

CSS

Cascading Style Sheets

CSS

- Incremento da qualidade e consistência gráficas dos web sites.
- Separação do **design e do conteúdo**:
 - **HTML** define a estrutura (conteúdo)
 - **CSS** define a formatação/design (cor, tipo de fonte, ...)



■ 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

Sintaxe CSS

- Cada regra **selecciona** um elemento e **declara** como deve ser mostrado

```
seletor { propriedade:valor;}
```

- Composta por:

- Seletor
- Declaração /conjunto de declarações

- **Seletor**

- Elemento crucial na definição de uma regra CSS
- Permite identificar os elementos HTML a que a regra CSS se aplica



Sintaxe CSS

- **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: *font*, *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;}

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



- Ficheiro Externo *.CSS

- Sintaxe CSS

seletor{propriedade:valor;}

```
/* style1.css */

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

- O código CSS **não contém tags**

▪ External Style Sheet <link ... />

- Atributo **href** indica a localização do ficheiro *.css
- Atributo **rel** define a relação do documento HTML com o ficheiro externo, neste caso é *stylesheet*
- Atributo **type** é opcional no HTML5. Em versões anteriores (X)HTML é obrigatório.

```
<!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:
 - Assume-se que a tag **<link>** é a forma mais eficaz de efetuar ligações a CSS externas

■ **Embedded Style Sheet**

- Necesita de ser definido em todos os documentos *.html
 - Menos flexível e eficaz do que a ligação a um ficheiro externo
 - Uma alteração na formatação implica uma alteração em cada um dos ficheiros



■ **Embedded Style Sheet**

- `<style> ... </style>`
 - Atributo `type="text/css"` é obrigatório nas versões (X)HTML anteriores ao HTML5.

```
<!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>
```

■ *Inline Style*

- aplicar o estilo localmente através do atributo **style**
- **Deve ser evitado!**
 - É a forma menos flexível e mais propensa a erros para formatar um elemento HTML
 - 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>
```

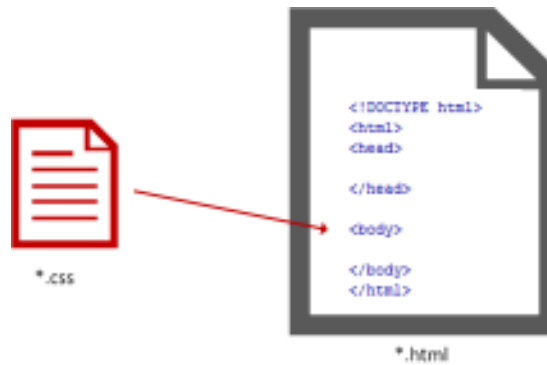
Inline Style: A evitar!

Seletores

Seletor CSS

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

seletor {propriedade:valor;}



Seletor CSS

- Seletor de **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>
```

Mesma formatação a h1 e h2
Através de selectores agrupados

Seletor CSS

■ Seletor Universal

■ *

- Permite selecionar todos os elementos

```
<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

■ Seletor de Contexto

■ Descendentes

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

```
<!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> <span>parágrafo</span></span></p>
  <p><span>2º</span> <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> <span>parágrafo</span></span></p>
  <p><span>2º</span> <span>parágrafo</span></span></p>
</body>
</html>
```

Referência

1º parágrafo

2º parágrafo

Seletor CSS

Seletores de Contexto

$element1 \sim element2$

- 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

Seletor CSS

Selectores de ID (#)

- Permite atribuir um estilo a uma única ocorrência de um elemento HTML

```
<!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>parágrafo</span></span></p>
  <p><span>2º </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

- Seletor de *class*.
 - Ao contrário do ID, a mesma *class* pode ser partilhada por múltiplos elementos
 - Um elemento pode ter definida mais de uma *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>
  ...
```

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

heading 1
heading 2
paragraph1
Other heading 2

≠

A ordem dos elementos é
determinante para uma correta
implementação de um seletor

```
...
<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
definidos **no**
contexto de um
outro elemento
cuja **class** é **c1**

heading 1
heading 2
paragraph1
Other heading 2

Seletor CSS

■ Attribute Selectors

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

■ ***element [attribute]***

- referencia os elementos 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

■ ***[attribute]***

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

```
<!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

- ***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 *= " ..."] / [attribute *= " ..."]**
 - 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** (nesta caso o element <body>)

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

- Os *pseudo class selectors* *:first-of-type* e *:first-child*, apesar de na maioria das situações produzirem resultados idênticos, não são equivalentes.

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

Seletor CSS

■ *Pseudo Class Selectors* específicos para formulários

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

The image shows a visual representation of the HTML form defined in the code. It consists of four input fields and a submit button. The 'First Name' field is disabled and has a light coral background. The 'Last Name' field is focused and has a light goldenrod yellow background. The 'Email' field is invalid and has a light gray background. The 'Password' field is required and has a light green background. The 'sign up' button is a standard gray button.

- 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**

