# Exercício

Exame Época Normal 21/22



## Exercicio (Exame Época Normal 21/22)

```
<body>
      <div>
         <p> Client <span>Side<span>Web Technologies </span></span></p>
         <ul>
            <li class="item">HTML</li>
            <li>CSS</li>
            <li>Java<span>Script</span></li>
         </ul>
      </div>
</body>
```

# Exercicio (Exame Época Normal 21/22)

```
<body>
      <div>
         <p> Client <span>Side<span>Web Technologies </span></span></p>
         <ul>
            <li class="item">HTML</li>
            <li>CSS</li>
            <li>Java<span>Script</span></li>
         </ul>
      </div>
</body>
```



# Unidades CSS

# Unidades CSS

- Unidades Relativas
  - São baseadas no tamanho de outro elemento/referência
    - Não pode existir um espaço em branco entre o valor e a unidade
    - Se o valor for 0 a unidade pode ser omitida

*2em* ✓ *2 em* ✗

| *em* | *rem* | *%* |
|---|---|---|
| Relativa ao font-size definido para a fonte default: 16px | Relativa ao font-size definido para a fonte do root element <html> default: 16px | Relativa às dimensões do elemento pai |

| *vmin* | *vh* | *vw* | *vh* |
|---|---|---|---|
| Relativa a 1% da menor dimensão do viewport (mobile) | Relativa a 1% da maior dimensão do viewport (mobile) | Relativa a 1% da largura do viewport (mobile) | Relativa a 1% da altura do viewport (mobile) |



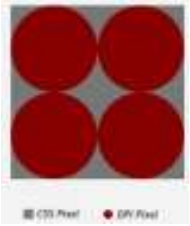
# Unidades CSS

- CSS pixel
  - É uma unidade abstrata, tudo o que é definido em ***px*** em CSS baseia-se no CSS pixel
    - exemplo: {font-size:16px;}

*The **px** unit is defined **to be small but visible, and such that a horizontal 1px wide line can be displayed with sharp edges**. What is sharp, small and visible **depends on the device** and the way it is used: do you hold it close to your eyes, like a mobile phone, at arms length, like a computer monitor, or somewhere in between, like an e-book reader?*

  - O CSS pixel pode ou não coincidir com os pixéis físicos (dpi)
    - ***Device pixel ratio***: informação de que o browser necessita para determinar quantos pixéis físicos são utilizados para desenhar um único CSS pixel.

| Modelo | Resolução Física (px) | CSS pixels | Device Pixel |
|---|---|---|---|
| iPhone 14 Pro | 1179 x 2556 | 393 x 852 | 3 |
| iPhone 14 Pro Max | 1290 x 2796 | 430 x 932 | 3 |



http://blog.popupdesign.com.br/desenvolvimento-responsivo-e-viewport/

# Box Model

Propriedades

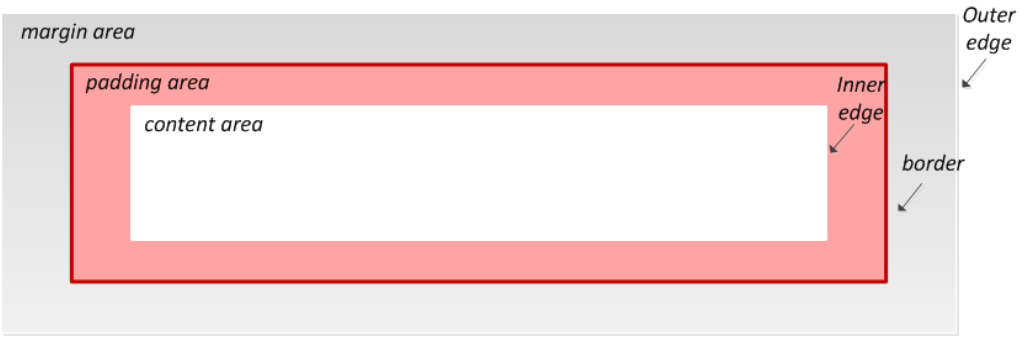


## Box Model

- Os elementos HTML *(inline/block)* são interpretados pelo *browser* como estando contidos em caixas rectangulares, às quais podem ser aplicadas propriedades.

```
        div{
            background-color: orange;
            color:white;
            font-size: 1.5em;
            width:300px;
            height:150px;
            border:darkgray 1px solid;
        }
    </style>
</head>
<body>
    <div> Block Element </div>
```

# Box Model




- *content area*
  - *área reservada ao conteúdo*
- *padding area*
  - área definida entre a área de conteúdo e o border **(opcional)**
- *border*
  - *linha que envolve a content area e a padding área* **(opcional)**
- *margin*
  - área adicionada no exterior do *border* **(opcional)**



