

CSS

Cascading Style Sheets

CSS

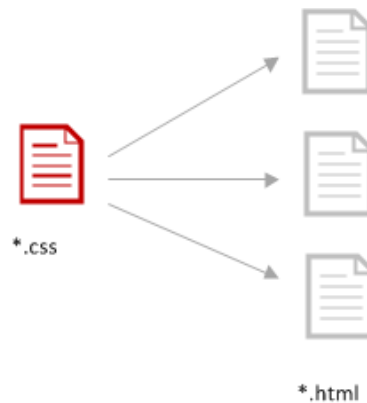
- Incremento da qualidade e consistência gráfica das aplicações
- Separação do **design e do conteúdo**:
 - **HTML** define a estrutura (conteúdo)
 - **CSS** define a formatação/design
 - Layouts
 - Posicionamentos
 - Cores
 - Texto
 - Formulários
 - Tabelas
 - Listas
 - ...

*



■ Vantagens

- Possibilidade de aplicação imediata a vários documentos HTML
 - Garante que diversos documentos HTML são submetidos às mesmas regras de formatação.
- Muito maior flexibilidade
 - Alterações centralizadas
 - Evita a repetição de formatações
- Maior correção
 - Melhora a portabilidade
 - Reduz a possibilidade de erros
 - Maior uniformidade
 - Melhora a consistência gráfica



Sintaxe CSS

- Composta por:
 - Seletor
 - Declaração /conjunto de declarações
 - Propriedade
 - Valor

seletor { propriedade:valor;

- Cada regra seleciona um elemento/conjunto de elementos e declara toda a formatação associada
- A sintaxe das CSS é **case sensitive**

■ Seletor

- Elemento crucial na definição de uma regra CSS
 - A compreensão do funcionamento dos seletores é absolutamente essencial para permitir uma correta implementação de CSS
- Permite identificar os elementos HTML a que a regra CSS se aplica



```
body{ background-color:#FFF;}
```

■ Declaração

- Uma atribuição de um valor a uma propriedade. Uma regra pode conter várias declarações (*declaration block*).
 - Propriedade
 - Atributo da folha de estilo ao qual deve ser atribuído um valor (ex: *color*, ...)

```
body{ background-color:#FFF;}
```

- Valor

- Especificação concreta da variável (ex: *arial*, *#0000FF*)

```
body{ background-color:#FFF;}
```

Sintaxe	Exemplo
selector { propriedade: valor}	body {background-color: #FFFFFF}
<i>bloco de declarações ;</i>	
selector { propriedade: valor; propriedade: valor; propriedade: valor; }	p { text-align:center; color:red; font-family: arial;} }
<i>seleção múltipla ,</i>	
selector1, selector2, selector3 { propriedade: valor;}	h1, h2, h3{ color:red;}

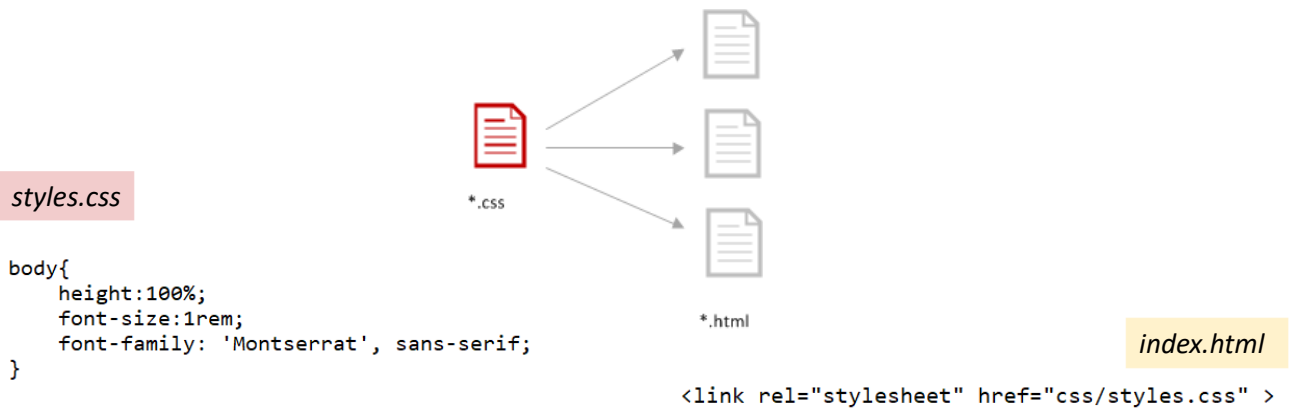
Ligação CSS / HTML

Ligação CSS / HTML

- Como aplicar estilos (regras CSS) aos documentos HTML?

- **External Style Sheet**

- Ficheiro externo de extensão *.css que contém as regras CSS
 - Pode ser aplicado a diversos documentos HTML
 - Forma mais poderosa e flexível de incorporar CSS



Ligação CSS / HTML

- **External Style Sheet (*.css)**

- Um ficheiro CSS (*.css) é exclusivamente constituído por regras CSS
 - O código CSS não contém tags

seletor{propriedade:valor;}

```
/* style1.css */  
  
p { font-weight:bold;  
    color:#F00;}
```

■ `<link ... />`

- Atributo **href** indica a localização do ficheiro *.css . Atributo obrigatório!
- Atributo **rel** define a relação do documento HTML com o ficheiro externo, neste caso é *stylesheet*. Atributo obrigatório!
- Atributo **type** é opcional na última versão do HTML5 (legado de versões anteriores)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8" />
<title> External Style Sheet</title>

<link href="style1.css" rel="stylesheet" type="text/css"/>

</head>

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

```
/* style1.css */
```

```
p { font-weight:bold;
    color:#F00;}
```

■ **External Style Sheet**

- através de **@import** e da tag **<style>**

```
<!DOCTYPE HTML>
<html>
<head>
  <style>
    @import url("style1.css");
    ...
  </style>
  <meta charset="utf-8" />
  <title> External Style Sheet</title>
</head>

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

```
/* style1.css */
```

```
p { font-weight:bold;
    color:#F00;}
```

- as duas formas de estabelecer a ligação a um ficheiro externo são suportadas pela maioria dos browsers originando resultados semelhantes, no entanto:
 - A tag **<link>** é a forma mais eficaz, aquela que assegura melhor performance, para efetuar a ligação de uma HTML a um CSS externo

■ *Embedded Style Sheet*

- Necesita de ser definido em todos os documentos *.html
 - Menos flexível e eficaz do que a ligação a um ficheiro externo
 - Aumenta o peso dos ficheiros
 - Uma alteração na formatação implica uma alteração em cada um dos ficheiros
 - em MPA pode originar inconsistência na formatação entre diferentes *.html



■ `<style> ... </style>`

- definido do `<head>` do *.html
 - Atributo `type="text/css"` é opcional (obrigatório nas versões anteriores ao HTML5).

Como é óbvio **não podem** ser definidas tags HTML nesta janela, trata-se de código CSS

```
<!DOCTYPE HTML>
<html>
<head>
  <style>
    p { font-weight:bold;
        color:#F00;}
  </style>
  <meta charset="utf-8" />
  <title> Embedded Style Sheet</title>
</head>

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

Janela de
código CSS

■ *Inline Style*

- aplicar o estilo localmente através do atributo **style**
 - É a forma **menos flexível** e **mais propensa** a erros para formatar um elemento HTML
 - Se efetuado de forma estática, favorece a inconsistência na formatação dos conteúdos
 - Implica tantas alterações quantas as definições efetuadas

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
</head>
<body>
  <h1 style="color:orange">Inline Style: A evitar!</h1>
</body>
</html>
```

Inline Style: A evitar!

Seletores CSS

Seletor CSS

- Permite identificar os elementos HTML aos quais a regra CSS se aplica

seletor { propriedade:valor;}

- **Diferentes tipos de seletores** de forma a permitir a flexibilidade necessária para selecionar qualquer dos elementos/conjunto de elementos definidos no HTML
- Os seletores CSS são cruciais para a aplicação da tecnologia CSS, mas também são usados pelo JavaScript / Frameworks JS (ex: React) para aceder a determinado elemento HTML.

Seletores CSS **não se esgotam** na tecnologia CSS!

- Devem ser definidos tendo sempre em consideração os conflitos de formatação:
 - Muito frequentes sempre que o código CSS ganha alguma escala.

Seletor CSS

- Principais tipos de seletores:

Elemento

Contexto

id

class

Atributo

- Principais tipos de seletores:

Elemento

Seletor CSS

- Seletor de **Elemento**

Elemento

- Referência direta ao elemento HTML que se pretende formatar
 - Vantagem: simplicidade
 - Desvantagem: pouco específicos

```
<style>
  p{border:1px solid blue;}
  h2{color:blue}
</style>
```

- Elementos Agrupados

- Referência direta aos elementos HTML que se pretendem formatar, os quais devem ser separados por **vírgula (,)**

```
<style>
  h1,h2{ color: #900;}
</style>
```

Seletor CSS

- Seletor Universal (*)
 - Permite selecionar todos os elementos
 - A utilização deste seletor deve ser cuidadosamente ponderada é aplicado a **todos os elementos**
 - podem ser originadas algumas formatações/efeitos não previstos

```
<style>  
  * {color:orange}  
</style>  
</head>  
  
<body>  
  <p>p first-child</p>  
  <ul>  
    <li>first option</li>  
    <li>second option</li>  
  </ul>  
  <p>p last-child</p>  
</body>
```

p first-child

- first option
- second option

p last-child

Seletor CSS

- Principais tipos de seletores:

Contexto

Seletor CSS

■ Seletor de **Contexto**

■ Descendentes

- A formatação dos elementos depende do contexto
- Elementos separados por **espaço em branco**

Contexto

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    li em {color:orange;}
  </style>
</head>
<body>
  <ul>
    <li><em>1ª opção</em></li>
    <li><em>2ª opção</em></li>
    <li><em>3ª opção</em></li>
  </ul>
  <p>1º <em>parágrafo</em></p>
  <p>2º <em>parágrafo</em></p>
</body>
</html>
```

- 1ª opção
- 2ª opção
- 3ª opção

1º parágrafo

2º parágrafo

Podem ser criados vários
níveis de dependências

```
<style>
  ul li em {color:orange;}
</style>
```

Seletor CSS

■ Seletores de **Contexto**

■ Casos particulares de seletores de contexto

- Descendentes diretos (**filhos**) - *child selector* (>)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    span {color:gray;}
    p > span {color:orange;font-size:1.2em}
  </style>
</head>
<body>
  <p><span>1º <span>parágrafo</span></span></p>
  <p><span>2º <span>parágrafo</span></span></p>
</body>
</html>
```

1º parágrafo

2º parágrafo

Seletor CSS

■ Seletores de **Contexto**

■ Elementos Adjacentes (**Adjacent Sibling Selector**) (+)

- Adjacente: elemento imediatamente seguinte ao elemento que define o contexto

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    → h1 + p {color:orange;font-size:1.2em}
  </style>
</head>
<body>
  <h1 id="p1"> Referência</h1>
  <p><span>1º </span>parágrafo</span></p>
  <p><span>2º </span>parágrafo</span></p>
</body>
</html>
```

Referência

1º parágrafo

2º parágrafo

Seletor CSS

■ Seletores de **Contexto**

■ *element1 ~ element2*

- seleciona todos os *element2* que são precedidos por um *element1*
 - Os elementos devem ter o mesmo pai
 - O *element2* não tem de ser imediatamente precedido pelo *element1*

```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <title></title>

  <style>
    → h1 ~ p {color:orange}
  </style>
</head>
<body>
  <p> 1º parágrafo </p>
  <h1 id="heading1"> Referência </h1>
  <p> 2º parágrafo </p>
  <p> 3º parágrafo </p>
</body>
</html>
```

1º parágrafo

Referência

2º parágrafo

3º parágrafo

- Principais tipos de seletores:

id

Seletor CSS

- Selectores de ID (#)

- Atribuir um estilo a uma única ocorrência de um elemento HTML

id

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    p {color:gray;}
    #p1 {color:orange;font-size:1.2em}
  </style>
</head>
<body>
  <p id="p1"> Parágrafo com id=p1</p>
  <p><span>1º</span> <span>parágrafo</span></span></p>
  <p><span>2º</span> <span>parágrafo</span></span></p>
</body>
</html>
```

Parágrafo com id=p1

1º parágrafo

2º parágrafo

- Os id's são únicos e como tal dispensam a especificação do elemento
 - seletores equivalentes
 - Especificidade diferente

#p1 p#p1

Seletor CSS

- Seletor de ID (#)
 - Podem ser utilizados como parte de um seletor de contexto

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li {color:gray;}
    #lista1 li {color:orange;}
  </style>
</head>
<body>
  <ul id="lista1">
    <li>1ª opção / lista id="lista1" </li>
    <li>2ª opção / lista id="lista1" </li>
  </ul>
  <ul>
    <li>1ª opção / lista sem id </li>
    <li>2ª opção / lista sem id </li>
  </ul>
</body>
</html>
```

- 1ª opção / lista id="lista1"
- 2ª opção / lista id="lista1"
- 1ª opção / lista sem id
- 2ª opção / lista sem id

Seletor CSS

- Principais tipos de seletores:

class

Seletor CSS

■ Seletor de *class* .

