

Objetivos:

- I. Elementos block box e inline box;
- II. Box Model;
- III. Propriedade CSS display;
- IV. Flexbox layout.

I. Elementos block box e inline box

No CSS existem dois tipos de blocos, a diferença entre eles está apenas no comportamento deles no fluxo da página e a sua relação com outros blocos na página:

- Block box (caixa): blocos que formam boxes (caixas), isto é, ocupam 100% da área horizontal disponível para ele, e, desta forma, não aceitam outros elementos à sua esquerda e direita. Exemplos de elementos que criam blocos são `<p>`, `<div>` e `<h1>`. Sobre as propriedades:
 - `width` e `height` são respeitadas em elementos que formam bloco;
 - `padding`, `margin` e `border` são computadas fazendo com que a largura e altura do componente ultrapasse 100%.
- Inline box (na linha – no fluxo normal): ocupam apenas a sua própria área e, desta forma, aceitam outros elementos à sua esquerda e direita. Exemplos de elementos inline são `<a>` e ``. Sobre as propriedades:
 - `width` e `height` não são respeitadas num elemento inline. No exemplo a seguir veja que o elemento `` não recebeu 100px de largura e altura;
 - `padding`, `margin` e `border` são aplicadas, mas `não` fazem com que outros elementos inline sejam deslocados na `vertical`. Porém os elementos inline serão deslocados na `horizontal`. No exemplo a seguir veja que o elemento `` não possui margem vertical.

Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      span {
        margin: 20px;
        padding: 20px;
        width: 100px;
        height: 100px;
        background-color: yellow;
        border: 2px solid blue;
        /* display: block; */
      }
    </style>
  </head>
  <body>
    <div>
      <span>Exemplo</span>
    </div>
  </body>
</html>
```