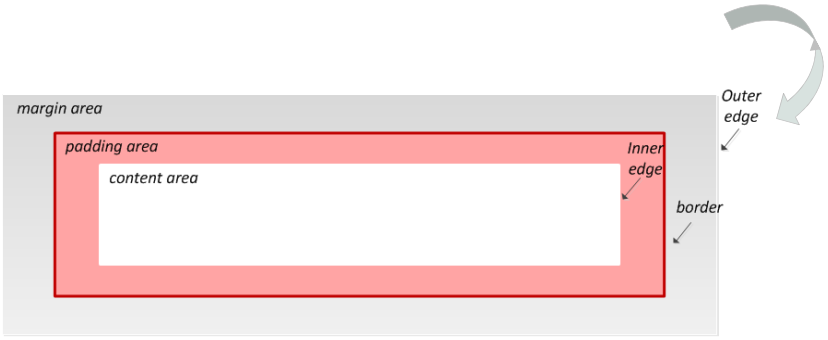
# CSS – Box Model

- Propriedades

| *Propriedade* | *Exemplo* | *Observações* |
|---|---|---|
| padding<br>*padding-top*<br>*padding-right*<br>*padding-left*<br>*padding-bottom* | p {padding:2em;} | Permite definir a *padding area*.<br>Valores: *length measurement, percentage, auto, inherit*. |
| margin<br>*margin-top*<br>*margin-right*<br>*margin-left*<br>*margin-bottom* | p {margin:2em;} | Permite definir a *margin area*.<br>Valores: *length measurement, percentage, auto, inherit*.<br><br>Nota importante: a inserção de valores negativos conduz geralmente à sobreposição dos elementos. |

# Box Model

- padding
  - seletor{padding:5px;} 1 valor (top/bottom/right/left)
  - seletor{padding:5px 10px;} 2 valores (top/bottom right/left)
  - seletor{padding:5px 10px 15px 5px;} 4 valores (top right bottom left)

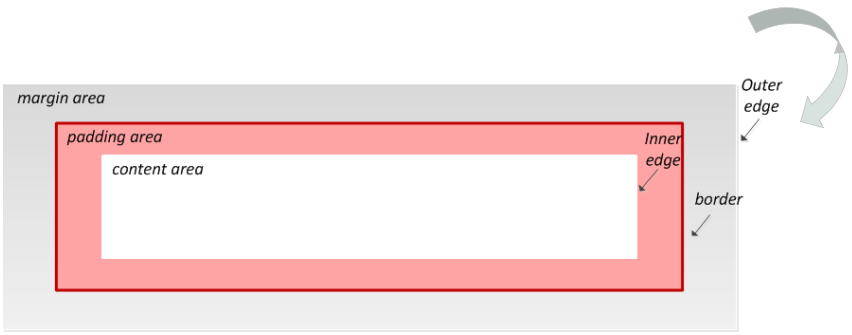



  - *padding-top; padding-right; padding-bottom; padding-left*



# Box Model

- margin
  - seletor{margin:5px;} 1 valor (top/bottom/right/left)
  - seletor{margin:5px 10px;} 2 valores (top/bottom right/left)
  - seletor{margin:5px 10px 15px 5px;} 4 valores (top right bottom left)




  - margin-*top;* margin-*right;* margin-*bottom;* margin-*left*

# Box Model

- *content area*
  - *width; height*
    - definem as dimensões da *content area*

```
        div{
            background-color: orange;
            color:white;
            font-size: 1.5em;
            width:300px;
            height:150px;
            border:darkgray 1px solid;
        }
    </style>
</head>
<body>
    <div> Block Element </div>
```




# Box Model

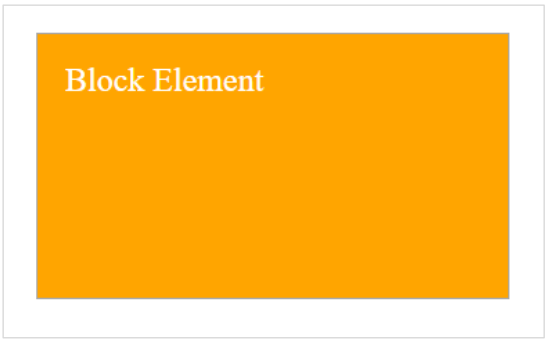
- padding area
  - *padding: top right bottom left;*
    - as dimensões globais resultam do somatório da área de conteúdo e de espaçamento interno

```
            div{
            background-color: orange;
            color:white;
            font-size: 1.5em;
            width:300px;
            height:150px;
            padding:20px;
            border:darkgray 1px solid;
        }
    </style>
</head>
<body>
    <div> Block Element </div>
```

# Box Model

- margin *area*
  - *margin: top right bottom left;*
    - estabelece as dimensões da *margin area* (área livre em redor do elemento)

```
        div{
            background-color: orange;
            color:white;
            font-size: 1.5em;
            width:300px;
            height:150px;
            padding:20px;
            margin:20px;
            border:darkgray 1px solid;
        }
    </style>
</head>
<body>
    <div> Block Element </div>
```

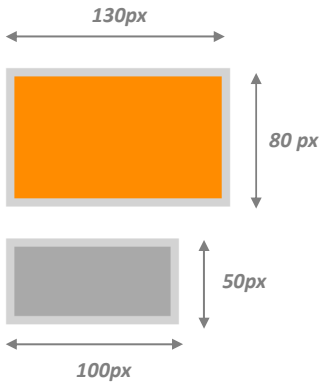





# *box model*

- Dimensões da área visível (***box-sizing***)
  - *content-box*
    - Valor por default, os valores de width/height referem-se à área de conteúdo
  - *border-box*
    - *content area + padding area + border*

```
#d1{width:100px;
    height:50px;
    background-color:darkorange;
    border: 5px lightgray solid;
    padding:10px; }


#d2{width:100px;
    height:50px;
    background-color:darkgray;
    border: 5px lightgray solid;
    padding:10px;
    box-sizing:border-box}
```