- Ao contrário do atributo ID:
 - o atributo *class* pode ser partilhada por múltiplos elementos
- Um elemento pode ter definida mais de uma *class*
 - Sem limite, separadas por espaço, sendo indiferente a ordem pela qual são definidas.

class

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li {color:gray;}
    .opc {color:orange;}
  </style>
</head>
<body>
  <ul id="lista1">
    <li class="opc">1ª opção / lista id="lista1" </li>
    <li class="opc">2ª opção / lista id="lista1" </li>
  </ul>
  <ul>
    <li class="opc">1ª opção / lista sem id </li>
    <li>2ª opção / lista sem id </li>
  </ul>
</body>
</html>
```

- 1ª opção / lista id="lista1"
- 2ª opção / lista id="lista1"
- 1ª opção / lista sem id
- 2ª opção / lista sem id

Seletor CSS

■ Seletor de *class* .

- *class* aplicada/definida com base num elemento específico

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li {color:gray;}
    p.opc {color:orange;}
  </style>
</head>
<body>
  <ul id="lista1">
    <li class="opc">1ª opção / lista id="lista1" </li>
    <li class="opc">2ª opção / lista id="lista1" </li>
  </ul>
  <ul>
    <li class="opc">1ª opção / lista sem id </li>
    <li>2ª opção / lista sem id </li>
  </ul>
  <p class="opc">1º Parágrafo</p>
</body>
</html>
```

- 1ª opção / lista id="lista1"
- 2ª opção / lista id="lista1"
- 1ª opção / lista sem id
- 2ª opção / lista sem id

1º Parágrafo

Seletor CSS

- Seletor de *class*.
 - Podem ser utilizados para criar um seletor contextual
 - São formatados os elementos integrados num outro elemento cujo atributo *class* foi definido com um valor específico.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li {color:gray;}
    .lst .opc {color:orange;}
  </style>
</head>
<body>
  <ul id="lista1">
    <li class="opc">1ª opção / lista id="lista1" </li>
    <li class="opc">2ª opção / lista id="lista1" </li>
  </ul>
  <ol class="lst">
    <li class="opc">1ª opção / lista sem id </li>
    <li>2ª opção / lista sem id </li>
  </ol>
  <p class="opc">1º Parágrafo</p>
</body>
</html>
```

- 1ª opção / lista id="lista1"
- 2ª opção / lista id="lista1"

1. 1ª opção / lista sem id
2. 2ª opção / lista sem id

1º Parágrafo

Seletor CSS

- **Importante!**
 - Especificação do elemento e seletor **id**

h1#d1{color:gray}

Formatação aplicada ao elemento **h1** cujo **id** é **d1**

≠

h1 #d1

- Especificação do elemento e seletor **class**

h2.c1{color:green}

Formatação aplicada ao elemento **h2** cuja **class** é **c1**

≠

h2 .c1

O espaço em branco permite implementar seletores totalmente diferentes

Seletor CSS

```
...  
<style>  
  h2.c1{color:orange}  
</style>  
  
</head>  
  
<body>  
  <div id="div1" class="c1">  
    <h1 id="d1">heading 1</h1>  
    <h2 class="c1">heading 2</h2>  
    <p class="c1">paragraph1</p>  
  </div>  
  <h2 class="c1">Other heading 2</h2>  
...
```

```
...  
<style>  
  .c1 h2{color:orange}  
</style>  
  
</head>  
  
<body>  
  <div id="div1" class="c1">  
    <h1 id="d1">heading 1</h1>  
    <h2 class="c1">heading 2</h2>  
    <p class="c1">paragraph1</p>  
  </div>  
  <h2 class="c1">Other heading 2</h2>  
...
```

Formatação aplicada aos elementos h2
cuja **class** é **c1**

heading 1
heading 2
paragraph1
Other heading 2

≠

A ordem dos elementos / espaço
em branco é determinante para
uma correta implementação de um
selector.

Formatação aplicada
aos elementos h2
definidos **no**
contexto de um
outro elemento cuja
class é **c1**

heading 1
heading 2
paragraph1
Other heading 2

Seletor CSS

- Principais tipos de seletores:

Atributo

Seletor CSS

■ Seletor de Atributo

■ [attribute]

Atributo

- Caso o elemento não seja especificado são afetados todos os elementos que possuem o atributo definido

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    [id] {color:orange;}
  </style>
</head>
<body>
  <p id="primeiro">1º Parágrafo</p>
  <p> 2º Parágrafo</p>
  <p id="terceiro">3º Parágrafo</p>
  <p> 4º Parágrafo</p>

  <ul id="lista"><li>primeira opção</li></ul>
</body>
</html>
```

1º Parágrafo
2º Parágrafo
3º Parágrafo
4º Parágrafo
• primeira opção

Seletor CSS

■ Attribute Selectors

- Referenciam um elemento através dos seus atributos ou dos valores que assumem

■ *element [attribute]*

- referencia os elementos especificados que têm declarado um atributo específico.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    p[id] {color:orange;}
  </style>
</head>
<body>
  <p id="primeiro">1º Parágrafo</p>
  <p> 2º Parágrafo</p>
  <p id="terceiro">3º Parágrafo</p>
  <p> 4º Parágrafo</p>

  <ul id="lista"><li>primeira opção</li></ul>
</body>
</html>
```

1º Parágrafo
2º Parágrafo
3º Parágrafo
4º Parágrafo
• primeira opção

Seletor CSS

■ ***element* [attribute="value"] / [atribute="value"]**

- referencia os elementos *element* cujo atributo assume um valor *específico*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    [id="lista">{color:orange;}
  </style>
</head>
<body>
  <p id="primeiro">1º Parágrafo</p>
  <p> 2º Parágrafo</p>
  <p id="terceiro">3º Parágrafo</p>
  <p> 4º Parágrafo</p>

  <ul id="lista"><li>primeira opção</li></ul>
</body>
</html>
```

1º Parágrafo

2º Parágrafo

3º Parágrafo

4º Parágrafo

- primeira opção

- Existem algumas variantes de seletor de atributo (menos utilizados) que selecionam os elementos a formatar com base em correspondências parciais dos valores dos atributos.

Seletor CSS

■ ***Attribute Selectors***

■ ***element* [attribute *= " ..."] / [atribute *= "..."]**

- referencia os elementos cujos valores do atributo especificado contêm uma dada **expressão** (palavra isolada ou inserida em outra palavra)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    p[id*="paragrafo"] {color:orange;}
  </style>
</head>
<body>
  <p id="paragrafo-primeiro">1º Parágrafo</p>
  <p id="segundo paragrafo">2º Parágrafo</p>
  <p id="paragrafo terceiro">3º Parágrafo</p>
  <p id="quarto">4º Parágrafo</p>
</body>
</html>
```

1º Parágrafo

2º Parágrafo

3º Parágrafo

4º Parágrafo

Seletor CSS

■ Attribute Selectors

■ **element** [attribute ^= "..."] / [attribute ^= "..."]

- referencia os elementos cujo valor do atributo **se inicia** com uma dada **expressão** (palavra isolada ou inserida em outra palavra)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    → p[id^="paragrafo"] {color:orange;}
  </style>
</head>
<body>
  <p id="paragrafo-primeiro">1º Parágrafo</p>
  <p id="segundo paragrafo">2º Parágrafo</p>
  <p id="paragrafo terceiro">3º Parágrafo</p>
</body>
</html>
```

1º Parágrafo

2º Parágrafo

3º Parágrafo

Seletor CSS

■ Attribute Selectors

■ **element** [attribute \$= "..."] / [attribute \$= "..."]

- referencia os elementos cujo valor do atributo **termina com** uma dada **expressão** (palavra isolada ou inserida em outra palavra)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    → p[id$="paragrafo"] {color:orange;}
  </style>
</head>
<body>
  <p id="paragrafo-primeiro">1º Parágrafo</p>
  <p id="segundo paragrafo">2º Parágrafo</p>
  <p id="paragrafo terceiro">3º Parágrafo</p>
</body>
</html>
```

1º Parágrafo

2º Parágrafo

3º Parágrafo

- Lista completa: http://www.w3schools.com/cssref/css_selectors.asp

```
<body>  
  <p class="cl1"> Parágrafo Exemplo <span> Elemento Inline </span></p>  
</body>
```

<style>

p {color:red;}

Parágrafo Exemplo Elemento Inline

span {color:blue;}

Parágrafo Exemplo Elemento Inline

p span {color:green;}

Parágrafo Exemplo Elemento Inline

p span {color:green;}

Parágrafo Exemplo Elemento Inline

p .cl1 {color:orange;}

Parágrafo Exemplo Elemento Inline

p.cl1 {color:orange;}

Parágrafo Exemplo Elemento Inline

</style>

Pseudo-Class Selectors

Pseudo-Class Selector

- Baseados no **estado** do elemento
 - *:link*
 - link não visitado
 - *:visited*
 - link anteriormente visitado
 - *:hover*
 - o cursor do rato encontra-se sobre o elemento
 - *:active*
 - o elemento está a ser selecionado
 - *:focus*
 - elemento selecionado e pronto para receber valores
 - *element:first-child*
 - elemento que é o primeiro filho do seu pai (**exceção!**)

Pseudo-Class Selector

■ Pseudo-Class selector (exemplo)

■ *:hover*

- Sobreposição do cursor do rato

Pseudo-Class: estado link

Pseudo-Class: **estado link**

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    a:hover {color:white;background-color:#FF6600;font-size:20px;
  </style>
</head>
<body>
  <p>Pseudo-Class: <a href="http://www.isec.pt">estado link</a></p>
</body>
</html>
```

Pseudo-Class Selector

■ Pseudo-Class selector (exemplo)

■ *li:first-child*

- seleciona o elemento que é o primeiro filho do seu pai

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li:first-child {color:orange;font-size:20px}
  </style>
</head>
<body>
  <ul>
    <li>Primeiro filho de ul</li>
    <li>Segundo filho de ul</li>
    <li>Último filho de ul</li>
  </ul>
  <p> primeiro parágrafo </p>
</body>
</html>
```

- Primeiro filho de ul
- Segundo filho de ul
- Último filho de ul

primeiro parágrafo

Seletor *sempre aplicado* no elemento filho

Pseudo-Class Selector

■ *li:last-child*

- seleciona o elemento que é o último filho do seu pai

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li:last-child {color:orange;font-size:20px}
  </style>
</head>
<body>
  <ul>
    <li>Primeiro filho de ul</li>
    <li>Segundo filho de ul</li>
    <li>Último filho de ul</li>
  </ul>
  <p> primeiro parágrafo </p>
</body>
</html>
```

- Primeiro filho de ul
- Segundo filho de ul
- Último filho de ul

primeiro parágrafo

Pseudo-Class Selector

■ `:first-child` ; `:last-child`

```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <title></title>

  <style>
    div:first-child{color:orange}
    div:last-child{color:lightblue}
  </style>
</head>
<body>
  <div> Primeiro elemento div</div>
  <div> Ultimo elemento div</div>
</body>
</html>
```

Primeiro elemento div

Ultimo elemento div

Alteração da
estrutura
mantendo o
CSS

```
<body>
  <p>primeiro filho</p>
  <div> Primeiro elemento div</div>
  <div> Ultimo elemento div</div>
  <p>ultimo filho</p>
</body>
```

?

primeiro filho

Primeiro elemento div

Ultimo elemento div

ultimo filho

A formatação não é aplicada, uma vez que a alteração da estrutura originou **que os elementos `<div>` deixassem de ser o primeiro e o ultimo filho do seu pai** (neste caso o element `<body>`)

Seletor CSS

■ `li:nth-child(n)`

- seleciona o elemento que é o enésimo filho

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    li:nth-child(2) {color:orange;font-size:20px}
  </style>
</head>
<body>
  <ul>
    <li>Primeiro filho de ul</li>
    <li>Segundo filho de ul</li>
    <li>Último filho de ul</li>
  </ul>
  <p> primeiro parágrafo </p>
</body>
</html>
```

- Primeiro filho de ul
- Segundo filho de ul
- Último filho de ul

primeiro parágrafo

Pseudo-Class Selector

- *:first-of-type* ; *:last-of-type*; *:nth-of-type(n)*

```
<style>
  div:first-of-type{color:orange}
  div:last-of-type{color:lightblue}
</style>

</head>
<body>
  <p>primeiro filho</p>
  <div>primeiro elemento</div>
  <div>ultimo elemento</div>
  <p>ultimo filho</p>
```

primeiro filho

primeiro elemento

ultimo elemento

ultimo filho

- Apesar de na maioria das situações produzirem resultados idênticos, os seletores:

- *:first-of-type* e *:first-child*
- *:last-of-type* e *:last-child*
- *:nth-child(n)* e *:nth-of-type(n)*

não são equivalentes!

Seletor CSS

- Formulários
 - *Pseudo Class Selectors* **específicos** para formulários

<i>seletor</i>	<i>Exemplo</i>	<i>Observações</i>
<code>:checked</code>	<code>input:checked</code>	seleciona todos os input checked
<code>:disabled</code>	<code>input:disabled</code>	seleciona todos os input que estão inativos
<code>:enabled</code>	<code>input:enable</code>	seleciona todos os input que estão ativos
<code>:focus</code>	<code>input:focus</code>	seleciona todos os input que “ganham” <i>focus</i>
<code>:invalid</code>	<code>input:invalid</code>	seleciona todos os input inválidos
<code>:optional</code>	<code>input:optional</code>	seleciona todos os input não obrigatórios
<code>:required</code>	<code>input:required</code>	seleciona todos os input obrigatórios

- *Pseudo Class Selectors* específicos para formulários

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    input:disabled { background-color:lightcoral}
    input:focus {background-color:lightgoldenrodyellow}
    input:invalid {background-color:lightgray}
    input:required {background-color:lightgreen}
  </style>
</head>
<body>
  <form>
    <input type="text" size="30" placeholder="First Name" disabled/> <br/><br/>
    <input type="text" size="30" placeholder="Last Name" autofocus/><br/><br/>
    <input type="email" size="30" placeholder="Email"/><br/><br/>
    <input type="password" size="30" placeholder="Password" required/><br/><br/>
    <input type="submit" value="sign up" />
  </form>
</body>
</html>
```



- Lista completa de *pseudo class selectors*: http://www.w3schools.com/cssref/css_selectors.asp

Pseudo-Elements Selectors

Seletor CSS

- Referem-se a elementos fictícios (não correspondem a elementos HTML), os quais são baseados na estrutura do documento.
- Utilizam uma notação “::” diferente da pseudo-classe “:”
 - ::first-line**
 - ::first-letter**
 - ::before**
 - Com base na propriedade **content** permite inserir conteúdo antes de um elemento
 - ::after**
 - Com base na propriedade **content** permite inserir conteúdo depois de um elemento
 - ::selection**
 - Parte de um elemento que é seleccionada pelo utilizador.