```

    }
  </style>
</head>
<body>
  <p>
    As propriedades margin (margem externa), padding (margin interna) e border são
    respeitadas na horizontal nos elementos block inline.
  </p>
  <p>
    Nos elementos block inline como <span>span</span> as propriedades width e
    height não são respeitadas.
  </p>
  <p>
    Veja que o resultado não possui 100px de largura e altura e não possui
    margem vertical.
  </p>
</body>
</html>

```

Resultado no navegador

As propriedades **margin** (margem externa), **padding** (margin interna) e **border** são respeitadas na horizontal nos elementos **block** **inline**.

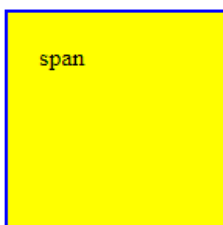
Nos elementos **block** **inline** como span as propriedades **width** e **height** não são respeitadas.

Veja que o resultado não possui 100px de largura e altura e não possui margem vertical.

O comportamento dos elementos **inline** pode ser alterado incluindo a propriedade CSS **display: block**. Para testar retire o comentário da propriedade **display: block** no exemplo anterior. Veja que o elemento **** passará a ter 100px de largura e altura, e as propriedades **margin**, **padding** e **border** passarão ser respeitadas na vertical. Além disso, os elementos que formam bloco não aceitam outros elementos à esquerda e direita, por este motivo o elemento **** foi renderizado numa linha separada no exemplo a seguir.

As propriedades **margin** (margem externa), **padding** (margin interna) e **border** são respeitadas na horizontal nos elementos **block** **inline**.

Nos elementos **block** **inline** como



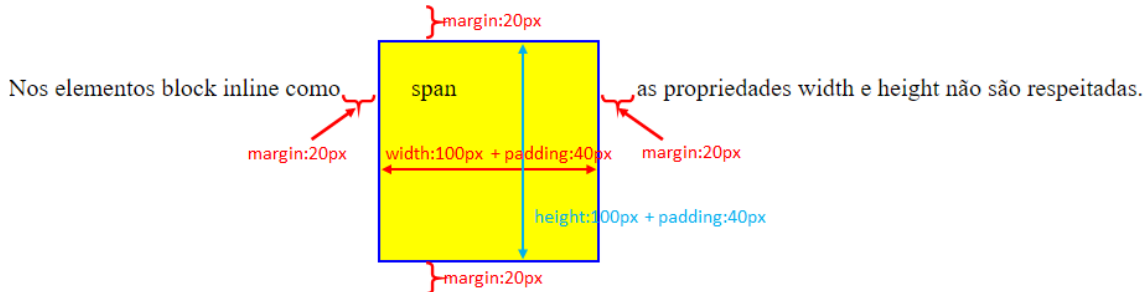
as propriedades **width** e **height** não são respeitadas.

Veja que o resultado não possui 100px de largura e altura e não possui margem vertical.

O valor **inline-block** da propriedade **display** é um meio termo entre **inline** e **block**. Ela faz com que a propriedades **width**, **height**, **padding**, **margin** e **border** sejam respeitadas, a diferença é que ela mantém o

fluxo, não causando quebra de linhas à esquerda e direita. No exemplo a seguir usou-se a propriedade **display: inline-block**.

As propriedades margem (margem externa), padding (margin interna) e border são respeitadas na horizontal nos elementos block inline.



Veja que o resultado não possui 100px de largura e altura e não possui margem vertical.

O valor **none** da propriedade **display** desativa a exibição do elemento, ou seja, o elemento não ocupa espaço no layout. No exemplo a seguir usou-se a propriedade **display: none**.

As propriedades margem (margem externa), padding (margin interna) e border são respeitadas na horizontal nos elementos block inline.

Nos elementos block inline como as propriedades width e height não são respeitadas.

Veja que o resultado não possui 100px de largura e altura e não possui margem vertical.

II. Box Model

O box model completo se aplica aos elementos block box, os elementos inline box usam apenas parte do comportamento definido no box model. O modelo define como as diferentes partes de um box (caixa) - margem, borda, preenchimento (padding) e conteúdo - trabalham juntas para criar um box na página. A figura a seguir mostra as partes do box model:



Fonte: https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/The_box_model#parts_of_a_box

- Content box: área onde o conteúdo é exibido, que pode ser dimensionado usando as propriedades CSS **width** e **height**. No exemplo a seguir ambos os elementos possuem o content box de 200px de largura;
- Padding box (caixa de preenchimento): o preenchimento fica em torno do conteúdo como espaço em branco (como se fosse uma margem interna). O seu tamanho pode ser controlado usando a propriedade

CSS **padding**. No exemplo a seguir o segundo elemento possui **padding-left:20px**, fazendo com que esse valor seja computado na largura total do elemento;

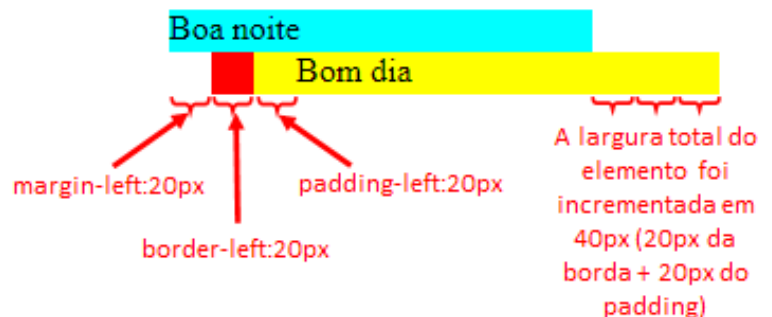
- Border box: a caixa de borda envolve o conteúdo e qualquer preenchimento. Seu tamanho e estilo podem ser controlados usando a propriedade CSS **border**. No exemplo a seguir o segundo elemento possui **border-left:20px**, fazendo com que esse valor seja computado na largura total do elemento;
- Margin box: a margem é a camada mais externa, envolvendo a borda como espaço em branco entre o próprio elemento e os demais. Seu tamanho pode ser controlado usando a propriedade CSS **margin**. No exemplo a seguir o segundo elemento possui **margin-left:20px**. A margem não é computada na largura final do elemento, porém ela fez a posição inicial do elemento ser deslocada em 20px para direita.

Por padrão, os navegadores usam o box model, onde a borda e o padding são computados nas dimensões finais do elemento na página. Porém, podemos usar a propriedade CSS **box-sizing:border-box** para dizer ao navegador para considerar a borda e padding na largura sugerida para o elemento. No exemplo a seguir retire o comentário de **box-sizing:border-box** e veja que o segundo elemento possuirá 200px de largura. O deslocamento (margin) continua existindo, pois a margem não faz parte da dimensão do elemento.

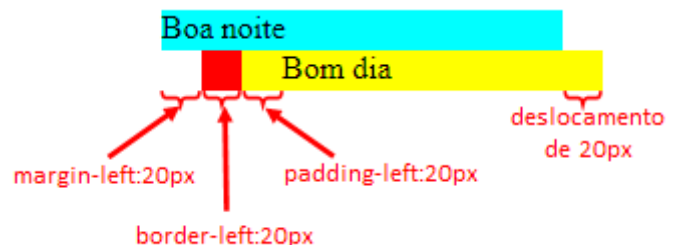
Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        width: 200px;
        height: 20px;
        background-color: cyan;
      }
      #dois {
        width: 200px;
        height: 20px;
        margin-left: 20px;
        padding-left: 20px;
        background-color: yellow;
        border-left: 20px solid red;
        /* box-sizing: border-box; */
      }
    </style>
  </head>
  <body>
    <div id="um">Boa noite</div>
    <div id="dois">Bom dia</div>
```