# Box Model

- *border*

| *Propriedade* | *Exemplo* | *Observações* |
|---|---|---|
| *border-style*<br>*border-top-style*<br>*border-right-style*<br>*border-left-style*<br>*border-bottom-style* | *p {border-style: solid;}* | Permite escolher o estilo do border.<br>**Valores:** *none*, dotted, dashed, solid, double, inset (sombra interior), outset (sombra exterior) …* |
| *border-width*<br>*border-top-width*<br>*border-right-width*<br>*border-left-width*<br>*border-bottom-width* | *p {border-style: solid; border-width: 4px;}* | Permite definir a largura do border.<br>**Valores:** *length units, thin, medium*, thick, inherit* |
| *border-color*<br>*border-top-color*<br>*border-right-color*<br>*border-left-color*<br>*border-bottom-color* | *p {border-style: solid; border-width: 4px; border-color: red;}* | Definição da cor do border.<br>**Valores:** *color name/RGB value, transparent, inherit* |
| *border*<br>*border-top*<br>*border-right*<br>*border-left*<br>*border-bottom* | *p{border: 4px solid red; }* | Propriedade agregada (width, style, color)<br>Mais frequentemente utilizada. |



# Box Model

- *overflow*

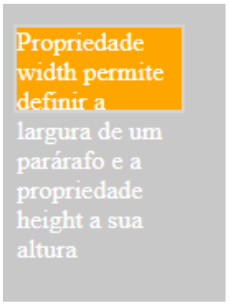
| *Propriedade* | *Exemplo* | *Observações* |
|---|---|---|
| overflow | p{overflow:scroll;} | Valores: *visible, hidden, scroll, …* |

```
        body{background-color:rgb(200,200,200)}
             p{
                height:50px;
                width:100px;
                background-color: orange;
                border:2px lightgray solid;
                color:white
             }
    </style>
</head>
<body>
    <p> Propriedade width permite definir a largura de um paráfo e a
        propriedade height a sua altura</p>
```

Propriedade width permite definir a largura de um paráfo e a propriedade height a sua altura

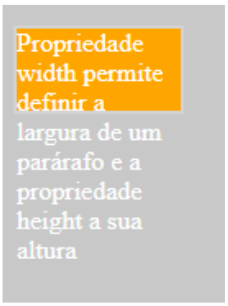
# CSS – Box Model

```
p{
  height:50px;
  width:100px;
  background-color: orange;
  border:2px lightgray solid;
  color:white;
  overflow:hidden}
```





```
p{
  height:50px;
  width:100px;
  background-color: orange;
  border:2px lightgray solid;
  color:white;
  overflow:scroll}
```







# *box model*

- Cantos arredondados
  - border-radius: valor;
    - O mesmo valor aplica-se a todos os cantos
  - border-radius: valor1 valor2;
    - valor1: canto superior direito/inferior esquerdo
    - valor2: canto inferior direito/superior esquerdo
  - border-radius: valor1 valor2 valor3 valor4;
    - valor1 canto superior esquerdo (sentido horário)
  - Propriedades individuais:
    - border-top-left-radius
    - border-top-right-radius
    - border-bottom-right-radius
    - border-bottom-left-radius

```
<!DOCTYPE html>
<html>
<head>
    <style>
    p{background-color:darkorange;
      color:white;
      width:200px;
      border:darkorange 2px solid;
      border-radius:4px;}
    </style>
</head>
<body>
    <p>BORDER RADIUS</p>
</body>
</html>
```

BORDER RADIUS

- Efeito Sombra (box)
  - {*box-shadow: h-shadow v-shadow blur (opt.) spread (opt.) color (opt.)  inset (opt.);*}

```
<!DOCTYPE html>
<html>
<head>
     <style>
        #d1 {width:100px;height:50px;
             background-color:darkorange;
             color:white;
             border:5px lightgray solid;
             padding:10px;
             box-shadow:lightgray 10px 10px 5px;}
    </style>
</head>
<body>
    <div id="d1">Box Shadow</div><br/>
</body>
</html
```




# Propriedades

*display*

# Block-level element

- Expande-se naturalmente para ocupar o seu *container*
  - A largura do elemento pode ser controlada (*width*)
  - Provoca uma quebra de linha
- Permite a definição de *margins* e/ou *padding*
- Se não for especificada a altura *(height)* o elemento expande-se para englobar os seus elementos filho
  - Este comportamento é alterado se o posicionamento dos filhos tiver sido alterado (*float/position*)
- A sua posição original é definida pelo fluxo do HTML
- Exemplos:
  - <p>, <h1>, , <form>, <ul>, <li>, …




# Block-level element

- *Exemplo:  expansão do elemento*

```
        div{ background-color:rgb(245, 245, 245);
        color:darkgray;
        padding-left:10px;
        border-top:1px lightgray solid;
        border-bottom:1px lightgray solid;
        padding-top: 15px;
        padding-bottom: 15px;}
    </style>
</head>
<body>

    <div id="div1">div1</div>
    <div id="div2">div2</div>
    <div id="div3">div3</div>
```

