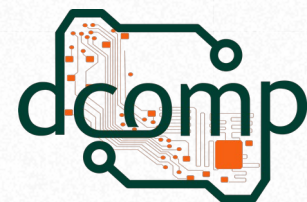




Universidade Federal do Espírito Santo
Centro de Ciências Exatas, Naturais e da Saúde
Departamento de Computação



Fundamentos de Programação Web

Grid Layout

Fundamentos de Programação WEB

Site: <http://www.jeiks.net/fundpweb>

E-mail: jacson.silva@ufes.br

Modo de Layout com Grid

- Além do Flexbox, outro *layout* que pode ser utilizado é a Grid:

display: grid; (*fará seu container ser block*)

display: inline-grid (*fará seu container ser inline*)

grid:

1	2	3
4	5	6
7	8	9

inline-grid:

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

Itens da Grid

- Para começar a usar a *grid*, deve-se definir um *container*:

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>

  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>

  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
</div>
```

```
<style>
.container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.item {
  background-color: rgba(255,255,255,0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```


Suas colunas

- Trabalharemos com as linhas verticais da grid:
colunas (*columns*):
grid-columns

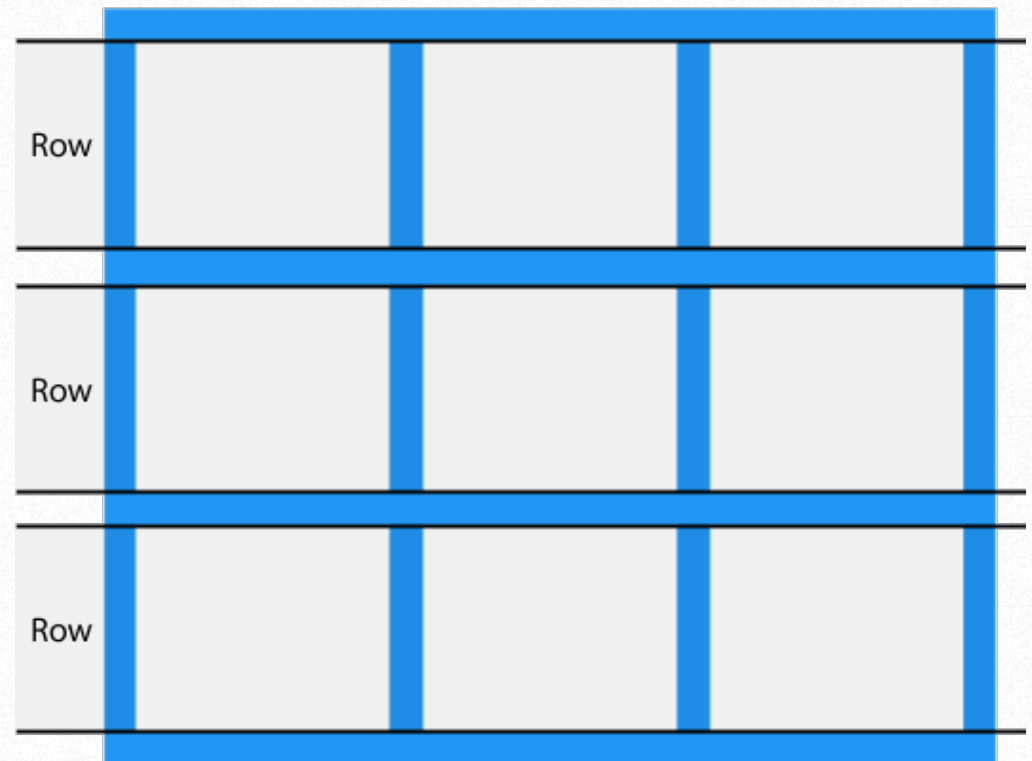
Column	Column	Column

Suas linhas

- Trabalharemos com linhas horizontais da grid:

linhas (*rows*):