Resultado no navegador



Resultado no navegador ao utilizar a propriedade **box-sizing:border-box**:



```
</body>
</html>
```

Para mais detalhes acesse https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/The_box_model#what_is_the_css_box_model.

III. Propriedade CSS display

Formalmente, a propriedade **display** define os tipos de exibição internos e externos de um elemento. O tipo externo define a participação de um elemento no layout de fluxo, o tipo interno define o layout dos filhos.

Para mais detalhes acesse <https://developer.mozilla.org/pt-BR/docs/Web/CSS/display>.

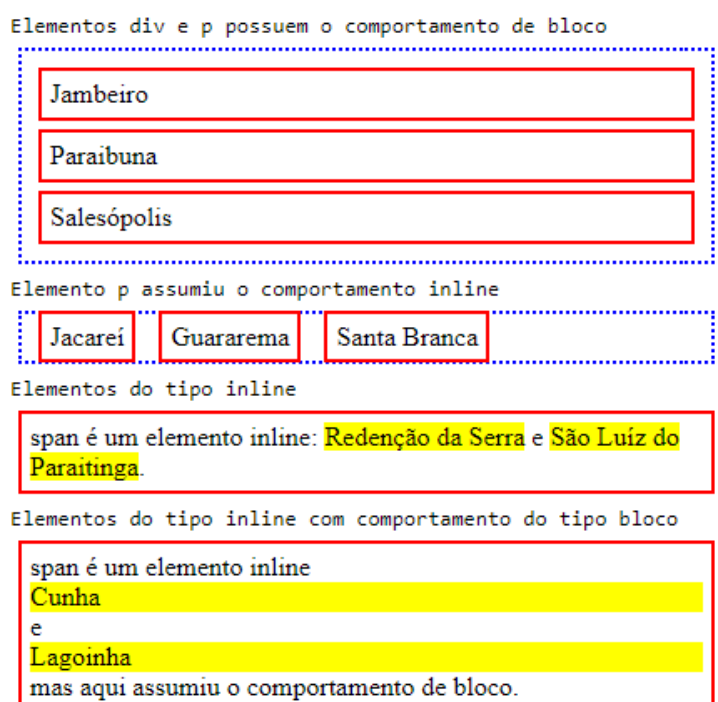
Exibição externa: definem a exibição do elemento em bloco ou em linha (inline):

- **block:** o elemento gera um bloco, gerando quebras de linha antes e depois do elemento no fluxo normal. O elemento `` não forma bloco veja no exemplo a seguir Redenção da Serra, mas ao receber a propriedade **display: block** o elemento recebe o comportamento de bloco, veja Cunha e Lagoinha;
- **inline:** o elemento se posiciona em linha, não gera quebras de linha antes ou depois de si mesmas. No fluxo normal, o próximo elemento estará na mesma linha se houver espaço; O elemento `<p>` forma bloco veja no exemplo a seguir Jambeiro, mas ao receber a propriedade **display: inline** o elemento recebe o comportamento em linha, veja que Jacaréí, Guararema e Santa Branca foram colocadas na mesma linha.

Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      p {
        border: 2px solid red;
        padding: 5px;
        margin: 5px;
      }
      div {
        border: 2px dotted blue;
        padding: 5px;
        margin: 5px;
      }
      span {
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <div>
      <p>Jambeiro</p>
      <p>Paraibuna</p>
      <p>Salesópolis</p>
    </div>
    <div>
      <p>Jacaréí</p>
      <p>Guararema</p>
      <p>Santa Branca</p>
    </div>
    <div>
      <p>span é um elemento inline: Redenção da Serra e São Luiz do Paraitinga.</p>
    </div>
    <div>
      <p>span é um elemento inline</p>
      <p>Cunha</p>
      <p>e</p>
      <p>Lagoinha</p>
      <p>mas aqui assumiu o comportamento de bloco.</p>
    </div>
  </body>
</html>
```

Resultado no navegador



```
.linha {
  display: inline;
}
.bloco {
  display: block;
}
</style>
</head>
<body>
  <code>Elementos div e p possuem o comportamento de bloco</code>
  <div>
    <p>Jambeiro</p>
    <p>Paraibuna</p>
    <p>Salesópolis</p>
  </div>
  <code>Elemento p assumiu o comportamento inline</code>
  <div>
    <p class="linha">Jacareí</p>
    <p class="linha">Guararema</p>
    <p class="linha">Santa Branca</p>
  </div>
  <code>Elementos do tipo inline</code>
  <p>
    span é um elemento inline: <span>Redenção da Serra</span> e <span>São Luíz do
    Paraitinga</span>.
  </p>
  <code>Elementos do tipo inline com comportamento do tipo bloco</code>
  <p>
    span é um elemento inline <span class="bloco">Cunha</span> e <span
    class="bloco">Lagoinha</span> mas aqui assumiu o comportamento de bloco.
  </p>
</body>
</html>
```

Exibição interna: especifica o layout dos filhos, que podem ser layout de fluxo, grid ou flex. Aqui abordaremos apenas o layout flex, no próximo item. Para mais detalhes sobre o layout grid acesse https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout.

IV. Flexbox layout

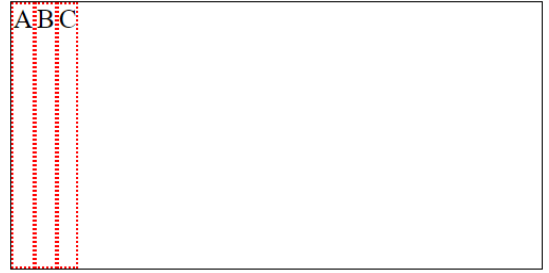
O Flexbox ajuda a construir layouts responsivos. Para criar um layout responsivo precisamos definir um elemento pai com a propriedade `display: flex`, desta forma, os elementos filhos serão flexíveis.

Se definirmos `display: flex` em um elemento, o tipo de exibição externa é `block`, mas o tipo de exibição interna é alterado para `flex`. Quaisquer filhos diretos desta caixa se tornarão itens `flex` e serão dispostos de acordo com as regras definidas na especificação do Flexbox, assim como é mostrado no exemplo a seguir.

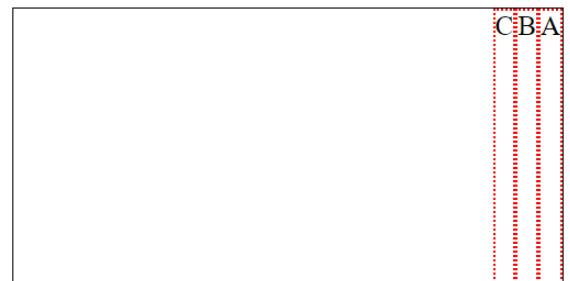
Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        display: flex;
        border: 1px solid black;
        height: 200px;
        width: 400px;
      }
      #um > div {
        border: 2px dotted red;
        font-size: 20px;
      }
    </style>
  </head>
  <body>
    <div id="um">
      <div>A</div>
      <div>B</div>
      <div>C</div>
    </div>
  </body>
</html>
```