Seletor CSS

- ::before ::after + propriedade content**

**** ANTES **** Pseudo elements (::before;::after) **** DEPOIS ****

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    p {color:gray;}
    span {color:orange;}
    p::before {content:"** ANTES **"}
    p::after {content:"** DEPOIS **"}
  </style>
</head>
<body>
  <p><span>Pseudo elements (::before;::after)</span></p>
</body>
</html>
```

■ **::selection**

- permite formatar o conteúdo que está a ser selecionado pelo utilizador

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    ::selection {background-color:orange;color:white; }
  </style>
</head>
<body>
  <p><span> Formatação diferente para uma seleção de texto </span></p>
</body>
</html>
```

Formatação diferente para uma seleção de texto

Conflitos de Formatação

CSS - Conflitos de Formatação

- Tipos de Conflito
 - Seletores iguais
 - Ordem pela qual são definidos
 - Seletores Diferentes
 - Especificidade
 - Herança

CSS - Conflitos de Formatação

- Seletores iguais
 - Ordem pela qual são definidos
 - *last one listed wins*: **prevalece a ultima definição** para **um determinado seletor**

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8" />
  <title> External Style Sheet</title>
  <style>
    h1{color:#F00;} /*red*/
    h1{color:#00F;} /*blue*/
  </style>
</head>

<body>
  <h1> Resolver Conflitos de Formatação </h1>
</body>
</html>
```

?

Resolver Conflitos de Formatação

CSS - Conflitos de Formatação

■ Seletores iguais

■ Ordem pela qual são definidos

- Um ficheiro externo foi incorporado **depois** da definição de um *embedded style*,

```
<style> ... </style>
<link ...>
```

em caso de conflito **para o mesmo seletor**, prevalece a formatação estabelecida no **ficheiro externo**.

- Um ficheiro externo foi incorporado **antes** da definição de um *embedded style*,

```
<link ...>
<style> ... </style>
```

em caso de conflito **para o mesmo seletor**, prevalece a formatação estabelecida no ***embedded style***.

CSS - Conflitos de Formatação

■ Conflitos

■ Seletores Iguais: Prioridade ?

- Critério: “Ordem pela qual são definidos”

1. Regra assinalada como **!important** [mais prioritário]

```
<style>
  h1{color:#F00 !important;}
</style>
```

!important

Prevalece sobre todas as formatações
É a **exceção** à regra da “ordem pela qual são definidos”

2. *Inline style* (atributo **style** na *opening tag*)

3. *Embedded Style Sheet* (**<style>...</style>**)

4. *External Style Sheet* (**<link .../>; @import**)

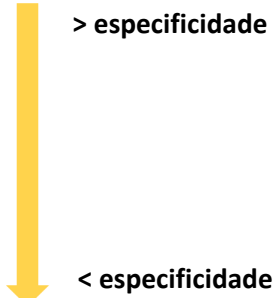
5. Definições por defeito do *browser* [menos prioritário]

Considera que o *embedded style* é declarado **após** a ligação ao ficheiro externo; caso contrário prevaleceria a declaração no ficheiro externo.

CSS - Conflitos de Formatação

■ Especificidade

- Os seletores **mais específicos tem mais peso** na resolução de conflitos de formatação:

1. ID Selectors (mais específico / mais prioritário)
 2. Class Selectors; Attribute Selectors
 3. Contextual Selectors (apenas com elementos)
 4. Individual Element Selectors
- 

- Entre seletores com **a mesma especificidade prevalece** a ultima declaração.

CSS - Conflitos de Formatação

■ Especificidade

- O browser **atribui pesos específicos** de acordo com o tipo de seletor
- Como calcular?
 - **d**: seletores de elemento e pseudo-elemento (**1 ponto**)
 - **c**: seletores de class, atributo e *pseudo-class* (**10 pontos**)
 - **b**: seletor de ID (**100 pontos**)
 - **a**: *inline style* (**1000 pontos**)

```
2 <style>  
  body h1{color:red;}  
1 h1{color:blue;}  
  </style>  
  </head>  
  
  <body>  
    <h1> Resolução de conflitos de formatação </h1>
```

Resolução de conflitos de formatação

CSS - Conflitos de Formatação

■ Conflitos de Formatação - Prioridades na aplicação de estilos

■ Especificidade

- Como calcular?

a	b	c	d

```
<style>
13  body div.especifica h1{color:darkolivegreen}
2   body h1{color:red}
1   h1 {color:blue}
</style>
</head>
<body>
  <div class="especifica">
    <h1 id="maisEspecifico">Resolução Conflitos de Formatação </h1>
  </div>
```

Resolução Conflitos de Formatação

CSS - Conflitos de Formatação

■ Conflitos de Formatação - Prioridades na aplicação de estilos

■ Especificidade

a	b	c	d

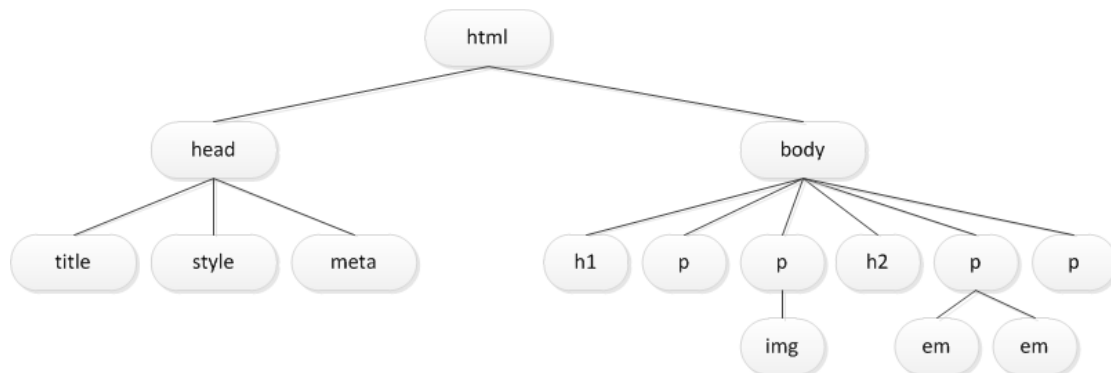
```
<style>
100 → #maisEspecifico{color:darkorange}
      body div.especifica h1{color:darkolivegreen}
      body h1{color:red}
      h1 {color:blue}
</style>
</head>
<body>
  <div class="especifica">
    <h1 id="maisEspecifico">Resolução Conflitos de Formatação </h1>
  </div>
```

Resolução Conflitos de Formatação

CSS - Conflitos de Formatação

■ Herança

- Documentos HTML tem uma estrutura hierárquica implícita



- body head filhos de html
- h1 p em h2 img descendentes de body
- ...

CSS - Conflitos de Formatação

■ Herança

- **Algumas** propriedades de **alguns** elementos HTML são herdadas
 - Geralmente as propriedades relacionadas com o estilo do texto são herdadas

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    p {color:gray;}
  </style>
</head>
<body>
  <p> Conceito Herança <span> formatação herdada de p </span></p>
</body>
</html>
```

Conceito Herança formatação herdada de p

O elemento é formatado por herança uma vez que não foi formatado diretamente.
Na realidade, apenas foi formatado o elemento pai <p>.

■ Herança

- Só é aplicada **caso não seja definido** o estilo do elemento.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    .p1 {color:gray;}
  </style>
</head>
<body>
  <p class="p1"> Conceito Herança <span> formatação herdada de p </span></p>
</body>
</html>
```

Conceito Herança formatação herdada de p

- Caso o elemento seja diretamente formatado, a herança não se aplica: **prevalece sempre a formatação direta do elemento**

Conceito Herança formatação herdada de p

```
<style>
  .p1 {color:gray;}
  span {color:orange;}
</style>
```

- Este funcionamento aplica-se sempre, mesmo que o seletor aplicado ao elemento tenha menor especificidade do que a especificidade do seletor aplicado ao elemento pai

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



em

Relativa ao font-size
definido para a fonte
default: 16px

rem

Relativa ao font-size
definido para a fonte do
root element <html>
default: 16px

%

Relativa às
dimensões do
elemento pai

vmin

Relativa a 1% da
menor dimensão do
viewport (mobile)

vmax

Relativa a 1% da
maior dimensão do
viewport (mobile)

vw

Relativa a 1% da
largura do viewport
(mobile)

vh

Relativa a 1% da
altura do viewport
(mobile)

■ 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 pixels físicos (dpi)

- **Device pixel ratio**: informação de que o browser necessita para determinar quantos pixels 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>
```

Block Element



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**)



■ Propriedades

Propriedade	Exemplo	Observações
padding <i>padding-top</i> <i>padding-right</i> <i>padding-left</i> <i>padding-bottom</i>	<code>p {padding:2em;}</code>	Permite definir a <i>padding area</i> . Valores: <i>length measurement, percentage, auto, inherit</i> .
margin <i>margin-top</i> <i>margin-right</i> <i>margin-left</i> <i>margin-bottom</i>	<code>p {margin:2em;}</code>	Permite definir a <i>margin area</i> . Valores: <i>length measurement, percentage, auto, inherit</i> . 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)

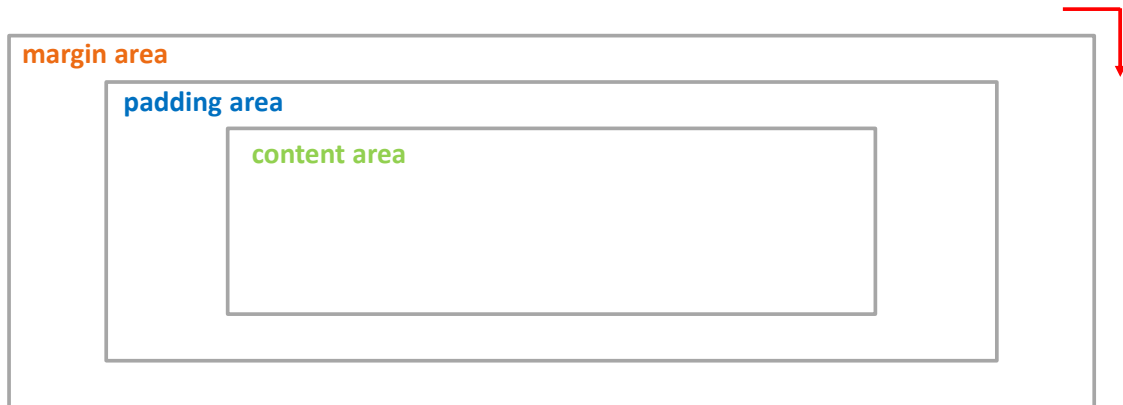


- *padding-top; padding-right; padding-bottom; padding-left*

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>
```

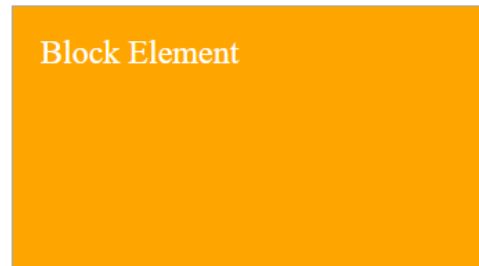
Block Element



Box Model

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

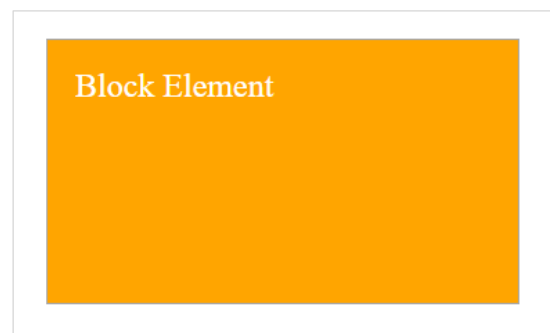
```
div{
  background-color: orange;
  color:white;
  font-size: 1.5em;
  width:300px;
  height:150px;
  padding:20px;
  border:darkgray 1px solid;
}
</style>
</head>
<body>
  <div> Block Element </div>
```



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

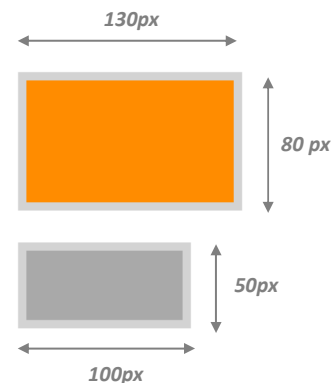
- Valor por default, os valores de width/height referem-se à área de conteúdo

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

Box Model

■ overflow

Propriedade	Exemplo	Observações
overflow	<code>p{overflow:scroll;}</code>	Valores: <i>visible, hidden, scroll, ...</i>

```
p{width:200px;
  height:100px;
  border: solid 3px lightsalmon;
}
</style>
</head>
<body>
```

```
<p> Propriedade Width permite definir a largura de um elemento.
Propriedade Width permite definir a largura de um elemento
Propriedade Width permite definir a largura de um elemento
Propriedade Width permite definir a largura de um elemento
Propriedade Width permite definir a largura de um elemento
</p>
```

Propriedade Width permite definir a largura de um elemento. Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento

CSS – Box Model

```
p{width:200px;
  height:100px;
  border: solid 3px lightsalmon;
  overflow:hidden
}
```

Propriedade Width permite definir a largura de um elemento. Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento

Propriedade Width permite definir a largura de um elemento. Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento Propriedade Width permite definir a largura de um elemento

```
p{width:200px;
  height:100px;
  border: solid 3px lightsalmon;
  overflow:scroll
}
```

Propriedade Width permite definir a largura de um elemento. Propriedade Width permite definir a largura de um elemento

■ 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>
```

Box Shadow

Propriedades

display

Block-level element

- A sua posição original é definida pelo fluxo do HTML
- Provoca uma quebra de linha
- Expande-se naturalmente para ocupar o seu *container*
 - A largura do elemento pode ser controlada (*width*)
 - 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
- Exemplos:
 - `<p>`, `<h1>`, `<div>`, `<form>`, ``, ``, ...