grid-rows



E seus espaçamentos

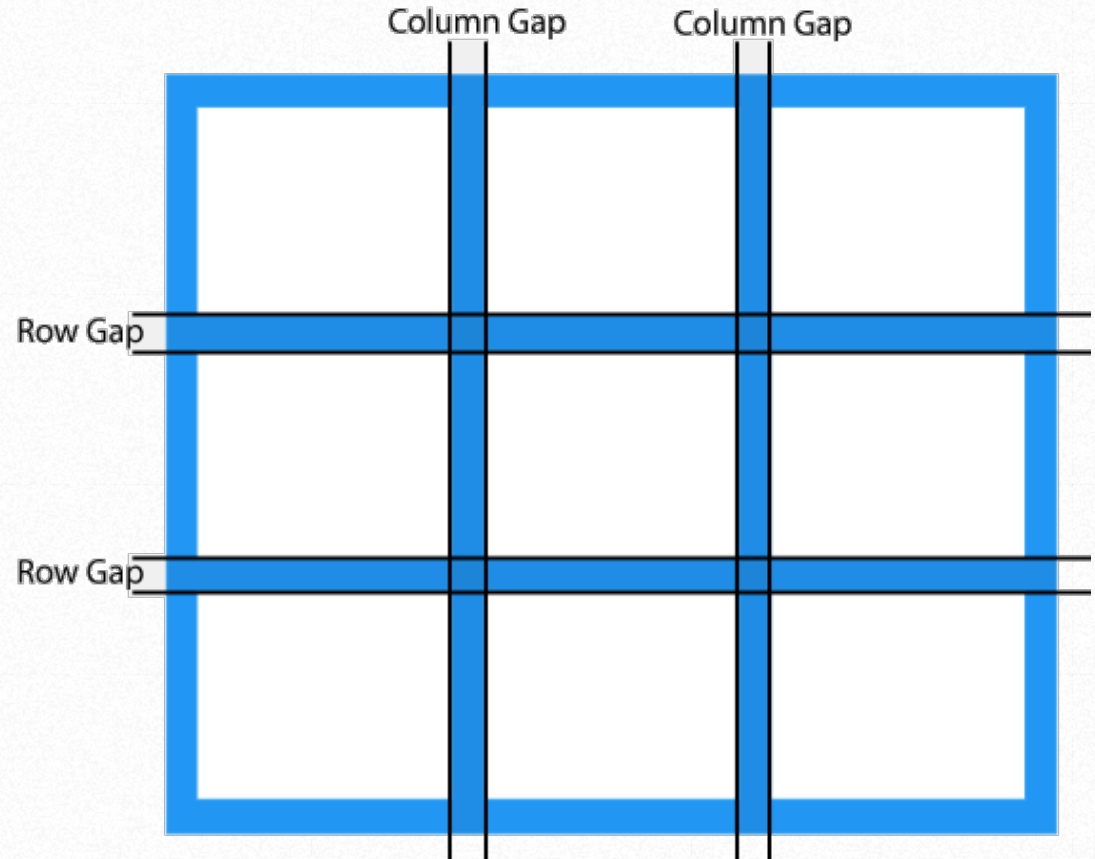
- O espaço entre cada linha (*row*) e cada coluna (*column*) é chamado de lacuna (*gap*):

- Lacuna da coluna:

Column gap

- Lacuna da linha:

Row gap



Propriedades do *container* Grid

- Após definir o *display* como *grid*, diversas propriedades podem ser utilizadas com:
 - as colunas;
 - as linhas;
 - as lacunas; e
 - o posicionamento.

Trabalhando com colunas

Grid-template-columns:

- Quantidade de colunas e sua disposição. Exemplos:

grid-template-columns: auto auto auto;

0	1	2
3	4	5
6	7	

grid-template-columns: auto auto auto auto;

0	1	2	3
4	5	6	7

grid-template-columns: 80px 200px min-content max-content;

00000	1	2	3
4	5	6	77777

grid-template-columns: repeat(7, auto)