** ** é um ***block level element***

Caso não seja definida a largura, o elemento expande-se considerando a totalidade da largura do *viewport*

div1

div2

div3

# Block-level element

- *Exemplo:  propriedade width*

```
        div{ background-color:rgb(245, 245, 245);
        color:darkgray;
        padding-left:10px;
        border-top:1px lightgray solid;
        border-bottom:1px lightgray solid;
        padding-top: 15px;
        padding-bottom: 15px;
        width:200px}
    </style>
</head>
<body>

    <div id="div1">div1</div>
    <div id="div2">div2</div>
    <div id="div3">div3</div>
```

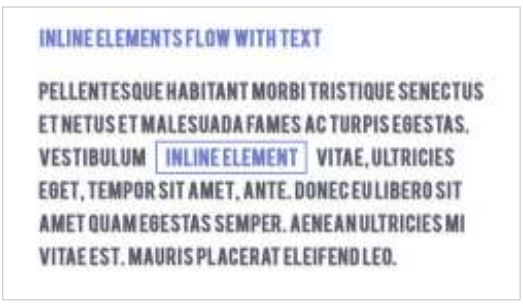
** ** é um ***block level element***

Caso a largura seja definida, o elemento não se expande assumindo a dimensão estabelecida.




# Inline-level Element

- Aplica-se a conteúdo textual
  - Pode ser percepcionado como uma caixa que se comporta como texto.
  - **Não se expande** à totalidade do espaço disponível (container/viewport)
- **Ignora** margens topo/fundo (*top/bottom*) mas permite a definição de *left/right margin* assim como a propriedade *padding*.
- **Ignora** as propriedades largura e altura (width/height).
  - Se for definido como um elemento *float (left/right)* torna-se automaticamente um *block-level element*
- Está sujeito a alinhamento vertical (*vertical-align*)
- Exemplos:
  - <a>, , …



# {display:***}

- *display*
  - propriedade muito importante nomeadamente na conceção de menus de navegação

| *Propriedade* | *Exemplo* | *Observações* |
|---|---|---|
| display | li {display:inline;} | Permite definir a disposição de elementos.<br>Valores: none; inline; block; inline-block; flex; … |



# {display:***}

- seletor{*display:**inline***}
  - Define o elemento como um ***inline element***
  - Muito útil para a implementação de menus horizontais *(horizontal navigation bar)*

```
      div{ background-color:rgb(245, 245, 245);
        color:darkgray;
        padding-left:10px;
        border-top:1px lightgray solid;
        border-bottom:1px lightgray solid;
        padding-top: 15px;
        padding-bottom: 15px;
        width:200px;
        }
     div{display:inline}
    </style>
</head>
<body>

    <div id="div1">div1</div>
    <div id="div2">div2</div>
    <div id="div3">div3</div>
```

# {display:***}

- seletor{display: **block**}
  - Define o elemento como um ***block element***
  - Útil para a implementação de menus verticais *(vertical navigation bar)*

```
    <style>
        a{
            font:Verdana;
            font-size: 1.1em;
            display:block;
            height:20px;
            width:150px;
            text-decoration: none;
            color:darkgray;
            padding-left:10px;
            border-top:1px lightgray solid;
            border-bottom:1px lightgray solid;
            border-radius: 1px;
            padding-top: 15px;
            padding-bottom: 15px;
            background-color: rgb(245, 245, 245);
        }
    </style>
    <title>Document</title>
</head>
<body>
    <a href="">Noticias</a>
    <a href="">Contactos</a>
    <a href="">Empresa</a>
    <a href="">Parcerias</a>
</body>
</html>
```

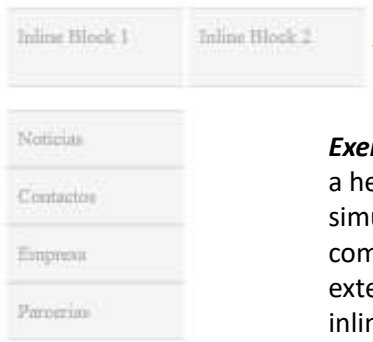
Notícias Contactos Empresa Parcerias

Notícias
Contactos
Empresa
Parcerias



# {display:***}

- seletor{display:**inline-block**}
  - O interior do elemento é considerado como um *block-level element*
  - O elemento é definido como um *inline-level element*
  - Mais versátil que *display:inline*




***Exemplo:*** alterar a height e em simultâneo ter o comportamento exterior de um inline element

```
        a{  font:Verdana;
            font-size: 1.1em;
            display:block;
            height:20px;
            width:150px;
            text-decoration: none;
            color:darkgray;
            padding-left:10px;
            border-top:1px lightgray solid;
            border-bottom:1px lightgray solid;
            border-radius: 1px;
            padding-top: 15px;
            padding-bottom: 15px;
            background-color: rgb(245, 245, 245);}
        a.c1{display:inline-block;
             height:40px;
             margin-bottom: 20px}
    </style>
</head>
<body>
    <a href="" class="c1">Inline Block 1</a>
    <a href="" class="c1">Inline Block 2</a>

    <a href="">Notícias</a>
    <a href="">Contactos</a>
```

