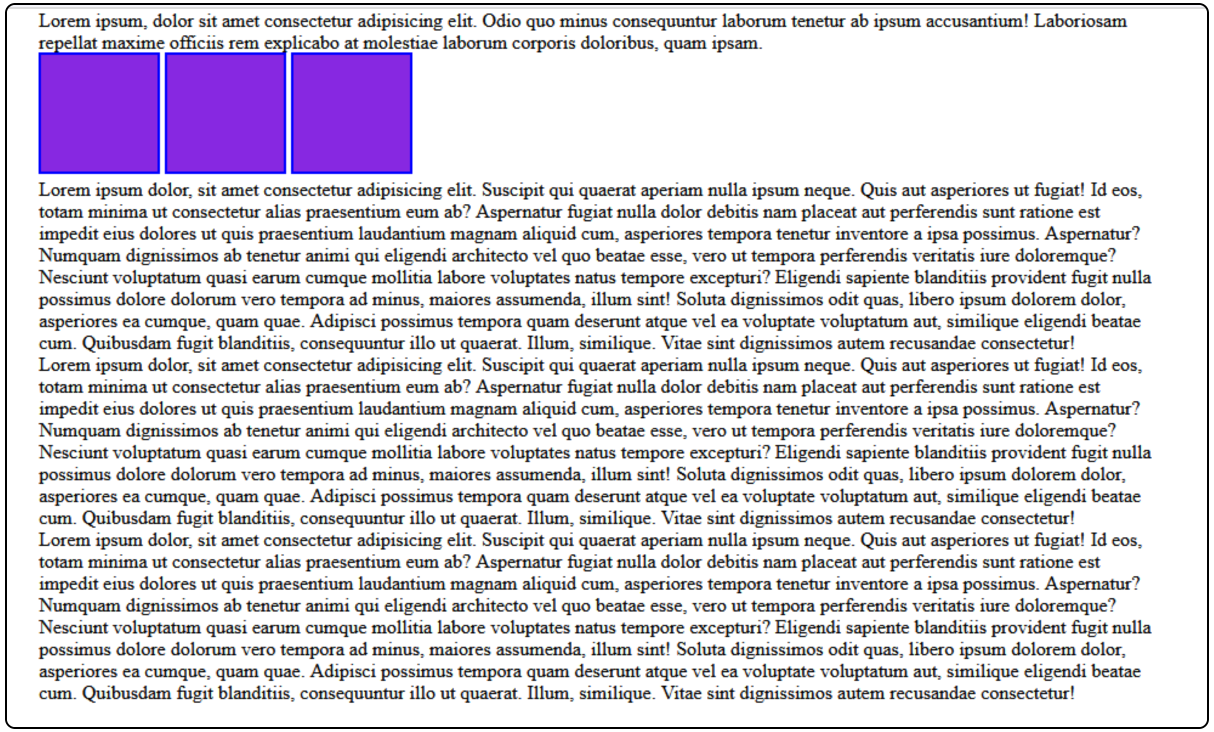
position: sticky

position: sticky

- Este é o posicionamento que geralmente causa bastante dúvida.
- O posisionamento adesivo funciona como um posicionamento relativo e muda para fixo com base nas propriedades `top`, `left`, `right`, `bottom`.
- Para entender o posicionamento sticky, precisamos compreender que temos:
  - um **sticky item** (item que será o adesivo)
  - um **sticky container** (item que é a referência para o adesivo)

position: sticky

- Considere o exemplo ex07.html. Ele produz o seguinte resultado:



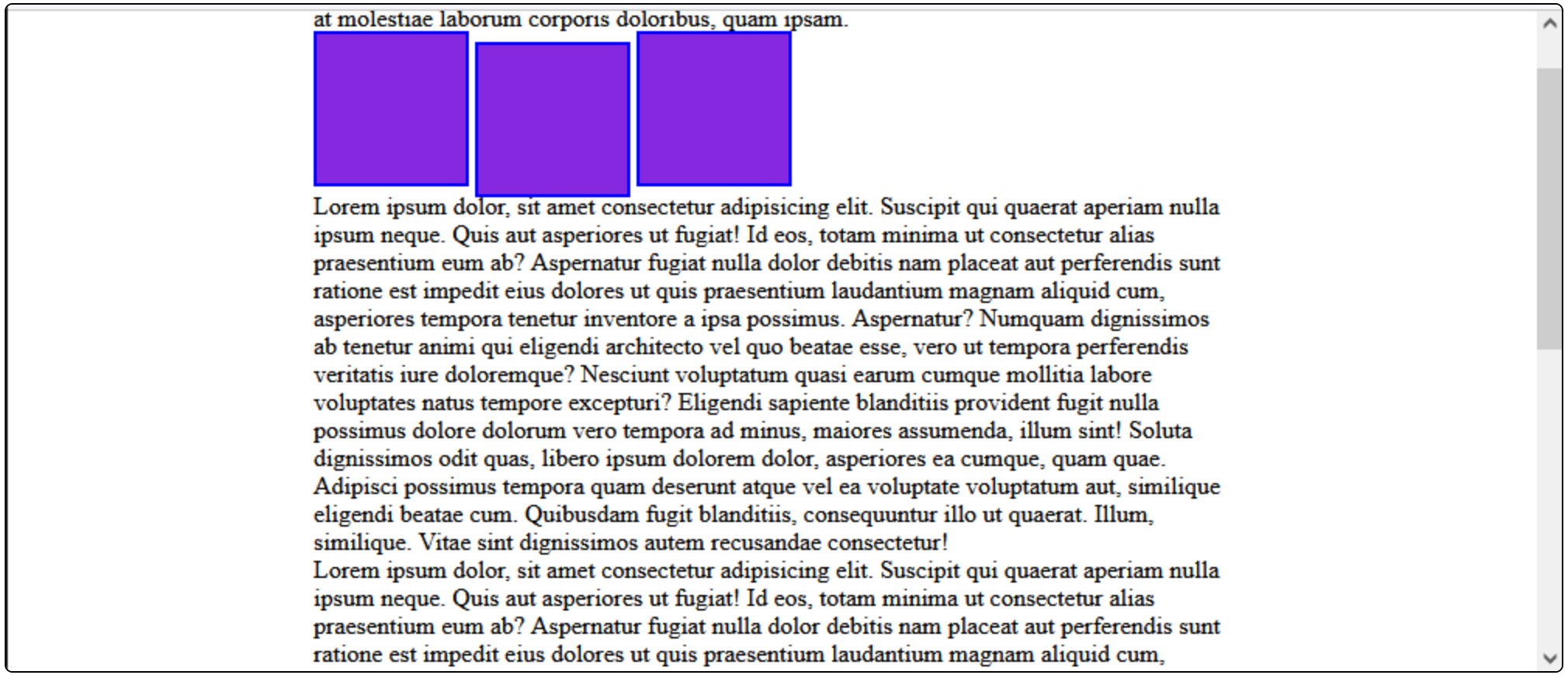
position: sticky

- A segunda caixa colorida está com `position: sticky` e vai se comportar como adesivo quando:
  - A página possuir barra de rolagem;
  - O container sticky atingir `top: 20px;`.

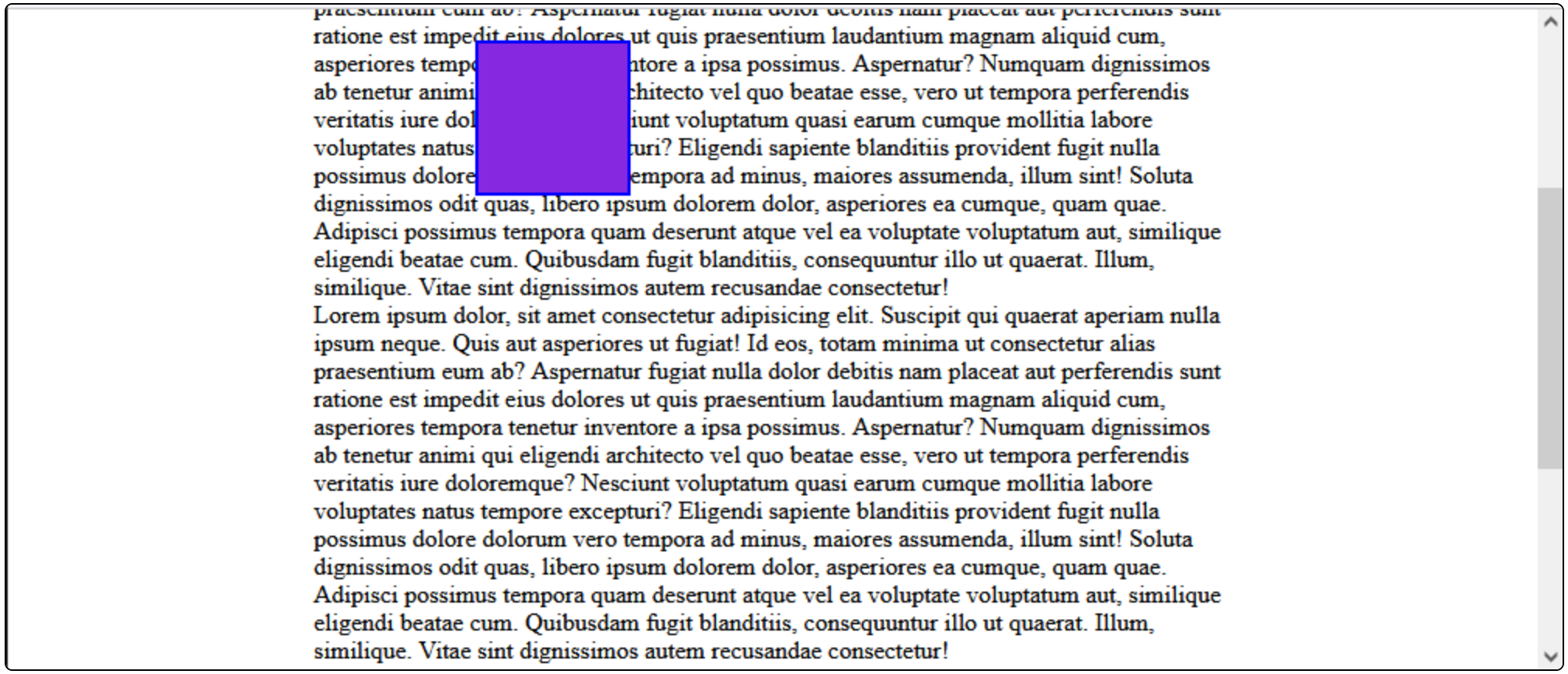
```css
.box {
    width: 100px;
    height: 100px;
    border: 2px solid blue;
    background-color: blueviolet;
    display: inline-block;
}
```

```css
.sticky {
    position: sticky;
    top: 20px
}
```

- A imagem abaixo mostra quando o container sticky (body neste exemplo) atingiu `top: 20px`



- Na imagem anterior, observe que a segunda caixa colorida começa a se deslocar. Isso signfica que ao rolar a página, a posição `top:20px` foi atingida. Ou seja, a borda superior da página está na posição `top: 20px`
- A partir daí, o elemento que se deslocou ficará como um adesivo a uma distância de 20px da borda superior.
- Veja a imagem no próximo slide.



- Veja aqui um vídeo ilustrando este comportamento;

Referências

- Explicação - Sticky