```
<!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>
```

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

■ **::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

■ Exemplo:

- Um ficheiro externo foi incorporado depois da definição de um *embedded style*, em caso de conflito **para o mesmo seletor**, prevalece a formatação estabelecida no ficheiro externo.

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**)

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

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

CSS - Conflitos de Formatação

■ Especificidade

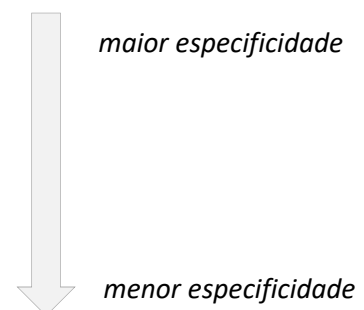
- Os seletores **mais específicos tem mais peso** na resolução de conflitos de formatação:

1. *ID Selectors* (mais específico / mais prioritário)

2. *Class Selectors; Attribute Selectors*

3. *Contextual Selectors*

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

```
13 <style>  
  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

100

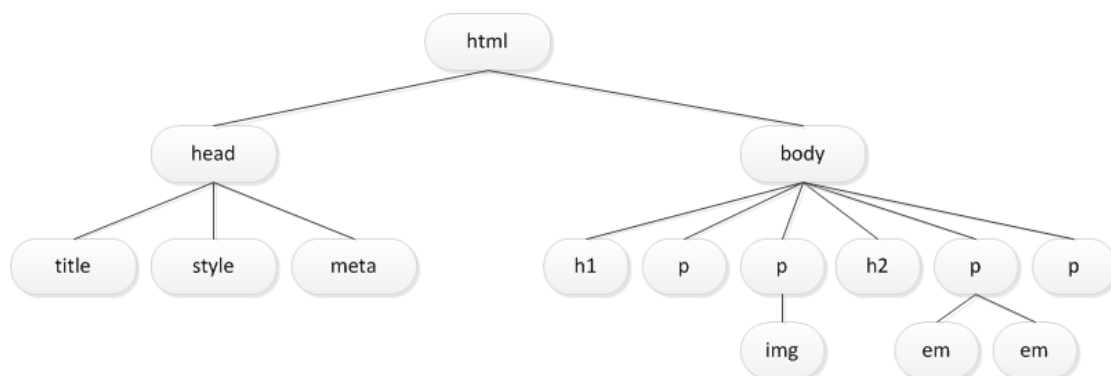
```
<style>
  #maisEspecifico{color:darkorange}
  body div.especifica h1{color:darkolivegreen}
  body h1{color:red}
  h1 {color:blue}
</style>
</head>
<body>
  <div class="especifica">
    <h1 id="maisEspecifico">Resolução Conflitos de Formatação </h1>
  </div>
```

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. Foi formatado o elemento pai <p>.

CSS - Conflitos de Formatação

■ 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

- mesmo que o seletor aplicado ao elemento tenha menor especificidade que o a especificidade do seletor aplicado ao pai, a herança não se aplica **prevalecendo a formatação direta do elemento**

Conceito Herança formatação herdada de p

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


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
 - Por *default*, 1em corresponde a um *font-size* de 16px

2em


2 em


Unit	Description	
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	
ex	Relative to the x-height of the current font (rarely used)	
ch	Relative to width of the "0" (zero)	
rem	Relative to font-size of the root element	
vw	Relative to 1% of the width of the viewport*	
vh	Relative to 1% of the height of the viewport*	
vmin	Relative to 1% of viewport's* smaller dimension	
vmax	Relative to 1% of viewport's* larger dimension	
%		

Unidades CSS

■ Unidades Absolutas

- Tem uma correspondência/significado físico
- Este tipo de unidade **deve ser evitado** dada a diversidade das dimensões dos ecrãs.

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

■ px*

http://www.w3schools.com/cssref/css_units.asp

- *Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.*

http://www.w3schools.com/cssref/css_units.asp

Unidades CSS

■ CSS pixel

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

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

- O CSS pixel pode ou não coincidir com os pixéis físicos (*dpi*)
 - **Device pixel ratio**: informação de que o browser necessita para determinar quantos pixéis físicos são utilizados para desenhar um único CSS pixel.
 - Exemplo: Os *Retina Display* da Apple possuem um *device pixel ratio* de 2
1 CSS pixel é formado por 4 pixéis físicos (2 altura/2 largura)



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

*“the units have nothing to do with properties, but **everything** with the output media: **screen or paper.**”*

	Recommended	Occasional use	Not recommended
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	

<https://www.w3.org/Style/Examples/007/units.en.html>

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.