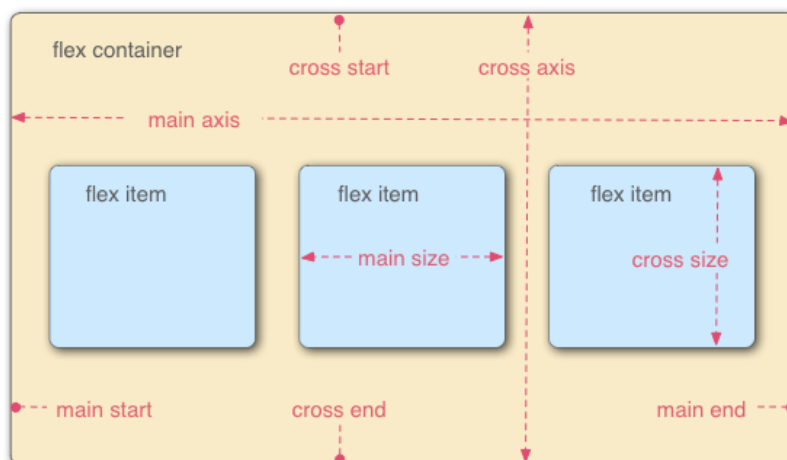
Resultado no navegador



Resultado no navegador usando:
`flex-direction: row-reverse`



Quando os elementos são definidos como caixas flexíveis (flex boxes), eles são dispostos ao longo de dois eixos (veja a figura a seguir):



Disposição dos elementos flexbox nos eixos principal e transversal

Fonte: https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Flexbox

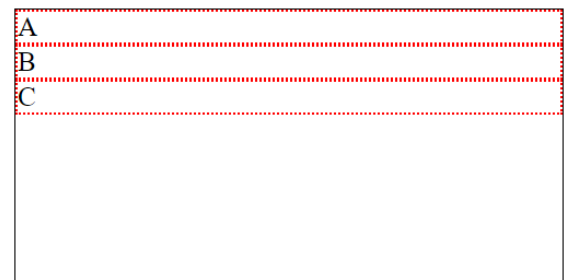
- **main axis**: é o eixo que corre na direção em que os flex items estão dispostos (por exemplo, as linhas ou colunas da página). O início e o fim do eixo são chamados de **main start** e **main end**;
- **cross axis**: é o eixo perpendicular que corre na direção em que os flex items são dispostos. O início e o fim deste eixo são chamados de **cross start** e **cross end**;
- **flex container**: o elemento pai que possui **display: flex** é chamado de flex container;
- **flex items**: os itens iniciados como flexible boxes dentro do flex container são chamados flex items.

Flexbox possui a propriedade **flex-direction** que especifica a direção do **eixo principal** - isto é, a direção que os filhos do flexbox estarão arranjados - que por padrão é row (linha). No exemplo a seguir adicionamos a propriedade **flex-direction: column** no elemento pai para mudar a direção do **eixo principal** de **row** para **column**, desta forma, os elementos filhos foram colocados na coluna.