0	1	2	3	4	5	6
7						

Trabalhando com colunas

- Opções de *grid-template-columns*:
 - *tamanho*: tamanho não negativo.
 - *porcentagem*: porcentagem de seu container.
 - *flex (1fr)*: dimensão com a unidade *fr* especificando o fator de flexibilidade.
 $1fr = (\text{largura_do_container} - \text{largura_fixa_da_coluna}) / [\text{soma dos flex}]$
 - *max-content*: palavra-chave, menor tamanho necessário para que o conteúdo caiba no elemento sem ser encapsulado, estourado, modificado.
 - *min-content*: palavra-chave, menor tamanho possível para que o conteúdo caiba no elemento sem que sobrescreva seu conteúdo. Mas ela pode, por exemplo, dividir um parágrafo em linhas.
 - *minmax(min, max)*: o valor do elemento deve ser entre min e max.
 - *auto*: automático, mas faz esticar os elementos para ocuparem todo seu container.
 - *repeat([inteiro_positivo | auto-fill | auto-fit] , <lista de opções>)*

Boa explicação em:

<https://css-tricks.com/almanac/properties/g/grid-template-columns/>

Expansão de colunas e linhas

- Colunas:

`grid-column-start: Coluna_Inicial;`

`grid-column-end: Coluna_Final+1;`

ou: `grid-column: Coluna_Inicial / Coluna_Final+1;`

ou: `grid-column: Coluna_Inicial / span Colunas_a_abranger;`

- Linhas:

`grid-row-start: Linha_Inicial;`

`grid-row-end: Linha_Final+1;`

ou: `grid-row: Linha_Inicial / Linha_Final+1;`

ou: `grid-row: Linha_Inicial / span Linhas_a_abranger;`

1 (col: 1-3)			2
3 (col: 1-4)			
4 (row: 3-5)	5	6	
	7	8	

Expansão de colunas e linhas

- Desenhando a área:

`grid-template-area:`

`ident`: identificador do item a expandir.

`ponto (.)`: posição a manter.

`aspas simples`: separam as linhas.

- Exemplo:

no item1:

`grid-area: nome;`

no container:

`grid-template-areas: 'nome nome . . ' 'nome nome . . ';`

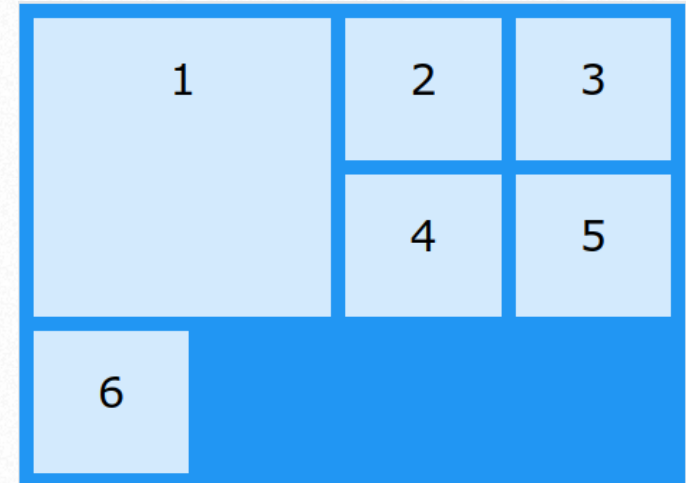
OU:

no container:

`grid-template-columns: repeat(4, auto);`

no item1:

`grid-area: 1 / 1 / span 2 / span 2;`



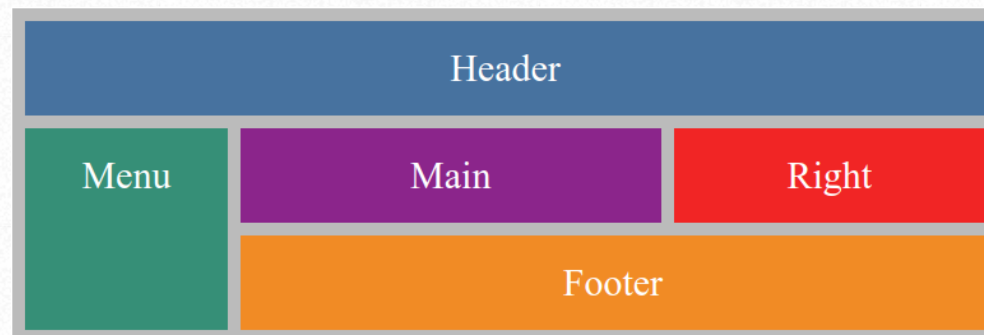
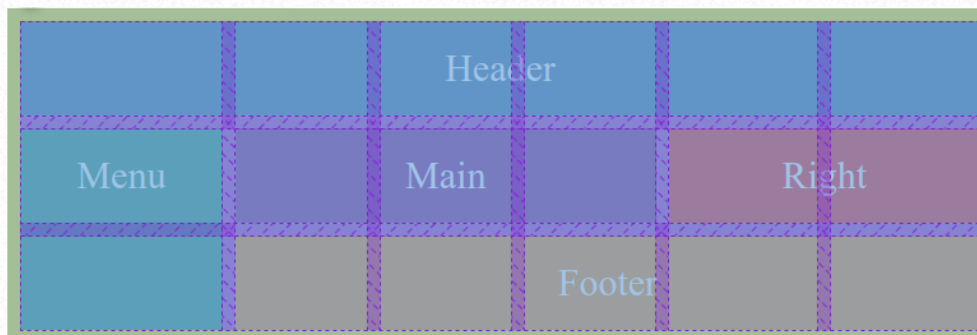
Expansão de colunas e linhas

- Exemplo da utilização de *grid-area*:

```
<div class="container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```



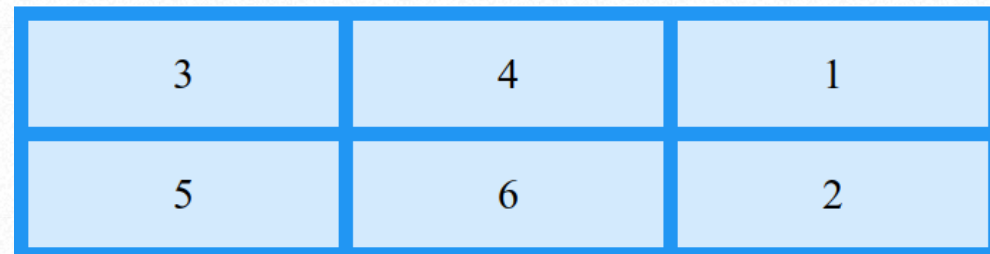
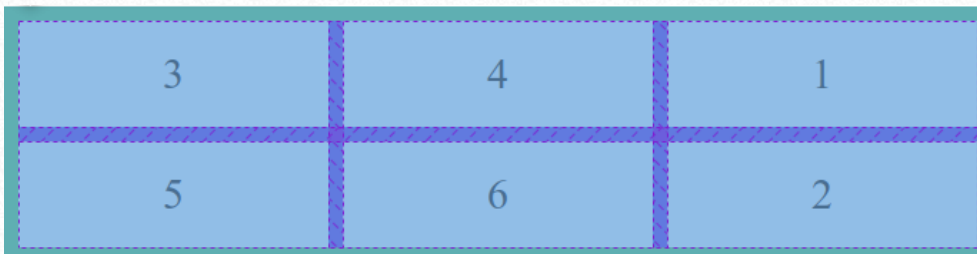
Expansão de colunas e linhas

- E a grade não precisa seguir a ordem do HTML:

```
<div class="container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
</div>
```

```
.container {
  display: grid;
  grid-template-columns: auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```

```
.item1 { grid-area: 1 / 3 / 2 / 4; }
.item2 { grid-area: 2 / 3 / 3 / 4; }
.item3 { grid-area: 1 / 1 / 2 / 2; }
.item4 { grid-area: 1 / 2 / 2 / 3; }
.item5 { grid-area: 2 / 1 / 3 / 2; }
.item6 { grid-area: 2 / 2 / 3 / 3; }
```