- seletor{display:**none**}
  - O elemento não é visualizado **nem tem qualquer** influência no layout

```
        a.c1{display:inline-block;
             height:40px;
             margin-bottom: 20px}
        a.c2{display:none}
    </style>

</head>
<body>
    <a href="" class="c1 c2">Inline Block 1</a>
    <a href="" class="c1">Inline Block 2</a>
```


  - lista completa dos valores propriedade ***display***
    - http://www.w3schools.com/cssref/pr_class_display.asp



# Layouts

Tipos

# Layouts

- Existem diferentes tipos de *layouts*:
  - ***fluid layout/flexible layout***
    - As dimensões são estabelecidas de forma a existir um redimensionamento dos conteúdos quando as dimensões do *viewport* são alteradas
  - *fixed layout*
    - As dimensões são especificadas em ***px*** sem atender às dimensões do *viewport* ou do tamanho do texto



# *fluid layout/flexible layout*

- As colunas contidas na página ajustam-se ao espaço disponível na janela do browser
  - layout versátil que se ajusta a diferentes dimensões da janela do browser, e como tal pode ser um elemento chave do *responsive web design* (adaptar conteúdos a diferentes *viewports*)
  - As dimensões são definidas em % ou tiram partido de novas abordagens CSS
    - Tipo de layout que atualmente deve ser implementado!

| Vantagens | Desvantagens |
|---|---|
| Ajustam o conteúdo ao espaço de visualização disponível; | |
| Permite a correta visualização em maior variedade de dispositivos (*desktops, mobile, …*); | É menos previsível que os layouts fixos, nomeadamente para dimensões extremas dos browsers; |
| Evita margens muito grandes (espaço vazio); | |
| Evita a necessidade de recorrer a *scrollbars* horizontais. | |

## *fixed layout*

- Largura fixa em pixéis (px)
  - Layout tradicional uma vez que a esmagadora maioria dos acessos era efetuada a partir de *desktops*
    - **Esta situação já não se verifica !!**
      - ex: 1024 x 768 (width: 960px)
  - Mesmo para uma dimensão fixa, inflexível e dependente do alinhamento relativamente à janela do browser:
    - à esquerda
      - margem **variável** do lado direito (resulta num layout muito desequilibrado);
    - centrado
      - garantido margens (esquerda/direita) de largura igual, o que melhora o enquadramento do conteúdo



# Layouts

Tipos

Abordagens

- CSS FlexBox
  - Baseia-se num conjunto de propriedades que permitem a disposição flexivel dos elementos ao longo de um eixo.
- CSS Grid Layout
  - Baseia-se na criação de um grid container para disposição dos elementos por linha/coluna
  - Suportado pelos browsers (sem vendor prefix) a partir de 2017
- Frameworks CSS
  - Forma rápida e poderosa de criação de um layout (exemplo:Bootstrap)
  - Aprendizagem de um novo framework
- *float /clear*
  - Implementação simples. Ligado ao fluxo do HTML o que reduz a sua flexibilidade



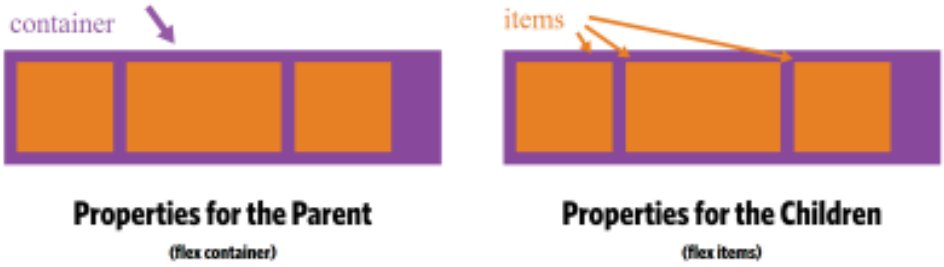
# CSS Flexbox (Layouts)

# CSS Flexbox

- *flexbox container*
  - permite agilizar  a definição de layouts

*According to the specification, the Flexbox model provides for an efficient way to **layout, align, and distribute space among elements within your document** — even when the viewport and the size of your elements is dynamic or unknown.*

https://medium.freecodecamp.org/understanding-flexbox-everything-you-need-to-know-b4013d4dc9af




- *{display:flex}*

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

  - cria  um *container* flexível com um conjunto de propriedades para o *container* e para os respetivos *items*



# CSS Flexbox *(layout)*

- Pode ser aplicado a diversos elementos
  - ul ->  flex container
  - li ->  flex item (filhos do flex container)

```
<ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
</ul>
```

```
ul{display:flex;}

li {    width: 100px;
       height: 100px;
       background-color:lightblue;
       margin:8px;

       list-style-type:none;
       text-align:center;
       color:white}
```

# CSS Flexbox (Layouts)

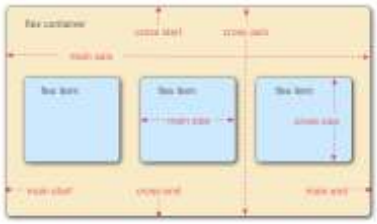
*flex container*

*flex item*



## *CSS Flexbox (layout)*

- Propriedades do ***flex container***:



  - *flex-direction*
    - Estabelece a direção do eixo principal ao longo do qual os elementos são dispostos
  - *flex-wrap*
    - Define se o *container* adapta a sua dimensão ou se cria múltiplas linhas para conter os *flex items*
    - ***flex-flow***
      - propriedade agregada de *flex-direction* e *flex-wrap* (ex: *flex-flow: row wrap*;)
  - align-items
    - define o alinhamento dos *items* ao longo do *cross axis*
  - align-content
    - define o alinhamento dos *items* ao longo do *cross axis* quando existe mais do que uma linha
  - justify-content
    - define o alinhamento dos *items* ao longo do *main axis*

# CSS Flexbox *(layout)*

- *Propriedades dos **flex container***
  - *flex-direction*: row |row-reverse | column | column-reverse


  - *flex-wrap*: nowrap|wrap|wrap-reverse
    - O comportamento por defeito é nowrap, o flex container adapta-se aumentando de tamanho para conter todos os flex items
    - O valor wrap cria múltiplas linhas

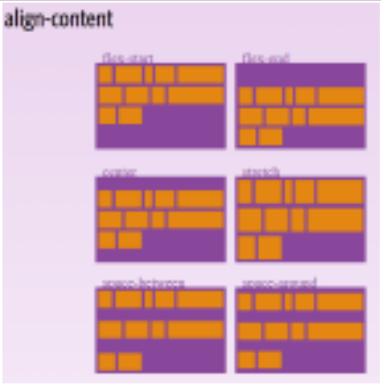

  - *align-items*: flex-start | flex-end | center | space-between | space-around | stretch;


https://css-tricks.com/snippets/css/a-guide-to-flexbox/



# CSS Flexbox *(layout)*

  - align-content: flex-start | flex-end | center | space-between |space-around | stretch;
    - **Não se aplica** quando existe apenas uma linha de *items*




  - justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;

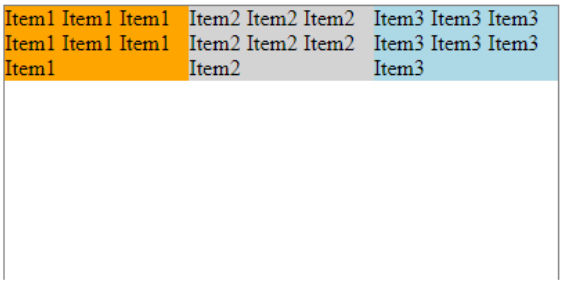

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

# CSS Flexbox *(layout)*

*Exemplo:*

```
<div id="container">
    <div id="d1">Item1 Item1 Item1 Item1 Item1 Item1 Item1 </div>
    <div id="d2">Item2 Item2 Item2 Item2 Item2 Item2 Item2 </div>
    <div id="d3">Item3 Item3 Item3 Item3 Item3 Item3 Item3 </div>
</div>
```

```
#container{
      height:400px;
      width:400px;
      display:flex;
      flex-direction: row;
      align-items:flex-start;}
```




- Na definição do *container* flexível ***(display:flex):***
  - Conteúdos apresentados por linha ***(flex-direction:row)***
  - Alinhamento dos *items* seria feito a partir do topo ***(align-items:flex-start)***

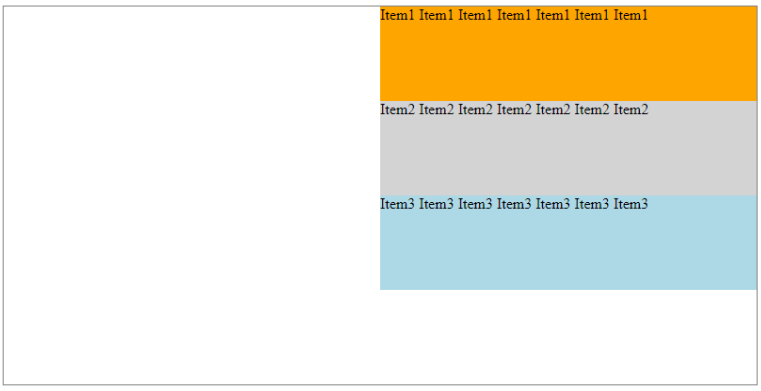
Os elementos aproveitam a largura disponível (400px), são dispostos com a mesma largura e pela ordem que surgem no HTML



# CSS Flexbox *(layout)*

```
<div id="container">
    <div id="d1">Item1 Item1 Item1 Item1 Item1 Item1 Item1 </div>
    <div id="d2">Item2 Item2 Item2 Item2 Item2 Item2 Item2 </div>
    <div id="d3">Item3 Item3 Item3 Item3 Item3 Item3 Item3 </div>
</div>
```

```
#d1{background-color: orange;}
#d2{background-color: lightgray;}
#d3{background-color: lightblue;}

#container{
    height:400px;
    width:800px;
    border: 1px gray solid;
    display:flex;
    flex-direction: column;
    align-items:flex-end;
}

#container div{
    height:100px;
    margin:0px;
    padding:0px;
    width: 400px;}
```




- Mesmos elementos (estrutura) mas:
  - Conteúdos apresentados por coluna ***(flex-direction:column)***
  - Alinhamento dos *items* a partir do final ***(align-items:flex-end)***