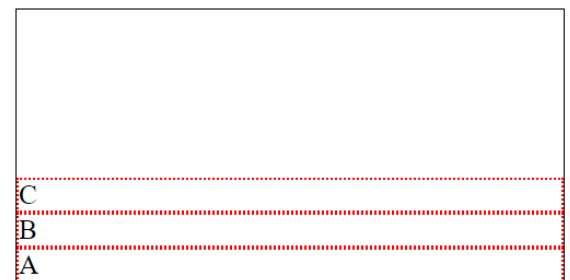
Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        display: flex;
        flex-direction: column;
        border: 1px solid black;
        height: 200px;
        width: 400px;
      }
      #um > div {
        border: 2px dotted red;
        font-size: 20px;
      }
    </style>
  </head>
  <body>
    <div id="um">
      <div>A</div>
      <div>B</div>
      <div>C</div>
    </div>
  </body>
</html>
```

Resultado no navegador



Resultado no navegador usando:
flex-direction: column-reverse



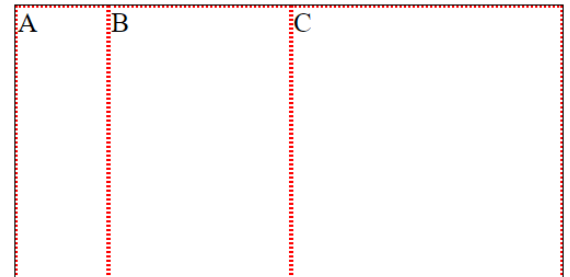
A propriedade **flex** pode ser usada nos elementos filhos para controlar a proporção de espaço que os flex items podem ocupar. Na prática podemos usar quaisquer valores na propriedade **flex**, no exemplo a seguir os

elementos A, B e C não possuem a mesma largura porque colocamos proporções diferentes: A ocupará $1/(1+2+3)$, B ocupará $2/(1+2+3)$ e C ocupará $3/(1+2+3)$. Cada elemento ocupará a mesma largura se definirmos **flex:1** em todos eles elementos filhos.

Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        display: flex;
        border: 1px solid black;
        height: 200px;
        width: 400px;
      }
      #um > div {
        border: 2px dotted red;
        font-size: 20px;
      }
    </style>
  </head>
  <body>
    <div id="um">
      <div style="flex:1">A</div>
      <div style="flex:2">B</div>
      <div style="flex:3">C</div>
    </div>
  </body>
</html>
```

Resultado no navegador



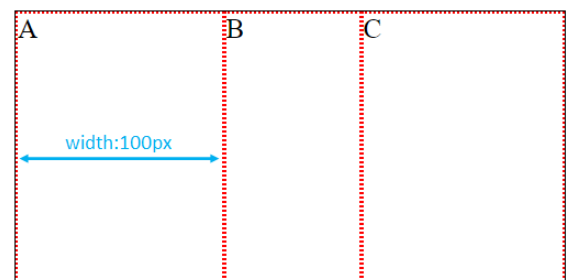
A propriedade **flex** aceita uma largura mínima para o elemento filho. No exemplo a seguir o elemento A ocupará o mínimo de **100px**. O restante da largura será dividido proporcionalmente entre os elementos B e C.

Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        display: flex;
        border: 1px solid black;

```

Resultado no navegador



```

        height: 200px;
        width: 400px;
    }
    #um > div {
        border: 2px dotted red;
        font-size: 20px;
    }
</style>
</head>
<body>
    <div id="um">
        <div style="flex:1 100px">A</div>
        <div style="flex:2">B</div>
        <div style="flex:3">C</div>
    </div>
</body>
</html>

```

As propriedades `align-items` e `justify-content` são usadas, respectivamente, para alinhar no eixo transversal e principal (neste exemplo, o eixo principal é row):

- `align-items`: pode receber os valores: `center`, `flex-start`, `flex-end` e `stretch` (valor padrão);
- `justify-content`: pode receber os valores: `flex-start` (valor padrão), `flex-end`, `center`, `space-around`, `space-between` (é similar ao `space-around`, exceto que ele não deixa nenhum espaço nas extremidades).

Código do arquivo index.html

```

<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta charset="utf-8" />
        <title>Exemplo</title>
        <style>
            #um {
                display: flex;
                align-items: center;
                justify-content: space-around;
                border: 1px solid black;
                height: 200px;
                width: 400px;
            }
            #um > div {
                border: 2px dotted red;
                font-size: 20px;
            }