Definição de lacunas

- Lacuna entre linhas:

```
row-gap: tamanho;
```

- Lacunas entre colunas:

```
col-gap: tamanho;
```

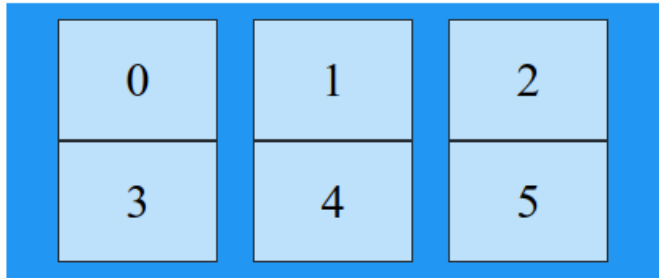
- Atalho para definir lacunas entre linhas e colunas:

```
gap: tamanho_linha tamanho_coluna;
```

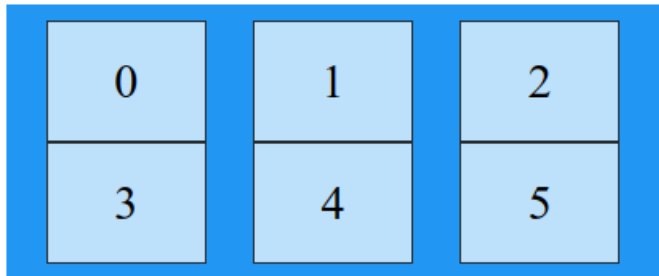
row-gap: 20px	column-gap: 20px	gap: 20px 20px																											
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8
0	1	2																											
3	4	5																											
6	7	8																											
0	1	2																											
3	4	5																											
6	7	8																											
0	1	2																											
3	4	5																											
6	7	8																											

Propriedade *justify-content*

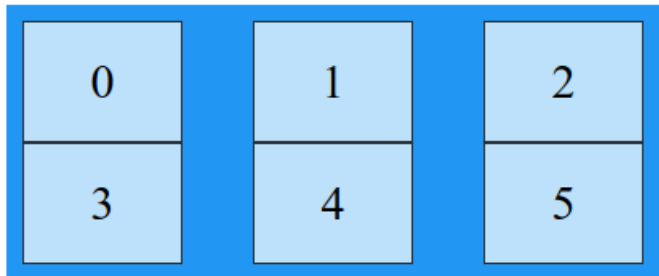
justify-content: space-evenly
(uniforme)



justify-content: space-around



justify-content: space-between



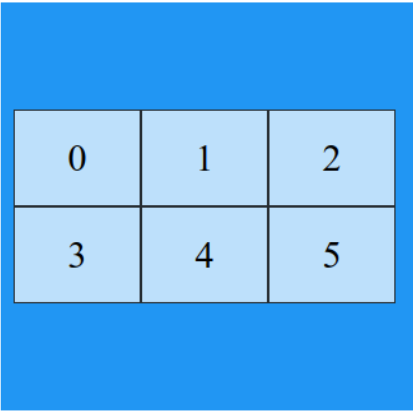
Espaçamento horizontal dos elementos:

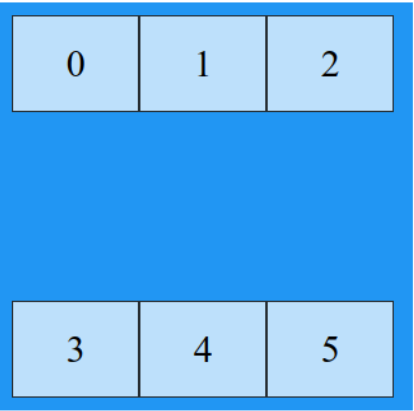
- *space-evenly*: espaçar uniformemente;
- *space-around*: espaçar ao redor;
- *space-between*: espaçar entre eles.