- O flex container permite que sejam aplicadas propriedades aos flex items:
  - Exemplo: neste caso podem ser definidos o *padding, a margin e  a height* de cada flex item

```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
}
```



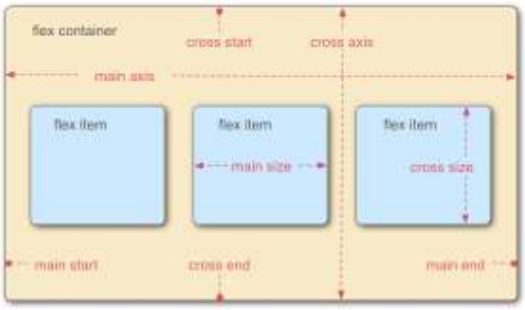

# CSS Flexbox (Layouts)

*flex container*

*flex item*

# CSS Flexbox *(layout)*

- *Propriedades dos **flex items***




  - *order*
    - permite reordenar os *flex items* num *container*
  - *flex-grow*
    - estabelece quanto é que um item pode crescer se existir espaço livre no *flex container*
  - *flex-shrink*
    - estabelece quanto é que um item pode reduzir caso não exista espaço no *flex container*
  - *flex-basis*
    - define a dimensão inicial de um *flex item*
  - *flex*
    - propriedade agregada (*flex: flex-grow flex-shrink flex-basis*)



# CSS Flexbox *(layout)*

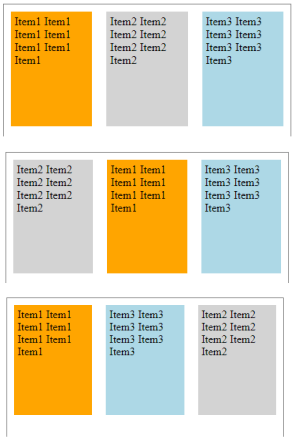
- Propriedades para os ***flex items***:
  - {*order: <integer>;*}
    - Por defeito a ordem dos *items* segue a ordem do HTML, sendo que todos os *flex items* possuem o valor 0 (*order; default value*).
      - É possível controlar a posição de um item controlando o valor desta propriedade
        - um valor negativo coloca o *item* em primeiro lugar
        - um valor positivo posiciona o item depois de todos aqueles com um valor de *order* inferior.

```
<div id="container">
    <div id="d1">Item1...</div>
    <div id="d2">Item2...</div>
    <div id="d3">Item3...</div>
</div>
```

```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
}
```

```
#d2{order:-1;}
```

```
#d2{order:1;}
```



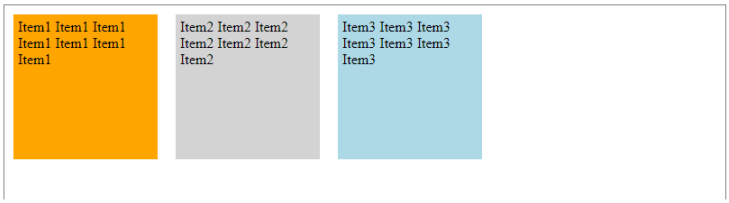
# CSS Flexbox *(layout)*

- *{flex-grow: <integer>;}*
  - Estabelece a capacidade de um item para crescer
  - Aceita um valor que define a quantidade de **espaço disponível** que cada um dos *items* vai ocupar.
    - Se todos os *items* possuírem o valor 1  o espaço disponível é dividido de igual forma pelos diversos items
    - Caso contrário o item com o maior valor vai ocupar mais espaço do que os outros *items*.

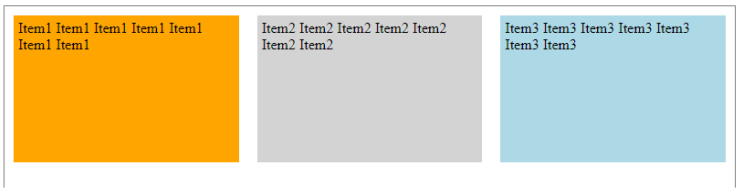
```
#container{
    height:400px;
    width:800px;
    border: 1px gray solid;
    display:flex;
    flex-direction: row;
    align-items:flex-start;}
```

```
#container div{ width:150px;
                height:150px;
                margin:10px;
                padding:5px;}
```

```
#container div{flex-grow: 1;}
```




Se largura/altura inicial dos diversos items é diferente, então as dimensões finais também serão diferentes apesar do espaço disponível ser distribuído de igual forma






# CSS Flexbox *(layout)*

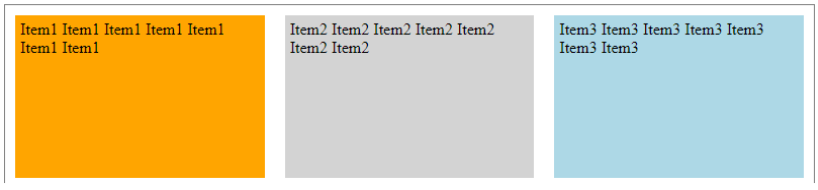
- {flex-shrink: <integer>;}
  - Capacidade de um item para reduzir a sua dimensão, o valor inteiro define o grau de redução atribuído a esse elemento.
  - Esta propriedade só se aplica em situação de **overflow** (largura items > largura do container)