<https://dev.to/akshayjaagirdhar/differences-between-html-inline-and-blocks-elements-43d7>

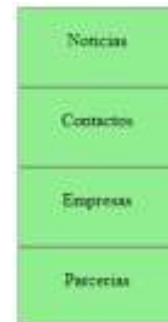
*{display:**}*

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

```
<a href="">Noticias</a>  
<a href="">Contactos</a>  
<a href="">Empresas</a>  
<a href="">Parcerias</a>
```

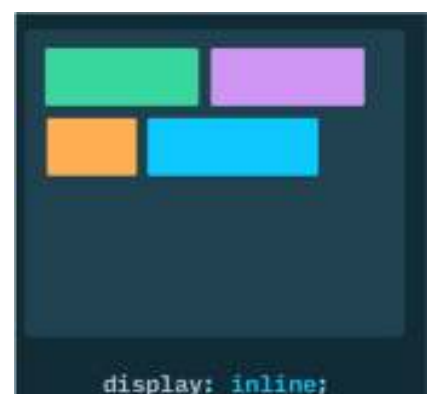
[Noticias](#) [Contactos](#) [Empresa](#) [Parcerias](#)

```
a{display:block;  
width:100px;  
height:30px;  
text-decoration:none;  
color:black;  
text-align:center;  
padding:20px;  
border:gray solid 1px;  
background-color: lightgreen;}
```



Inline-level Element

- Aplica-se a conteúdo textual
 - **Não se expande** à totalidade do espaço disponível
 - Container/viewport
- **Ignora** as propriedades:
 - largura e altura (width/height)
 - margens topo/fundo (*top/bottom*)
- Permite a definição de:
 - *left/right margin*
 - *padding*.
 - alinhamento vertical (*vertical-align*)
- Exemplos:
 - `<a>`, ``, ...



{display:***}

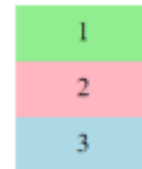
■ seletor{display:inline}

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

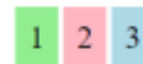
```
div{width:100px;
    height: 30px;
    font-size: 1.5em;
    padding:10px;
    text-align: center;}

#d1{background-color:lightgreen}
#d2{background-color:lightpink}
#d3{background-color:lightblue}
</style>
</head>
<body>

<div id="d1">1</div>
<div id="d2">2</div>
<div id="d3">3</div>
```



div{display:inline}

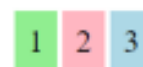


{display:***}

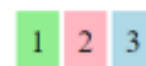
■ seletor{display:inline-block}

- preserva o comportamento inline mas com a possibilidade de definir largura altura do elemento

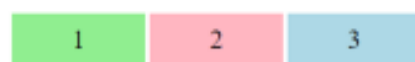
div{display:inline}



```
div{display:inline;
    width:200px;
    height:50px;}
```



```
div{display:inline-block;
    width:200px;
    height:50px;}
```



{display:***}

■ *display*

- propriedade muito importante nomeadamente na conceção de menus de navegação

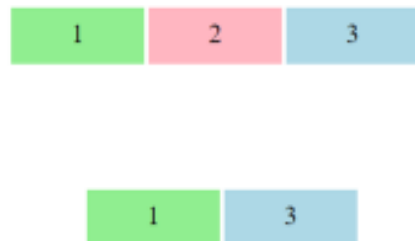
Propriedade	Exemplo	Observações
display	li {display:inline;}	Permite definir a disposição de elementos. Valores: none; inline; block; inline-block; flex; grid;...

{display:***}

■ `seletor{display:none}`

- O elemento não é visualizado **nem tem qualquer** influência no layout

```
div{display:inline-block;  
width:200px;  
height:50px;}  
#d2{display:none}
```



- lista completa dos valores propriedade ***display***

- http://www.w3schools.com/cssref/pr_class_display.asp

Exercício Seletores

Exercício Seletores

```
<p class="p1" >Parágrafo 1<span class="s1">inline element  
<span>parágrafo1</span></span></p>  
  
<p id="p2">Parágrafo 2<span>inline element parágrafo 2</span></p>  
  
<p>Parágrafo 3<span>inline element parágrafo 3</span> </p>
```

Exercício Seletores

```
<p class="p1" >Parágrafo 1<span class="s1">inline element  
<span>parágrafo1</span></span></p>  
  
<p id="p2">Parágrafo 2<span>inline element parágrafo 2</span></p>  
  
<p>Parágrafo 3<span>inline element parágrafo 3</span> </p>
```

Exercício Seletores

```
<p class="p1" >Parágrafo 1<span class="s1">inline element  
<span>parágrafo1</span></span></p>  
  
<p id="p2">Parágrafo 2<span>inline element parágrafo 2</span></p>  
  
<p>Parágrafo 3<span>inline element parágrafo 3</span> </p>
```

```
<p class="p1" >Parágrafo 1<span class="s1">inline element  
<span>parágrafo1</span></span></p>  
  
<p id="p2">Parágrafo 2<span>inline element parágrafo 2</span></p>  
  
<p>Parágrafo 3<span>inline element parágrafo 3</span> </p>
```

Layouts

Tipos

- É possível agrupar **layouts** da seguinte forma:

fluid layout
flexible layout

As dimensões são estabelecidas de forma a **existir um redimensionamento** dos conteúdos quando as dimensões do viewport são alteradas

Versatilidade!

Flexibilidade!

Mobile!

fixed layout

As dimensões são especificadas em px **sem atender** às dimensões do viewport ou do tamanho do texto

Deprecated!

Layouts

Tipos

Abordagens

■ CSS FlexBox

- Baseia-se num conjunto de propriedades que permitem a disposição flexível 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* (deprecated!)

- Implementação simples. Ligado ao fluxo do HTML o que reduz a sua flexibilidade

CSS Flexbox (Layouts)

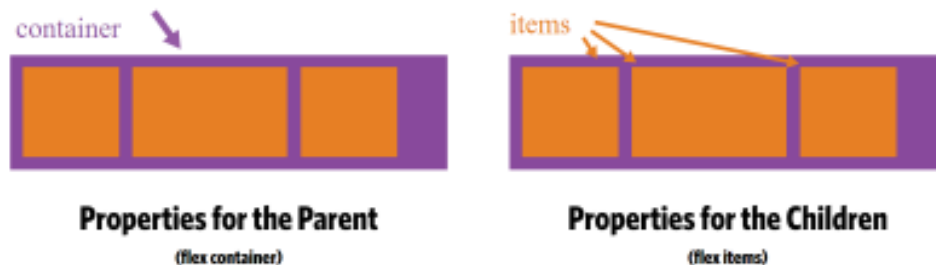
CSS Flexbox

■ 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)

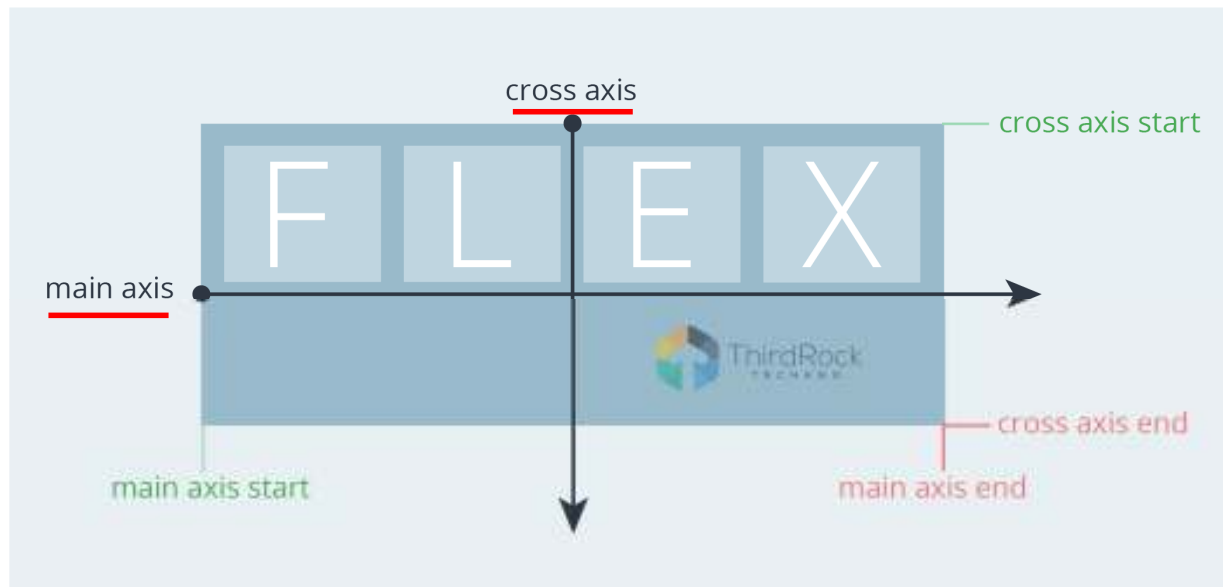
```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

```
ul{display:flex;}

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

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





<https://www.thirdrocktechkno.com/blog/how-to-improve-css-layout-with-flex/>

CSS Flexbox (Layouts)

flex container

flex item

■ 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**

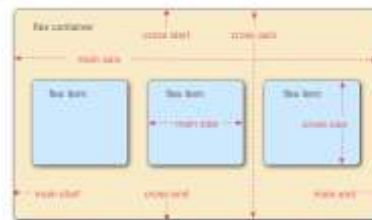
- define o alinhamento dos *items* ao longo do *cross axis*

■ **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*



■ Propriedades dos **flex container**

■ **flex-direction**: row | row-reverse | column | column-reverse



■ **flex-wrap**: nowrap | wrap | wrap-reverse

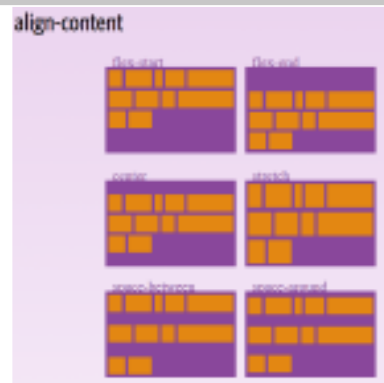
- O comportamento por defeito é nowrap, o flex container adapta-se aumentando de tamanho para conter todos os flex items
- O valor wrap cria múltiplas linhas



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



- **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/>

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;}
```

Item1	Item1	Item1	Item2	Item2	Item2	Item3	Item3	Item3
Item1	Item1	Item1	Item2	Item2	Item2	Item3	Item3	Item3
Item1			Item2			Item3		

- 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***)

Item1 Item1 Item1 Item1 Item1 Item1 Item1

Item2 Item2 Item2 Item2 Item2 Item2 Item2

Item3 Item3 Item3 Item3 Item3 Item3 Item3

CSS Flexbox (layout)

- 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;
}
```

Item1 Item1
Item1 Item1
Item1 Item1
Item1

Item2 Item2
Item2 Item2
Item2 Item2
Item2

Item3 Item3
Item3 Item3
Item3 Item3
Item3

CSS Flexbox (Layouts)

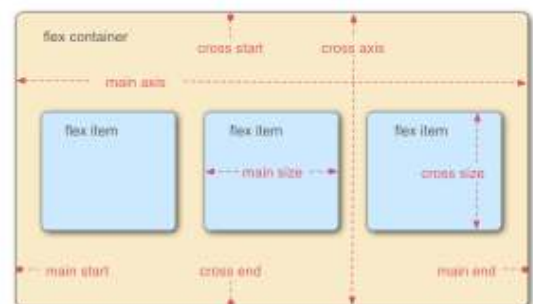
flex container

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*)



■ 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;}
```

■ {*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;}
```

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

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

CSS Flexbox (layout)