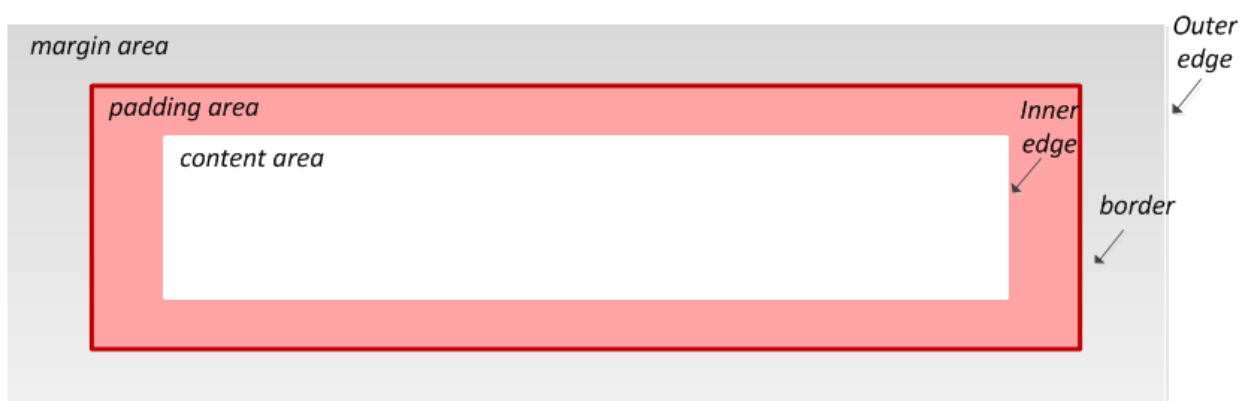
```
div{
  background-color: orange;
  color:white;
  font-size: 1.5em;
  width:300px;
  height:150px;
  border:darkgray 1px solid;
}
</style>
</head>
<body>
  <div> Block Element </div>
```

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**)



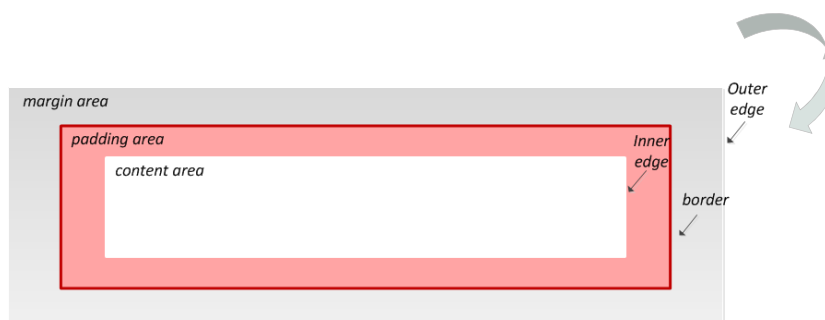
■ Propriedades

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</i> , <i>percentage</i> , <i>auto</i> , <i>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</i> , <i>percentage</i> , <i>auto</i> , <i>inherit</i> . Nota importante: a inserção de valores negativos conduz geralmente à sobreposição dos elementos.

Box Model

■ *padding*

- `seletor{padding:5px;}` (top/bottom/right/left)
- `seletor{padding:5px 10px;}` (top/bottom right/left)
- `seletor{padding:5px 10px 15px 5px;}` (top right bottom left)

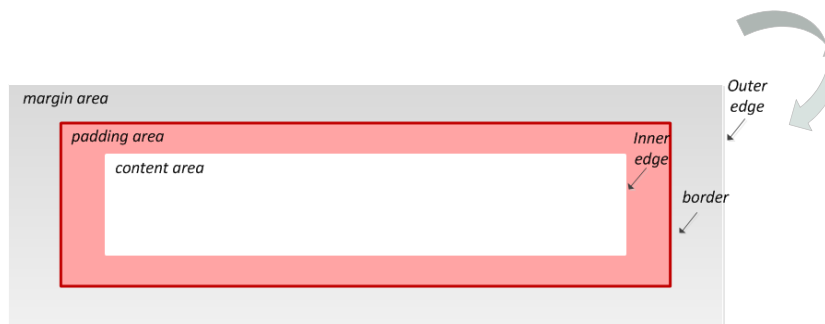


- *padding-top; padding-right; padding-bottom; padding-left*

Box Model

■ *margin*

- `seletor{margin:5px;}` (top/bottom/right/left)
- `seletor{margin:5px 10px;}` (top/bottom right/left)
- `seletor{margin:5px 10px 15px 5px;}` (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