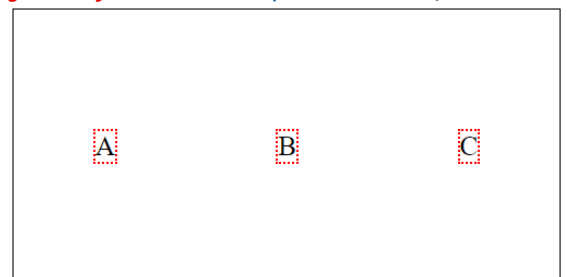
```

Usando:

```

align-items: center;
justify-content: space-around;

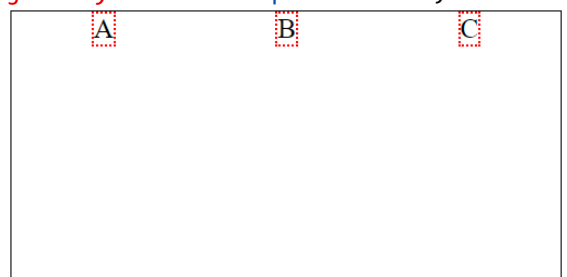
```



```

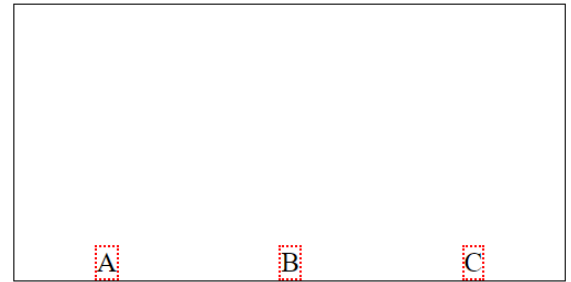
align-items: flex-start;
justify-content: space-around;

```

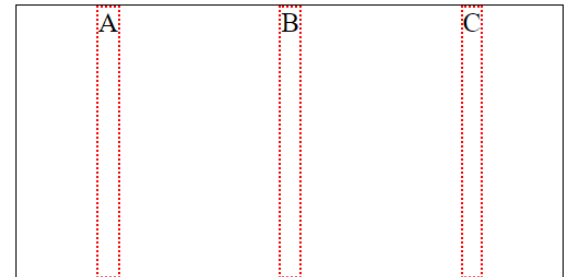


```
</style>
</head>
<body>
  <div id="um">
    <div>A</div>
    <div>B</div>
    <div>C</div>
  </div>
</body>
</html>
```

```
align-items: flex-end;
justify-content: space-around;
```



```
align-items: stretch;
justify-content: space-around;
```



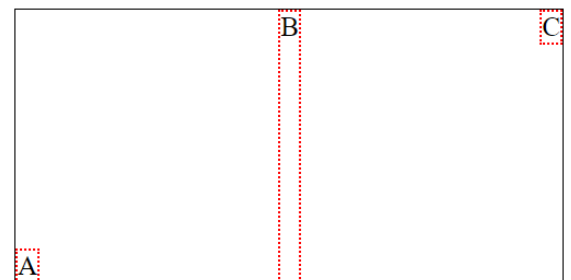
A propriedade `align-self` sobreescreve a propriedade `align-items` para elementos individuais. No exemplo a seguir os elementos filhos possuem alinhamentos distintos daquele especificado no elemento pai.

Observe que a propriedade `align-items` é definida no elemento pai e a propriedade `align-self` é definida nos elementos filhos – mas ambas as propriedades são aplicadas nos elementos filhos.

Código do arquivo index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <style>
      #um {
        display: flex;
        justify-content: space-between;
        border: 1px solid black;
        height: 200px;
        width: 400px;
      }
      #um > div {
        border: 2px dotted red;
        font-size: 20px;
      }
    </style>
  </head>
```

Resultado no navegador



```
<body>
  <div id="um">
    <div style="align-self: flex-end">A</div>
    <div style="align-self: stretch">B</div>
    <div style="align-self: flex-start">C</div>
  </div>
</body>
</html>
```

Para mais detalhes https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Flexbox.