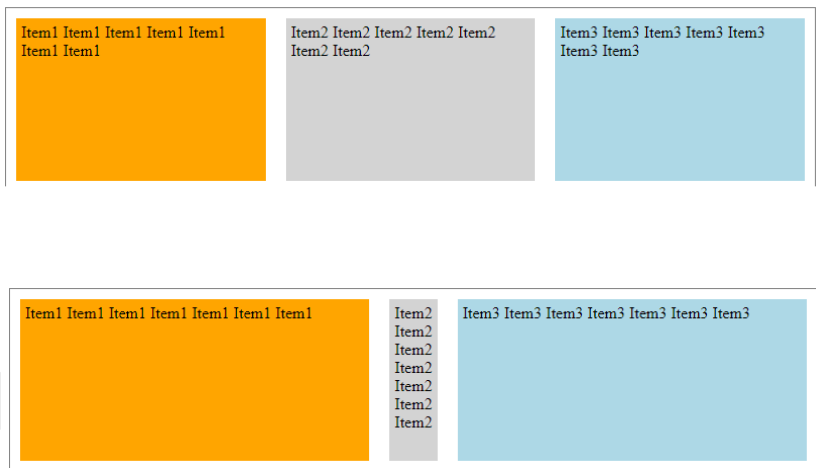
- `{flex-shrink: <integer>;}`
 - Capacidade de um item para reduzir a sua dimensão, o valor inteiro define o grau de redução atribuído a esse elemento.
 - Esta propriedade só se aplica em situação de **overflow** (largura items > largura do container)

```
#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;}
```

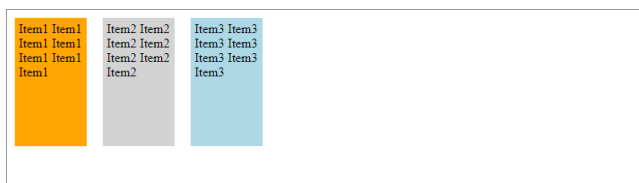
Como a largura está definida e existe uma situação de *overflow*, a dimensão dos *flex-items* irá ajustar-se à largura do *flex container*



CSS Flexbox (layout)

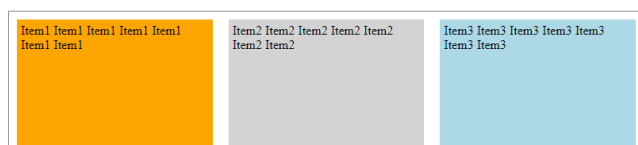
- `{flex-basis: <length> | auto;}`
 - Define a dimensão (% , em, ...) inicial de um flex item.

```
#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*

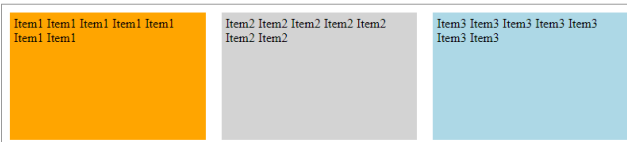
```
#container div{  
  height:150px;  
  margin:10px;  
  padding:5px;  
  flex-grow: 1;  
}  
  
#container div{flex-basis:10%;}
```



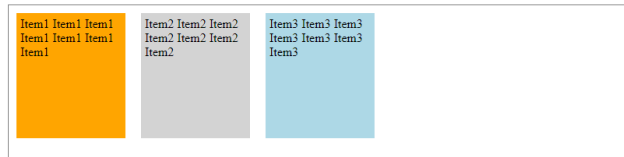
CSS Flexbox (layout)

- {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;}
```



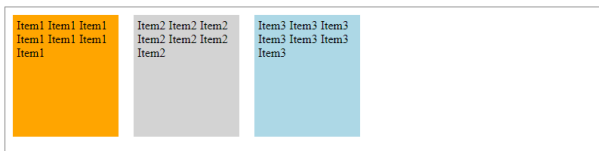
```
#container div{  
  height:150px;  
  margin:10px;  
  padding:5px;  
  /* flex-grow: 1; */  
  width: 130px;}  
  
#container div{flex-basis:auto;}
```



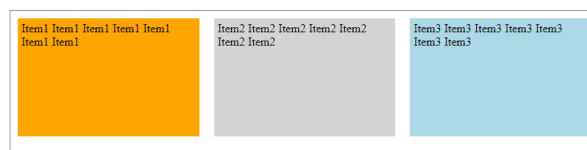
CSS Flexbox (layout)

- {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;}
```

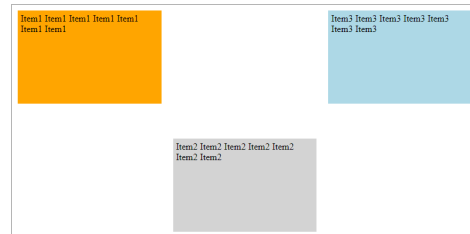


```
#container div{  
  height:150px;  
  margin:10px;  
  padding:5px;  
  width: 130px;}  
  
#container div{flex:1 1 auto;}
```



- **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*

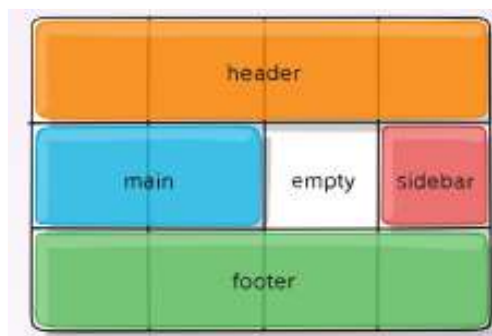
```
#container div{  
    height:150px;  
    margin:10px;  
    padding:5px;  
    width: 130px;}  
  
#container div{flex:1 1 auto;}  
  
#d2{align-self:flex-end}
```



CSS Grid Layout

CSS Grid Layout

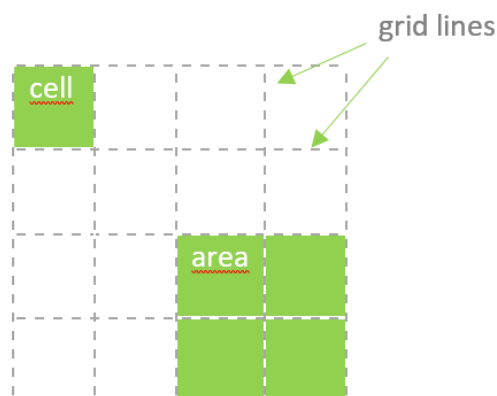
- A criação de um *grid container* é efetuada através de `{display: grid}`
 - Deve ser estabelecido o número de linhas/colunas do grid container
 - Todos os filhos **diretos** de um *grid container* tornam-se *grid items*
 - Associar cada *grid item* a uma área no *grid container*
 - pode ser criada uma grid dentro de uma outra grid e assim criar vários contextos de posicionamento



<https://css-tricks.com/snippets/css/complete-guide-grid/>

CSS Grid Layout

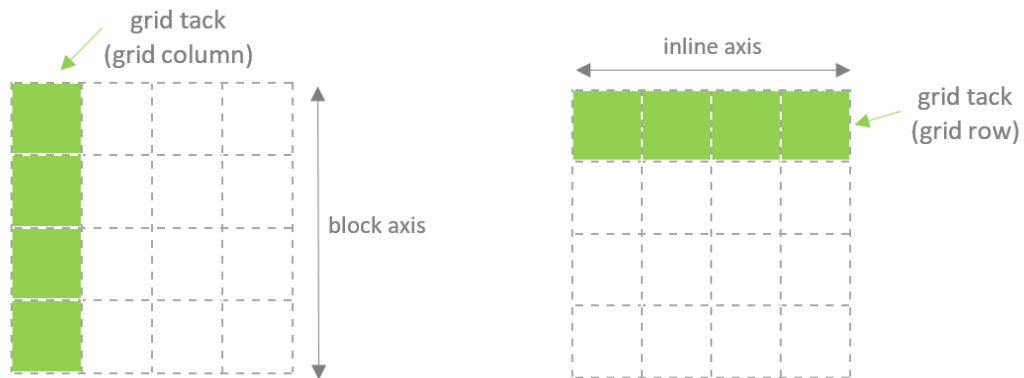
- CSS Grid
 - *line*
 - linhas horizontais e verticais que constituem a *grid*
 - *cell*
 - unidade mais pequena de uma *grid* a qual é delimitada por 4 *grid lines* adjacentes
 - *area*
 - área retangular constituída por uma ou mais *grid cells* adjacentes



■ CSS Grid

■ track

- o espaço entre células adjacentes é um *grid track*, o qual pode ser um *grid column* ou *grid row*
 - *grid column* é vertical (*block axis*)
 - *grid row* é horizontal (*inline axis*)



CSS Grid Layout

Grid Container

display: grid
grid-template-rows
grid-template-columns
grid-template-areas

Definição de um Grid Container

- A declaração de um CSS grid é feito através da propriedade **display:grid**
 - **grid-template-rows; grid-template-columns** definem n.º e dimensões de linhas/colunas

```
#layout{display: grid;
      grid-template-rows: 100px 400px 100px;
      grid-template-columns: 200px 500px 200px;}
</style>
</head>
<body>

<div id="layout">
  <div id="one">One</div>
  <div id="two">Two</div>
  <div id="three">Three</div>
  <div id="four">Four</div>
  <div id="five">Five</div>
</div>
```

Definidas 3 linhas e 3 colunas.
O *template* pode ser dividido no número de linhas e colunas definidos pelos valores atribuídos a **grid-template-rows;**
grid-template-columns

One	Two	Three
Four	Five	

Definição de um Grid Container

- A propriedade **grid-template-areas** permite identificar as várias *grid cells*
 - Importante!
 - Trata-se apenas da identificação das células uma vez que a grid e as respetivas dimensões das grid tracks são definidas através das propriedades **grid-template-rows, grid-template-columns**

```
#layout{display: grid;
      grid-template-rows: 100px 400px 100px;
      grid-template-columns: 200px 500px 200px;
      grid-template-areas:
        "header header header"
        "ads main links"
        "footer footer footer"}
</style>
</head>
<body>

<div id="layout">
  <div id="one">One</div>
  <div id="two">Two</div>
  <div id="three">Three</div>
  <div id="four">Four</div>
  <div id="five">Five</div>
  <div id="six">Six</div>
</div>
```

Células são referenciadas por **linha**

header	header	header
ads	main	links
footer	footer	footer

- Existe ainda a propriedade **grid** que agrega as propriedades:
 - grid-template-rows
 - grid-template-columns
 - grid-template-areas
- não é muito utilizada uma vez que origina uma sintaxe mais complexa do que a utilização das propriedades individuais:

```
#layout{display: grid;
  grid:
    "header header header" 100px
    "ads main links"       400px
    "footer footer footer" 100px
    / 200px 500px 200px
}
```

Especificação feita por **linha**
/ dimensão das colunas

CSS Grid Layout

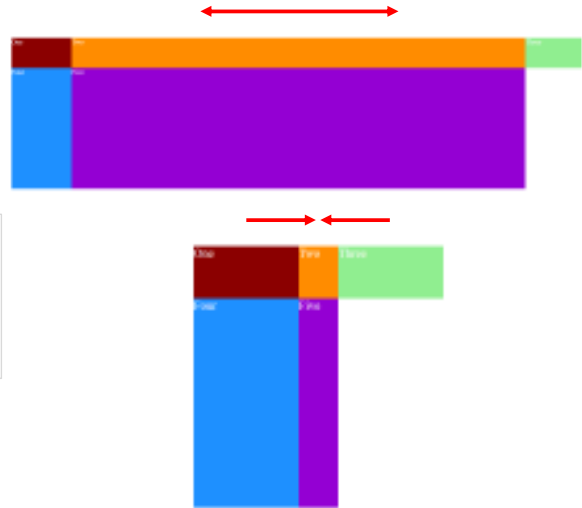
Grid Container

Unidades

Unidades

- Na CSS grid as unidades a utilizar devem assegurar a flexibilidade necessária para um *responsive web design* (diversidade de dispositivos)
 - px (fixed); %
 - fr (*fractional unit*)
 - permite que os grid tracks se ajustem ao espaço disponível, expandam/retraiam de acordo com as variações do *viewport*.