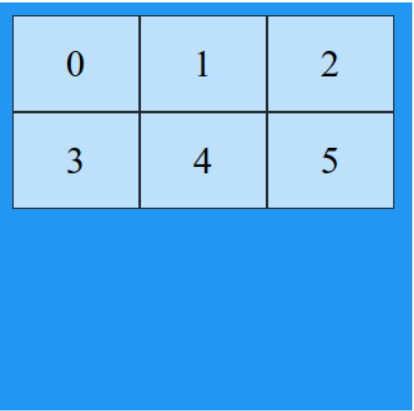
Propriedade *align-content*

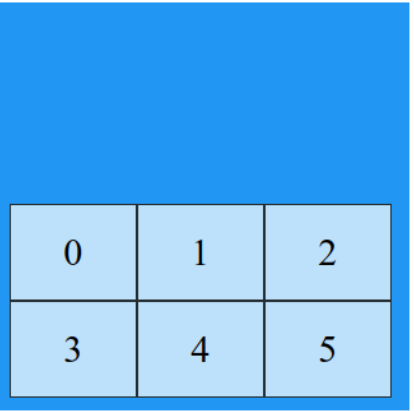
Alinhamento vertical dos elementos:

- *center*: alinhar ao centro;
- *space-evenly*: espaçar uniformemente;
- *space-around*: espaçar ao redor;
- *space-between*: espaçar entre eles;
- *start*: alinhar no início;
- *end*: alinhar no fim.

align-content: center


align-content: space-between


align-content: start


align-content: end


Comentários adicionais: box-sizing

- Propriedade: box-sizing:
 - permite incluir o preenchimento e a borda na largura e altura total de um elemento,
isto é:
 - permite incluir o *padding* e a *border* no valor *width* e *height* definido para o elemento.
- Por padrão, a largura e a altura de um elemento são calculadas assim:
 $\text{largura} + \text{preenchimento} + \text{borda} = \text{largura real de um elemento}$
 $\text{width} + \text{padding} + \text{border} = \text{largura real de um elemento}$
 $\text{altura} + \text{preenchimento} + \text{borda} = \text{altura real de um elemento}$
 $\text{height} + \text{padding} + \text{border} = \text{altura real de um elemento}$
- Isso significa:
 - Quando você define a largura/altura de um elemento, o elemento geralmente aparece maior do que você definiu.
→ a *border* e o *padding* são adicionados às *width/height* definida do elemento.

Comentários adicionais: box-sizing

- Exemplo de dois *divs* com as mesmas width e height:

Este div é menor (largura é 300px e altura é 100px).

Este div é maior (a largura também é 300px e a altura é 100px).

```
.div1 {  
  width: 300px;  
  height: 100px;  
  border: 1px solid blue;  
}  
  
.div2 {  
  width: 300px;  
  height: 100px;  
  padding: 50px;  
  border: 1px solid red;  
}
```

- A propriedade `box-sizing` resolve este problema.

Comentários adicionais: box-sizing

- Ao definir em um elemento:
box-sizing: border-box;
 - o preenchimento e a borda serão incluídos na largura e na altura.
 - Exemplo:

Ambos os divs são do mesmo tamanho agora!

Viva!

```
.div1 {  
  width: 300px;  
  height: 100px;  
  border: 1px solid blue;  
  box-sizing: border-box;  
}  
  
.div2 {  
  width: 300px;  
  height: 100px;  
  padding: 50px;  
  border: 1px solid red;  
  box-sizing: border-box;  
}
```

Comentários adicionais: box-sizing

- Por ter um resultado muito bom,
 - o “the box-sizing: border-box” costuma ser utilizado por muitos desenvolvedores.
 - Assim, todos os elementos da páginas funcionam dessa maneira.
- O código abaixo garante que todos os elementos sejam dimensionados dessa maneira mais intuitiva:

```
* {  
    box-sizing: border-box;  
}
```

Muitos navegadores já usam essa propriedade em muitos elementos, mas não todos. Por isso, aplicar essa propriedade a todos os elementos é uma alternativa segura e eficiente.

Comentários adicionais: box-sizing

- Valores da propriedade **box-sizing**:

content-box (padrão): o tamanho do objeto é o tamanho de sua largura (width), altura (height), borda (border) e espaçamento (padding).

border-box: o tamanho do objeto é sua largura (width) e altura (height). A borda (border) e o espaçamento (padding) são ajustados dentro desses valores.

initial: o valor é igual ao valor inicial do elemento, mesmo que seu pai receba outro estilo.

inherit: herda a propriedade de seu elemento pai.