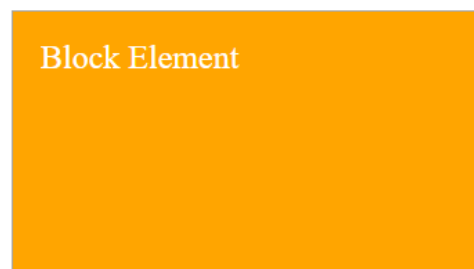
Box Model

■ *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>
```



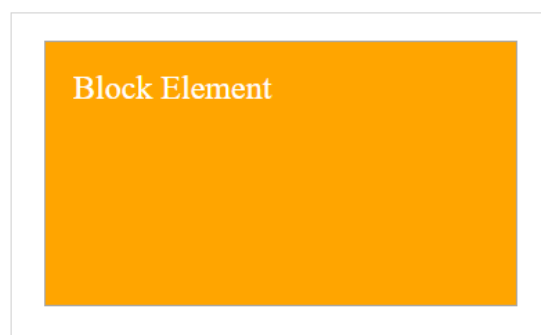
Box Model

■ *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

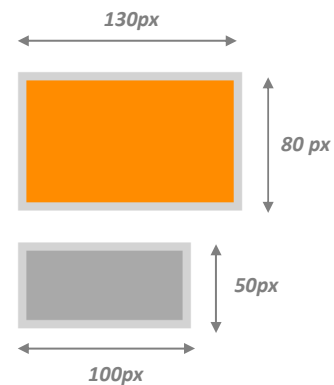
- Valor por default, os valores de width/height referem-se à área de conteúdo

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

Box Model

■ overflow

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

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

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

CSS – Box Model

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

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

Propriedade
width permite
definir a

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

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



■ 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>
```

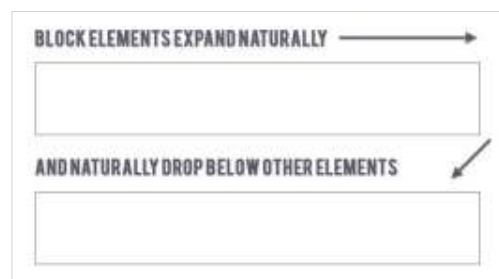
Box Shadow

Propriedades

display

Block-level element

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



Block-level element

■ Exemplo: <div> expansão do elemento

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

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

<div> é um **block level element**

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

div1

div2

div3

Block-level element

■ Exemplo: <div> propriedade width

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

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

<div> é um **block level element**

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

div1

div2

div3

Inline-level Element

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

INLINE ELEMENTS FLOW WITH TEXT

PELLENTESQUE HABITANT MORBITRISTIQUE SENECTUS
ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS.
VESTIBULUM **INLINE ELEMENT** VITAE, ULTRICIES
EGET, TEMPOR SIT AMET, ANTE. DONEC ULIBERO SIT
AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI
VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

{display:***}

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

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

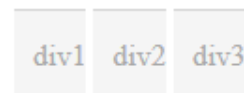
{display:***}

■ seletor{display:inline}

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

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

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



{display:***}

■ seletor{display: block}

- Define o elemento como um **block element**
- Muito útil para a implementação de menus verticais (*vertical navigation bar*)

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

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



{display:***}

■ seletor{display:inline-block}

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



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

```
a{ font:Verdana;
font-size: 1.1em;
display:block;
height:20px;
width:150px;
text-decoration: none;
color:darkgray;
padding-left:10px;
border-top:1px lightgray solid;
border-bottom:1px lightgray solid;
border-radius: 1px;
padding-top: 15px;
padding-bottom: 15px;
background-color: rgb(245, 245, 245);}

a.c1{display:inline-block;
height:40px;
margin-bottom: 20px}

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

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

{display:***}

■ seletor{display:none}

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

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

</style>

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



- lista completa dos valores propriedade **display**

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

Propriedades

display

visibility

*{visibility:****}*

■ Controlar a visibilidade de um elemento

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

`{visibility:***}`

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

```
a{
  font:Verdana;
  font-size: 1.1em;
  display:block;
  height:20px;
  width:150px;
  text-decoration: none;
  color:rgb(120,120,120);
  padding-left:10px;
  border-top:1px lightgray solid;
  border-bottom:1px lightgray solid;
  border-radius: 1px;
  padding-top: 15px;
  padding-bottom: 15px;
  background-color: rgb(245, 245, 245);}
a.c1{display:inline-block;
      height:40px;
      margin-bottom: 20px}
a.c2{visibility:hidden}