```
#layout{  
  display: grid;  
  grid-template-rows: 100px 400px 100px;  
  grid-template-columns: 200px 1fr 200px}
```

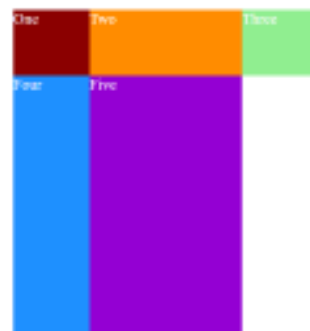


Unidades

- As *fractional units* podem ser usadas para:
 - combinar áreas de dimensão variável com dimensão fixa (exemplo anterior)
 - combinar áreas de dimensão variável

```
#layout{  
  display: grid;  
  grid-template-rows: 100px 400px 100px;  
  grid-template-columns: 1fr 2fr 1fr}
```

A coluna central terá sempre o dobro da capacidade de ajustamento das colunas dos extremos



- A função `minmax()` permite limitar as dimensões de uma grid track
 - tornar essa *grid track* flexível mas dentro de limites pré-definidos

```
#layout{
  display: grid;
  grid-template-rows: 100px 400px 100px;
  grid-template-columns: 200px minmax(5em,15em) 200px}
```

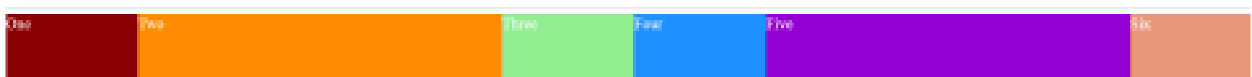
- As dimensões de uma grid track também podem ser definidas com base nas dimensões do seu próprio conteúdo
 - `min-content`
 - o menor valor para não criar situações de overflow (ex: a palavra mais longa; a imagem mais larga, ...)
 - `max-content`
 - o valor máximo que permita conter o conteúdo sem wrapping.
 - **Atenção!** Não é adequado para parágrafos, o efeito pode ser imprevisível.

- As dimensões da grid podem ser definidas tirando partido de:
 - `repeat (number_of_repetitions, pattern)`

```
#six{background-color:darksalmon}

#layout{display: grid;
  grid-template-rows: 100px 400px 100px;
  grid-template-columns: repeat(2, 200px 1fr 200px)}

</style>
</head>
<body>
  <div id="layout">
    <div id="one">One</div>
    <div id="two">Two</div>
    <div id="three">Three</div>
    <div id="four">Four</div>
    <div id="five">Five</div>
    <div id="six">Six</div>
  </div>
```



CSS Grid Layout

Grid Container

Unidades

Grid Items

Posicionamento Grid Items

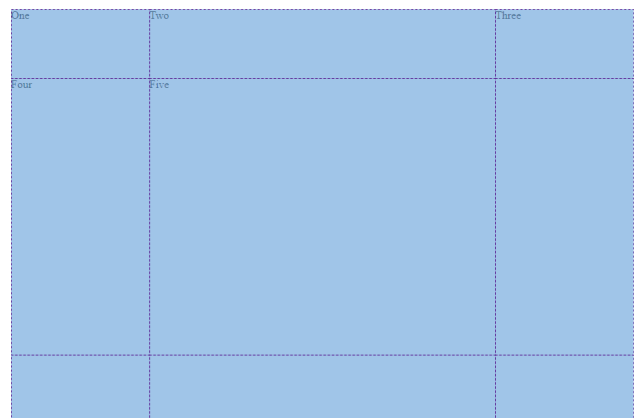
- Cada um dos *grid items* é atribuído de forma automática a cada uma das *grid cells*
 - por defeito essa atribuição é sequencial e feita por linha

```
#layout{display: grid;
  grid-template-rows: 100px 400px 100px;
  grid-template-columns: 200px 500px 200px;}

div{color:white; font-size: 1.2em}
#one{background-color:darkred}
#two{background-color:darkorange}
#three{background-color:lightgreen}
#four{background-color:dodgerblue}
#five{background-color:darkviolet}

</style>
</head>
<body>

  <div id="layout">
    <div id="one">One</div>
    <div id="two">Two</div>
    <div id="three">Three</div>
    <div id="four">Four</div>
    <div id="five">Five</div>
  </div>
```



CSS Grid Layout

Grid Container

Unidades

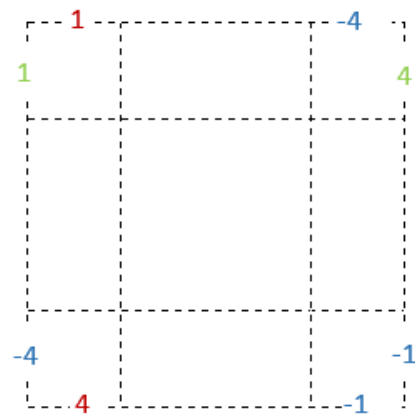
Grid Items : Posicionamento

Grid Items: Posicionamento

■ Identificação das linhas

■ Identificação numérica

- row line (1 – 4)
- column line (1-4)
- Contagem Inversa
 - a contagem inicia-se na ultima linha sempre com o valor -1 e processa-se de forma decrescente (-2, -3, ...)



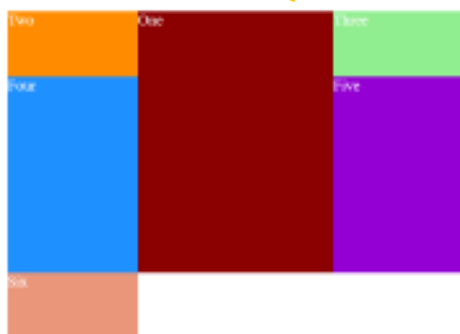
■ Identificação baseada em nomes [...]

```
#layout{display: grid;
  grid-template-rows: [header-start] 100px [content-start] 400px [footer-start] 100px;
  grid-template-columns: [adds] 200px [main] 500px [links] 200px}
```


Grid Items: Posicionamento

- A alteração do posicionamento é efetuado a partir da identificação das linhas tendo por base as propriedades:

- grid-row-start
- grid-row-end
- grid-column-start
- grid-column-end



```
#layout{display: grid;
  grid-template-rows: 100px 300px 100px;
  grid-template-columns: 200px 300px 200px;
  grid-template-areas:
    "header header header"
    "ads main links"
    "footer footer footer"
}

#one{
  grid-row-start: 1;
  grid-row-end:3;
  grid-column-start: 2;
  grid-column-end: 3;
}
```

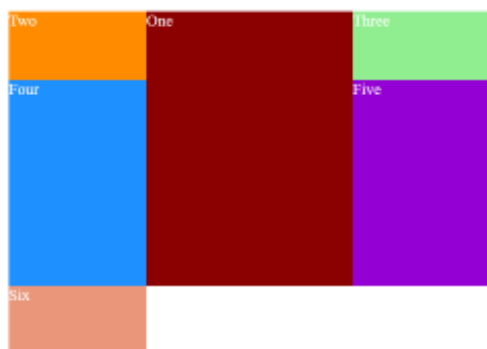
```
</style>
</head>
<body>
  <div id="layout">
    <div id="one">One</div>
```

O item é posicionado, sendo que todos os outros reajustam de acordo com o espaço disponível

O posicionamento é feito pelo número/nome da linha

Grid Items: Posicionamento

- O posicionamento anterior pode ser efetuado de forma mais condensada:
- grid-row: start line / end line
- grid-column: start line / end line



```
#layout{display: grid;
  grid-template-rows: 100px 300px 100px;
  grid-template-columns: 200px 300px 200px;
  grid-template-areas:
    "header header header"
    "ads main links"
    "footer footer footer"
}

#one{
  grid-row: 1 / 3;
  grid-column: 2 / 3;
}
```

```
</style>
</head>
<body>
  <div id="layout">
    <div id="one">One</div>
    <div id="two">Two</div>
    <div id="three">Three</div>
    <div id="four">Four</div>
    <div id="five">Five</div>
    <div id="six">Six</div>
  </div>
```

Grid Items: Posicionamento

- O posicionamento através de grid-area também pode ser feito com base na designação das áreas criadas através de **grid-template-areas**

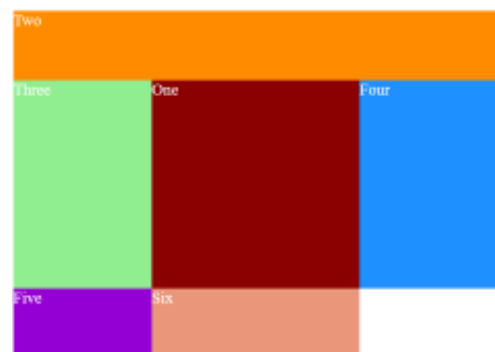
```
#layout{display: grid;
  grid-template-rows: 100px 300px 100px;
  grid-template-columns: 200px 300px 200px;
  grid-template-areas:
    "header header header"
    "ads main links"
    "footer footer footer"
}

#one{
  grid-area: main;
}

#two{
  grid-area: header;
}

</style>
</head>
<body>
  <div id="layout">
    <div id="one">One</div>
    <div id="two">Two</div>
    <div id="three">Three</div>
```

header	header	header
ads	main	links
footer	footer	footer



Grid Items: Posicionamento

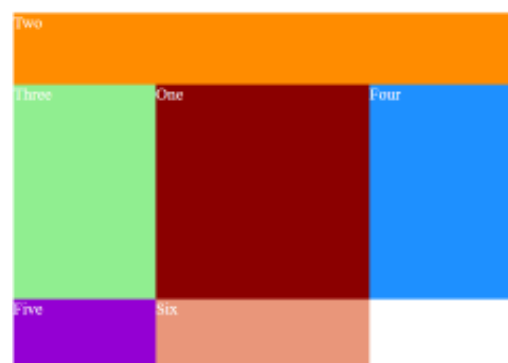
- A propriedade **grid-area** pode também ser aplicada com base em 4 grid lines as quais definem uma área:
 - `grid-area: row-start / column start / row-end / column-end;`

```
#one{
  grid-area: main;
}

#two{
  grid-area: 1 / 1 / 2 / 4;
}

</style>
</head>
<body>
  <div id="layout">
    <div id="one">One</div>
    <div id="two">Two</div>
    <div id="three">Three</div>
```

header	header	header
ads	main	links
footer	footer	footer



Grid Items: Posicionamento

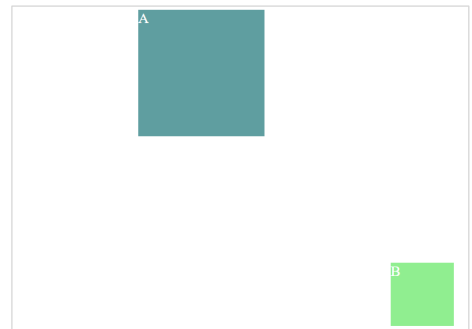
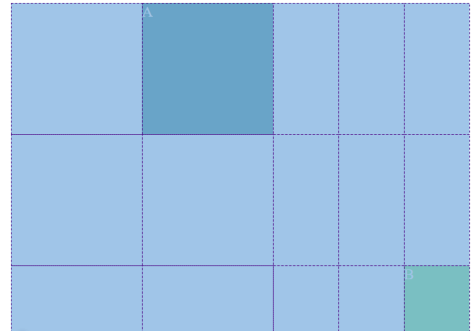
- Se o posicionamento for definido **fora da grid inicial** podem ser geradas automaticamente linhas/colunas para esse posicionamento ser possível
 - grid-auto-rows** e **grid-auto-columns** utilizadas para geração automática de *tracks* e aplicar à grid prédefinida

```
div{color:white; font-size: 1.2em}
#littlegrid{
  display: grid;
  grid-template-columns: 200px 200px;
  grid-template-rows: 200px 200px;
  grid-auto-columns: 100px;
  grid-auto-rows: 100px }

#a{
  grid-row: 1 / 2;
  grid-column: 2 / 3;
  background-color: cadetblue;}

#b{
  grid-row: 3 / 4;
  grid-column: 5 / 6;
  background-color: lightgreen; }

</style>
</head>
<body>
  <div id="littlegrid">
    <div id="a">A</div>
    <div id="b">B</div>
  </div>
```



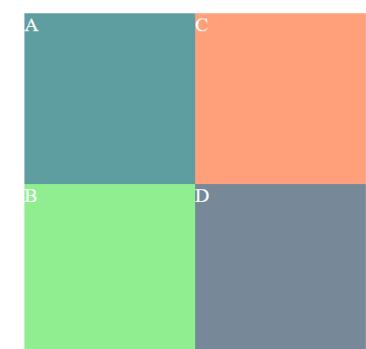
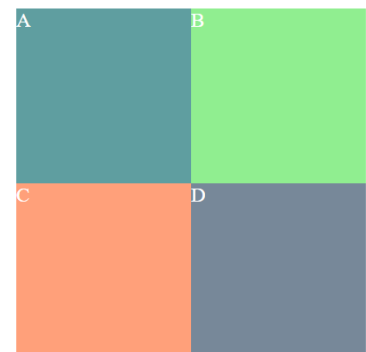
Grid Items: Posicionamento

- A propriedade **grid-auto-flow** permite estabelecer a direção do posicionamento (*row (default); column*)

```
div{color:white; font-size: 1.2em}
#littlegrid{
  display: grid;
  grid-auto-flow: column;
  grid-template-columns: 200px 200px;
  grid-template-rows: 200px 200px;
  grid-auto-columns: 100px;
  grid-auto-rows: 100px }

#a{background-color: cadetblue;}
#b{background-color: lightgreen; }
#c{background-color: lightsalmon}
#d{ background-color: lightslategrey }

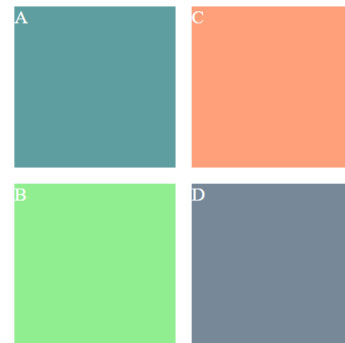
</style>
</head>
<body>
  <div id="littlegrid">
    <div id="a">A</div>
    <div id="b">B</div>
    <div id="c">C</div>
    <div id="d">D</div>
  </div>
```



Grid Items: Posicionamento

- As propriedades **grid-row-gap/grid-column-gap** permitem estabelecer um espaço entre tracks
 - **grid-gap** permite agregar estes dois valores por linha e coluna

```
#littlegrid{
  display: grid;
  grid-auto-flow: column;
  grid-gap: 20px;
  grid-template-columns: 200px 200px;
  grid-template-rows: 200px 200px;
  grid-auto-columns: 100px;
  grid-auto-rows: 100px
}
```



Grid Items: Posicionamento

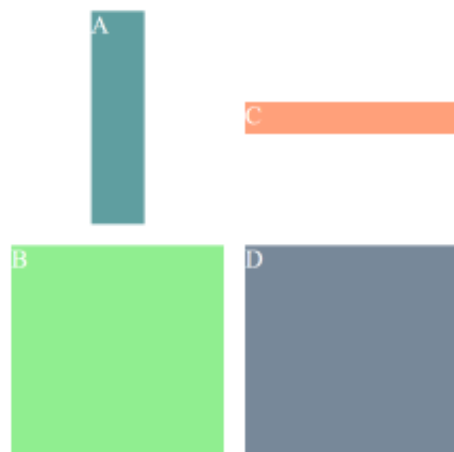
- Quando os items possuem dimensão inferior à grid que foi criada, podem ser alinhados (posicionados) tendo por base as propriedades:
 - **justify-self** (alinhamento horizontal)
 - **align-self** (alinhamento vertical)