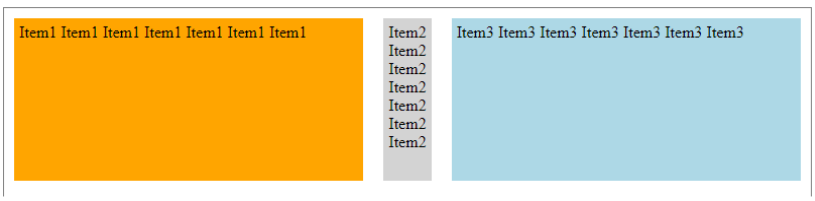
```
#container{
    height:400px;
    width:800px;
    border: 1px gray solid;
    display:flex;
    flex-direction: row;
    align-items:flex-start;}
```

```
#container div{
        height:150px;
        margin:10px;
        padding:5px;
        flex-grow: 1;
        flex-shrink: 1;
        width: 400px;}
```

```
#container #d2{flex-shrink:6;}
```

Como a largura está definida e existe uma situação de *overflow*, a dimensão dos *flex-items* irá ajustar-se à largura do *flex container*

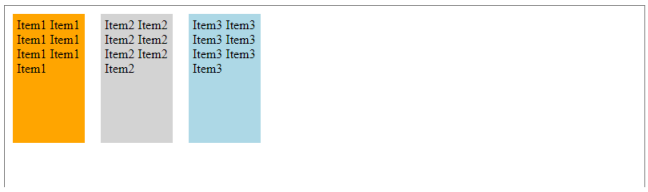


# CSS Flexbox *(layout)*

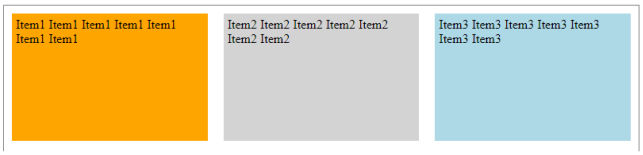
- {flex-basis: <length> | auto;}
  - Define a dimensão (%, em, …) inicial de um flex item.

```
#container div{
          height:150px;
          margin:10px;
          padding:5px;
         /*flex-grow: 1;*/
         }

#container div{flex-basis:10%;}
```




  - Se *flex-grow* definido o espaço livre é repartido pelos *flex-items*

```
#container div{
          height:150px;
          margin:10px;
          padding:5px;
          flex-grow: 1;
         }

#container div{flex-basis:10%;}
```

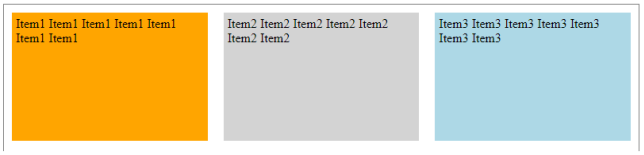




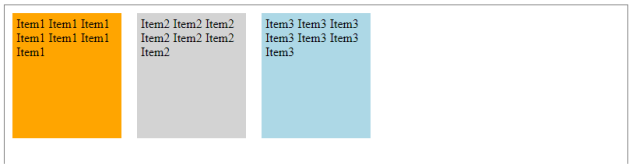

# CSS Flexbox *(layout)*

- {flex-basis: <length> | auto;}
  - Se a largura do elemento não se encontrar definida, o valor **auto** ajusta ao conteúdo de cada um dos flex items
    - Caso a largura do *flex-item* tenha sido definida (*width*), *{flex-basis:auto}* não altera essa definição.

```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
 /*  flex-grow: 1; */
 /*   width: 130px;*/}

#container div{flex-basis:auto;}
```



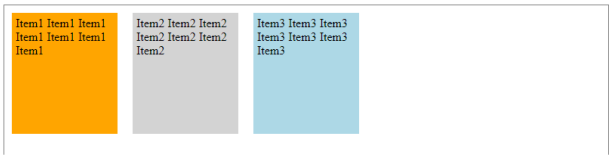

```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
 /* flex-grow: 1; */
    width: 130px;}

#container div{flex-basis:auto;}
```



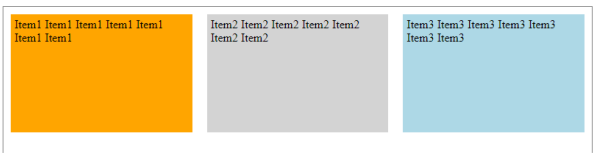
# CSS Flexbox *(layout)*

- {flex: 0 1 auto}
  - propriedade agregada para os valores *flex-grow, flex-shrink* (opcional) e *flex-basis* (opcional) .
  - Os valores por defeito são (0 1 auto).

```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
    width: 130px;}

#container div{flex:0 1 auto;}
```




```
#container div{
    height:150px;
    margin:10px;
    padding:5px;
    width: 130px;}

#container div{flex:1 1 auto;}
```






# CSS Flexbox *(layout)*

- ***align-self***: auto || flex-start || flex-end || center || baseline || stretch
  - Permite o alinhamento de um *flex item*, mantendo o alinhamento previamente definido para os restantes *flex items*

```
#container div{
        height:150px;
        margin:10px;
        padding:5px;
        width: 130px;}

#container div{flex:1 1 auto;}

#d2{align-self:flex-end}
```

# Exercício Seletores