```
#littlegrid{
  display: grid;
  grid-auto-flow: column;
  grid-gap: 20px;
  grid-template-columns: 200px 200px;
  grid-template-rows: 200px 200px;
  grid-auto-columns: 100px;
  grid-auto-rows: 100px }

#a{background-color: cadetblue;
  width:50px;
  justify-self: center;}

#b{background-color:lightgreen; }

#c{background-color:lightsalmon;
  height:30px;
  align-self:center}
```



Grid Items: Posicionamento

- A propriedade ***justify-items*** permite o alinhamento horizontal para todos os items

```
div{color:white; font-size: 1.2em;
width:50px;
height:30px}

#littlegrid{
display: grid;
grid-auto-flow: column;
justify-items: start;
grid-gap: 20px;
grid-template-columns: 200px 200px;
grid-template-rows: 200px 200px;}
```



```
#littlegrid{
display: grid;
grid-auto-flow: column;
justify-items: end;
grid-gap: 20px;
...}
```

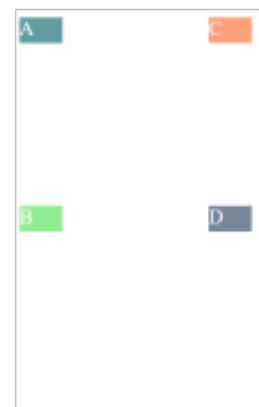


Grid Items: Posicionamento

- A propriedade ***align-items*** permite o alinhamento vertical para todos os items

```
div{color:white; font-size: 1.2em;
width:50px;
height:30px}

#littlegrid{
display: grid;
grid-auto-flow: column;
align-items: start;
grid-gap: 20px;
grid-template-columns: 200px 200px;
grid-template-rows: 200px 200px}
```



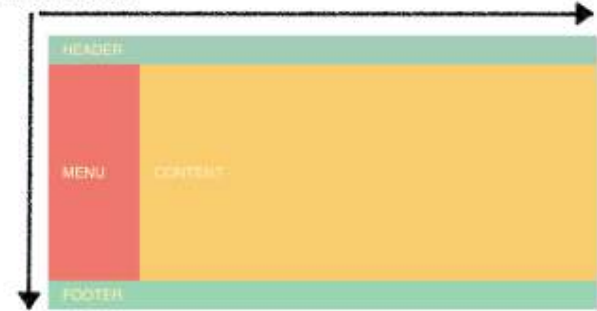
```
#littlegrid{
display: grid;
grid-auto-flow: column;
align-items: end;
...}
```



One dimension



Two dimensions



Flexbox – 1D

Mais flexível
Menos código
Mais fácil de manter

CSS Grid – 2D

Podem ser combinadas!
(ex: disposição de elementos no header/footer)

<https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf>

Framework CSS

layouts

■ Bootstrapp

- Instalação diretamente a partir de um *Content Delivery Network (CDN)*



<https://9official.com/2017/08/28/content-delivery-network-cdn/>

- Download rápido (possibilidade de *pre-cached files*) assim como controlo de versões

- <https://www.sitepoint.com/7-reasons-to-use-a-cdn/>

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" ></script>
```

■ Bootstrapp

All you need to do is learn the class names and apply them to your html markup. They'll do what they are designed to do.

<https://medium.freecodecamp.org/bootstrap-4-everything-you-need-to-know-c750991f6784>

The screenshot shows a Bootstrap Pricing table with three columns: Free, Pro, and Enterprise. Each column lists features and includes a call-to-action button. Below the pricing table, there are sections for Features, Resources, and About.

Free	Pro	Enterprise
\$0 / mo	\$15 / mo	\$29 / mo
10 users included 2 GB of storage Email support Help center access	20 users included 10 GB of storage Priority email support Help center access	30 users included 15 GB of storage Phone and email support Help center access
Sign up for free	Get started	Contact us

Features
Cool stuff
Random feature
Team feature
Stuff for developers
Another one
Last time

Resources
Resource
Resource name
Another resource
Final resource

About
Team
Locations
Privacy
Terms

<https://getbootstrap.com/docs/4.0/examples/>

Propriedades

display

[flexbox]

[CSS Grid]

float

*{float:***}*

■ **float**

- permite que um elemento seja movido o mais possível para a esquerda/direita e que seja envolvido pelo conteúdo que se lhe segue.

- Sem aplicação da propriedade *float*

```
<body>
  <div>FLOAT</div>
  <p>"Lorem ipsum dolor sit amet,
    consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
    ea commodo consequat.</p>
</body>
```



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

{float:***}

■ seletor {float:left}

```
div{
  width:70px;
  height:70px;
  background-color: orange;
  color:white;
  float:left
}
```

FLOAT

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

■ seletor {float:right}

```
div{
  width:70px;
  height:70px;
  background-color: orange;
  color:white;
  float:right
}
```

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

FLOAT

{float:***}

- CSS permitem aplicar a propriedade *float* a qualquer elemento (**block ou inline**)
 - **float** aplicado a *inline elements*

```
p{width:300px}
span{
  float:right;
  width:100px;
  margin:10px;
  background-color:lightgreen;
}
```

Exemplo de um elemento com inline com float

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged

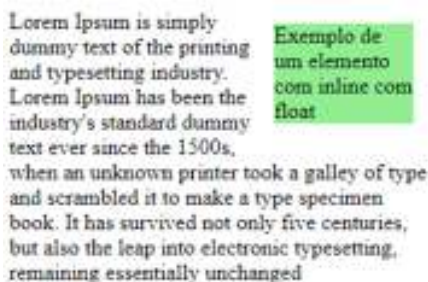
```
<p><span>Exemplo de um elemento com inline com float</span>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged</p>
```

■ **float** aplicado a *inline elements*

- Um elemento posicionado com *float* não pode ser posicionado antes da sua posição natural no código fonte.

■ **Importante!**

- Todos os **floated elements** comportam-se como **block elements**.
- Sempre que se pretende efetuar o float de **elementos de texto** é necessário especificar a **width do elemento**, caso contrário toda a largura disponível é aproveitada.



■ Posicionamento flutuante

- Atualmente assume uma importância cada vez mais reduzida, mas por vezes ainda é utilizado


Propriedade	Exemplo	Observações
float	.paragrafo {float:left;}	Permite definir elementos com posicionamento flutuante. Valores: <i>left, right, none, inherit</i>
clear	.paragrafo {clear:both;}	Especifica em que lado do elemento não são permitidos elementos com posicionamento flutuante Valores: <i>left, right, both, none, inherit</i>

clear

- Sempre que se pretende interromper o efeito da propriedade float deve ser aplicada a propriedade *clear*


```
div{ width:100px; height:150px; color:white;font-size:1.5em; text-align: center}
#global{width:600px;height:500px}
#d1{background-color:lightgray; float:left}
#d2{background-color: darkorange; float:right}

#footer{ color:darkgray; text-align:left}
</style>
</head>
<body>
  <div id="global">
    <div id="d1">1</div>
    <div id="d2">2</div>
    <p id="footer">Lorem ipsum dolor sit amet,
      consectetur ... </p>
  </div>
```



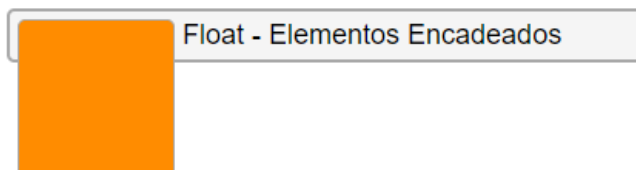
- interromper o efeito *float* (ex: posicionar um elemento de rodapé)

```
#footer{ clear: both; color:darkgray; text-align:left}
</style>
</head>
<body>
  <div id="global">
    <div id="d1">1</div>
    <div id="d2">2</div>
    <p id="footer">Lorem ipsum dolor sit amet,
      consectetur </p>
  </div>
```



clearfix

- *floats* em elementos encadeados
 - Quando a altura do container não é pré-definida:
 - O elemento interior é posicionado sem que o container acompanhe as suas dimensões



Altura não
especificada

```
.wrapper {
  max-width: 600px;
  margin: 40px auto;}

.container {
  border: 2px solid darkgray;
  border-radius: 5px;
  background-color: whitesmoke;
  width: 400px;
  padding: 5px;}

.item {
  height: 100px;
  width: 100px;
  margin-right: 5px;
  background-color:darkorange;
  border: 1px solid darkgray;
  border-radius: 5px;
  float: left;}

</style>
</head>
<body>

  <div class="wrapper">
    <div class="container">
      <div class="item"></div>
      float - Elementos Encadeados
    </div>
```

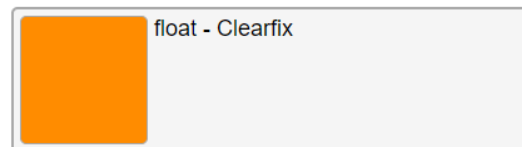
- Para resolver esta situação foi proposta uma solução (*clearfix*):

```

.item {
    height: 100px;
    width: 100px;
    margin-right: 5px;
    background-color:darkorange;
    border: 1px solid darkgray;
    border-radius: 5px;
    float: left;}

.container2::after {
    content: "";
    display: block;
    clear: both;}
</style>
</head>
<body>
    <div class="wrapper">
        <div class="container container2">
            <div class="item"></div>
            float - Clearfix
        </div>

```



Imediatamente após o <div> com class container 2, é adicionado **um elemento vazio**, o qual é definido como **block element** e sobre o qual é aplicada a propriedade **clear**

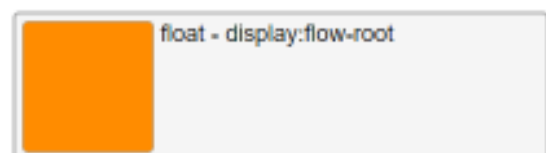
- Recentemente foi criado um novo valor para a propriedade *display* que origina um efeito semelhante ao *clearfix* anterior.

```

.item {
    height: 100px;
    width: 100px;
    margin-right: 5px;
    background-color:darkorange;
    border: 1px solid darkgray;
    border-radius: 5px;
    float: left;}

.container3 {
    display: flow-root;
}
</style>
</head>
<body>
    <div class="wrapper">
        <div class="container container3">
            <div class="item"></div>
            float - display:flow-root
        </div>

```



fluid Layout (float)

- Deprecated?
 - O CSS FlexBox e CSS GRID apresentam-se como alternativas muito mais flexíveis e vantajosas
 - A propriedade float era utilizada como base para a construção de layouts

```
#wrapper{width:100%;  
    margin:0px auto;  
    font-size: 25px;}  
#header,#footer{width:calc(100% - 4%);  
    height:25px;  
    padding:2%;  
    background-color: lightgray;}  
#nav{float:left;  
    width:20%;  
    height:500px;  
    padding:1%;  
    margin:1%;  
    background-color:orange;}  
#main{float:left;  
    width:72%;  
    height:500px;  
    padding:1%;  
    margin:1%;  
    background-color:lightblue;}  
#footer{clear:both;}
```

```
<div id="wrapper">  
    <div id="header">Header</div>  
    <div id="nav">Nav</div>  
    <div id="main">Main</div>  
    <div id="footer">Footer</div>  
</div>
```



Propriedades

display

position

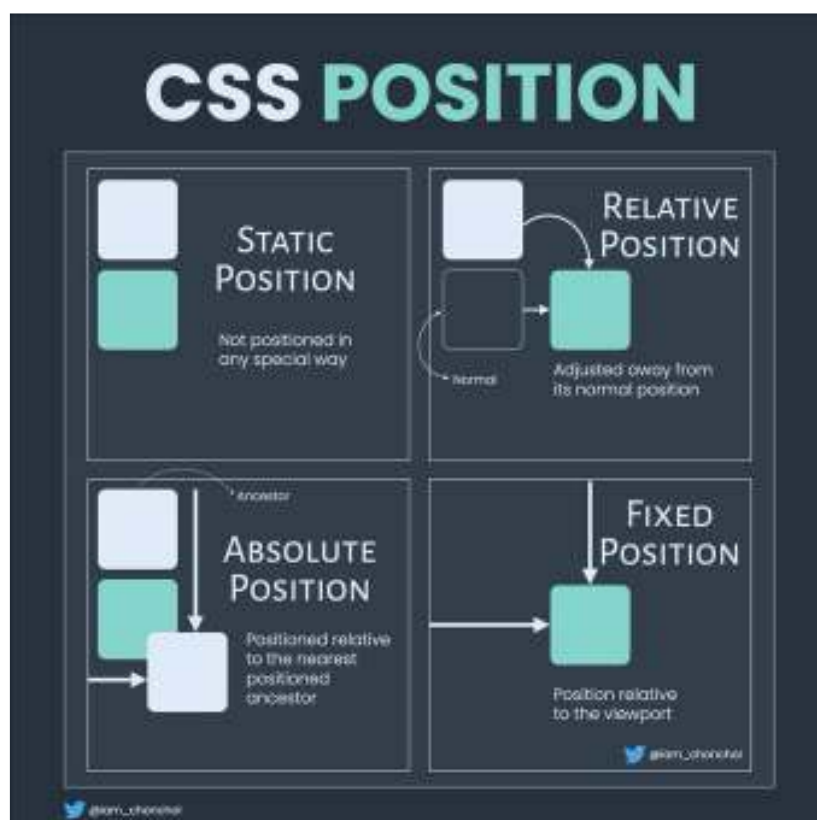
[flexbox]

[CSS Grid]

float

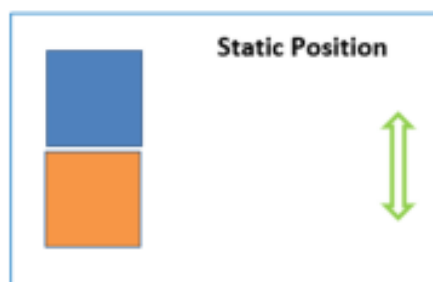
■ Posicionamento

Propriedade	Exemplo	Observações
position	div{position:absolute}	<p>Permite definir o tipo de posicionamento de um elemento.</p> <p>Valores:</p> <ul style="list-style-type: none">static*relativeabsolutefixedinheritsticky



Posicionamento static

- `{position:static}` (default)
 - Os elementos são dispostos pela ordem que são definidos (top->bottom; left->right);
 - *Block elements*
 - Dispostos sequencialmente na vertical (a partir do topo)
 - Ocupam o espaço disponível na janela do browser ou definido pelo respetivo *container*
 - *Inline elements*
 - dispostos alinhados de forma a preencher os *block elements*
 - Não provoca nenhuma alteração ao posicionamento dos elementos



<https://www.csssolid.com/css-position.html>

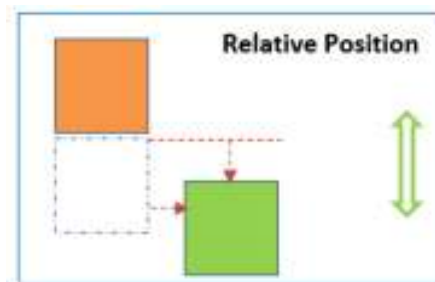
Posicionamentos ≠ static

- Os posicionamentos **relativo; absoluto e fixo** baseiam-se em deslocamentos a partir das referências: *top, left, right, bottom*
 - Nestes casos a propriedade *position* requer a especificação de propriedades de deslocamento:

Propriedade	Exemplo	Observações
top ↓	div{top:30px;left:50px;}	Permite definir os deslocamentos relativamente às referências, as quais dependem do valor especificado em position Valores: <i>length measurement, percentage, auto, inherit</i>
right ←		
bottom ↑		
left →		

Posicionamento Relativo

- Um elemento com `{position:relative}` posiciona-se em relação à sua **posição original** no fluxo HTML:
 - Os elementos com posicionamento relativo conservam o seu espaço original, assim:
 - Afetam o fluxo do HTML
 - Os elementos colocados depois das camadas relative, terão em conta as dimensões dos elementos anteriores
 - Os valores de posicionamento (top, left, ...) não afetam os elementos seguintes.



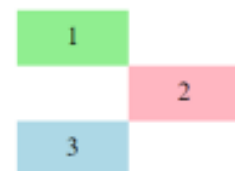
Posicionamento Relativo

- seletor `{position:relative}`
 - block-level elements*

```
<div id="d1">1</div>
<div id="d2">2</div>
<div id="d3">3</div>
```

```
div{width:100px;
height: 30px;
font-size: 1.5em;
padding:10px 0px;
text-align: center;}
#d1{background-color:lightgreen}
#d2{background-color:lightpink;
position:relative;
top:0px;
left:100px;}
#d3{background-color:lightblue}
</style>
```

A referência para os deslocamentos é a posição original do elemento deslocado.



O posicionamento relativo **preserva** o espaço original:

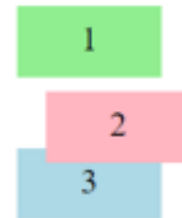
3 é posicionado tendo em conta o espaço originalmente ocupado por **2**

Posicionamento Relativo

- seletor `{position:relative}`
 - *block-level elements*

Sobreposição de elementos

```
#d2{background-color:lightpink;
    position:relative;
    top:10px;
    left:20px;}
#d3{background-color:lightblue}
```



O posicionamento de **3** não considera valores de top/left de **2** o que pode originar sobreposições

Posicionamento Relativo

- seletor `{position:relative}`
 - *inline-level elements*

A referência para os deslocamentos é a posição original do elemento deslocado.

```
<style>
  span{
    position:relative;
    top:50px;
    left:30px;
    background-color: orange;
    color:white;
  }
</style>
</head>
<body>
```

```
  <p>Posicionamento Relativo: <span>SPAN</span>sofreu um deslocamento.</p>
```

Posicionamento Relativo: sofreu um deslocamento.

SPAN

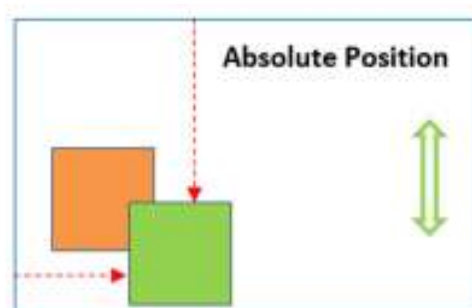
- O posicionamento relativo pode originar a sobreposição de elementos.

```
<style>
  span{
    position:relative;
    left:60px;
    background-color: orange;
    color:white;}
</style>
```

Posicionamento Relativo: sofSPAN deslocamento.

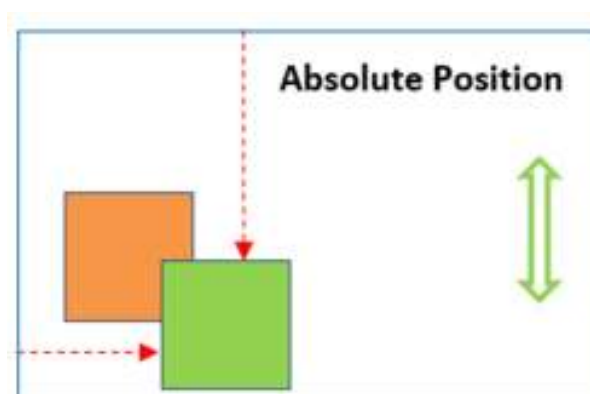
Posicionamento Absoluto

- O posicionamento absoluto é efetuado relativamente:
 - Ao *body* ou ao *container* mais próximo do elemento a deslocar.
 - `container ≠ <body>`
 - Se o elemento a posicionar estiver contido num elemento com **position ≠ static** então:
 - O elemento será posicionado relativamente a esse *container*
 - Caso contrário, é posicionado relativamente ao *body*



Posicionamento Absoluto

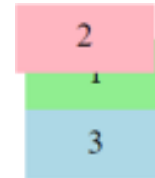
- `<body>`
 - Se o elemento a posicionar não está contido em outro elemento com **position ≠ static** então será posicionado relativamente ao `<body>` (diferente do posicionamento relativo ao *viewport*)



Posicionamento Absoluto

- **{position: absolute}**
 - A posição original do elemento **não é preservada, assim** o espaço original ocupado pelo elemento deslocado é preenchido pelos elementos que se lhe seguem no ficheiro HTML.
 - Criado um novo contexto de posicionamento

```
#d1{background-color:lightgreen}  
  
#d2{background-color:lightpink;  
  position: absolute;  
  top:0px;  
  left:0px;}  
  
#d3{background-color:lightblue}  
</style>
```



Neste caso o div 3 ocupa a posição de 2 uma vez que este elemento foi posicionado de forma absoluta

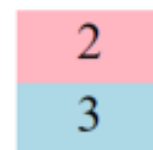
Posicionamento Absoluto

- Referência de posicionamento: container ≠ <body>

```
<div id="ref">  
  <div id="d1">1</div>  
  <div id="d2">2</div>  
  <div id="d3">3</div>  
</div>
```

A referência de posicionamento é #ref uma vez que é container dos div interiores e a sua posição é ≠ static

```
#ref{position: relative;}  
  
#d1{background-color:lightgreen}  
  
#d2{background-color:lightpink;  
  position: absolute;  
  top:0px;  
  left:0px;}  
  
#d3{background-color:lightblue}
```



Posicionamento Absoluto

- {position:**absolute**}
- *inline elements*

```
span{ position:absolute;
      color: darkorange;
      font-size: 1.5em;
      top:50px;
      left:50px}

</style>
</head>
<body>
  <p> Position Absolute: <span> elemento </span> posicionado de forma absoluta </p>
```

Position Absolute: posicionado de forma absoluta
↙
elemento

<body> é a referência para o deslocamento uma vez que o container <p> tem position:static

Posicionamento Absoluto

- {position:**absolute**}
- *inline elements*
 - *container* do elemento deslocado utilizado como referência para os deslocamentos

```
span{ position:absolute;
      color: darkorange;
      font-size: 1.5em;
      top:50px;
      left:50px}
p{position:relative}
</style>
</head>
<body>
  <p> Position Absolute: <span> elemento </span> posicionado de forma absoluta </p>
```

Position Absolute: posicionado de forma absoluta
↙
elemento

o posicionamento do *container* <p> foi alterado, o que origina que passe a ser a nova referência para os deslocamentos

Posicionamento fixo

- `{position:fixed}`
 - o posicionamento é efetuado relativamente ao *viewport* (área visível da janela do browser)
 - O posicionamento do elemento relativamente à área visível permanece inalterado mesmo quando é feito um *scroll* da página (ex: menu de navegação, ...)



Posicionamento *sticky*

- `{position:sticky}`
 - o elemento a posicionar, ao contrário do position fixed, acompanha o scroll da página até uma posição limite definida.

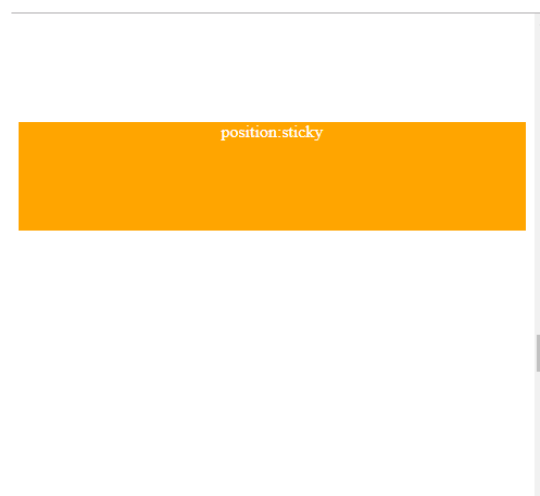
```
#d1{
  height:100px;
  position:sticky; top:100px;
  background-color: orange;
  color:white;
  text-align: center; }
</style>
</head>
<body>
  <div id="global">

    <br>... <br>

    <div id="d1">position:sticky</div>

    <br>...<br>

  </div>
</body>
```



Sobreposição de Elementos

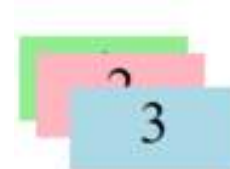
Propriedade	Exemplo	Observações
z-index	<code>div {z-index:5}</code>	Permite definir porque ordem são sobrepostos os elementos. Para valores positivos, quanto maior o valor mais à frente será posicionado o elemento (<i>bring to front</i>), a situação oposta ou a especificação de valores negativos enviar o elemento para trás (<i>send to back</i>)

```
#d1{background-color:lightgreen;  
  z-index:1;}
```

```
#d2{background-color:lightpink;  
  position:absolute;  
  top:10px;  
  left:10px;  
  z-index:2;}
```

```
#d3{background-color:lightblue;  
  position:absolute;  
  top:30px;  
  left:30px;  
  z-index:3;}
```

```
<div id="ref">  
  <div id="d1">1</div>  
  <div id="d2">2</div>  
  <div id="d3">3</div>  
</div>
```



Exemplo

Funcionamento submenu (display/position)

submenu (controlo de funcionamento)

■ submenu.html

```
<h2>Submenu mouseover</h2>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>
```

Submenu mouseover

Dropdown

■ estilos.css

```
.dropbtn {
  background-color: darkorange;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
  cursor: pointer;}
```

Formatação do botão

```
.dropdown {
  position: relative;
}
```

Referência de posicionamento
do submenu

submenu (controlo de funcionamento)

■ submenu.html

```
<h2>Submenu mouseover</h2>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>
```

Submenu mouseover

Dropdown

■ estilos.css

```
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  z-index: 1;
}
```

Este div fica oculto e posicionado de forma absoluta
(permite ajustar a posição do submenu)

```
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}
```

Formatação dos links, que são interpretados
como block level elements

```
<h2>Submenu mouseover</h2>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>
```

Submenu mouseover

Dropdown

Submenu mouseover

Dropdown

Link 1

Link 2

Link 3

■ estilos.css : efeito hover

```
.dropdown:hover .dropbtn {
  background-color: lightsalmon;
}

.dropdown:hover .dropdown-content {
  display: block;
}

.dropdown-content a:hover {background-color:lightyellow}
```

Propriedades

display

position

[flexbox]

visibility

[CSS Grid]

float

■ Controlar a visibilidade de um elemento

Propriedade	Exemplo	Observações
visibility	li {visibility:hidden;}	Permite definir a visibilidade de um elemento. Valores: hidden, visible,

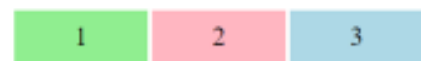
■ seletor{visibility:hidden}

- O elemento não é visualizado mas ao contrário do {display:none} mantém o espaço original

```
div{width:100px;
height: 30px;
font-size: 1.5em;
padding:10px;
text-align: center;}

#d1{background-color:lightgreen}
#d2{background-color:lightpink}
#d3{background-color:lightblue}
</style>
</head>
<body>

<div id="d1">1</div>
<div id="d2">2</div>
<div id="d3">3</div>
```



```
div{display:inline-block;
width:200px;
height:50px;}

#d2{visibility: hidden;}
```