</style>
</head>
<body>

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



Propriedades

display

visibility

float

■ Posicionamento flutuante

- Assume uma importância elevada no contexto das CSS

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>

■ propriedade **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**

```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut e.... </p>
```

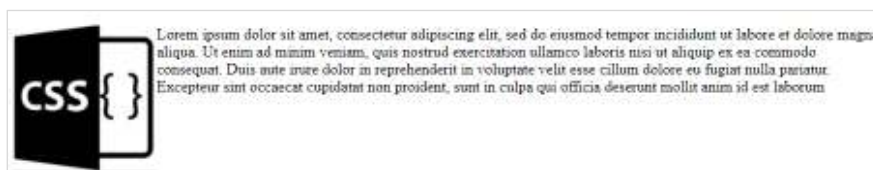


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

{float:***}

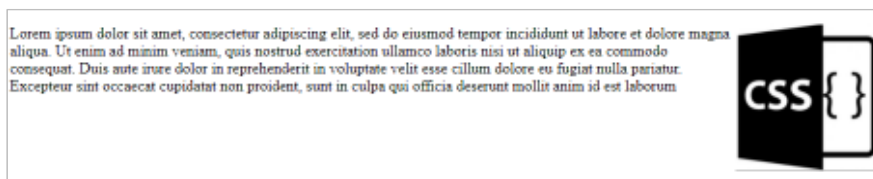
■ seletor {float:left}

```
img{ width:150px;  
     height:150px;  
     float:left;}
```



■ seletor {float:right}

```
img{ width:150px;  
     height:150px;  
     float:right;}
```



{float:***}

■ CSS permitem aplicar a propriedade *float* a qualquer elemento (**block ou inline**)

■ *float* aplicado a *inline elements*

```
...  
<style>  
  span {float:right;  
        width:200px;  
        margin:10px;  
        color:white;  
        background-color:gray;  
        padding:4px; }  
  p {padding:15px;  
     background-color:lightgray;  
     border: 2px darkgray solid; }  
</style>  
</head>  
<body>  
  <p><span>Exemplo de elemento inline ao qual é aplicado um posicionamento float</span>  
    Texto do parágrafo que envolve o span.Texto do parágrafo que envolve o span.  
    ...  
</p>
```

Texto do parágrafo que envolve o span.Texto do parágrafo que envolve o span. Texto do parágrafo que envolve o span. Texto do parágrafo que envolve o span. Texto do parágrafo que envolve o span. Texto do parágrafo que envolve o span. Texto do parágrafo que envolve o span.

Exemplo de elemento inline ao qual é aplicado um posicionamento float

Importante: todas as *float*ed elements comportam-se como **block elements**.

Assim, sempre que se pretende efectuar o *float* de **elementos de texto** é necessário especificar a **width**, caso contrário toda a largura disponível é aproveitada

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

{float:***}

■ **float** (múltiplos elementos)

■ Construção de menus horizontais/barras de navegação

```
...
    ul {list-style-type:none;
        margin:0px;padding:0px;}

    ul li {float: left;}

    ul li a {display:block;
        width:100px;
        background-color:lightcoral;
        color:white;
        padding:10px;
        margin:5px;
        text-align:center;
        text-decoration:none;
        border-top-left-radius:15px;
        border-top-right-radius:15px;}

</style>
</head>
<body>
    <ul>
        <li><a href="#">1ª opção</a></li>
        <li><a href="#">2ª opção</a></li>
        <li><a href="#">3ª opção</a></li>
        <li><a href="#">4ª opção</a></li>
    </ul>
```

Mostrar o *inline element* <a> como um *block element*, de forma a permitir aplicar algumas formatações (ex: width)



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:lightgrey; 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>
```



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

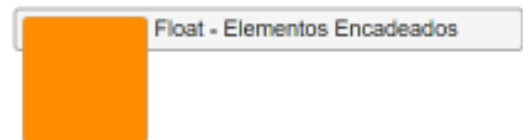
```
.wrapper {
  max-width: 600px;
  margin: 40px auto;}

.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>
```



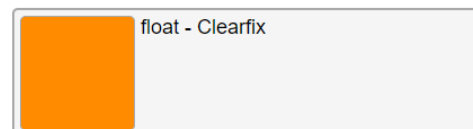
- Para resolver esta situação foi proposta uma solução (*clearfix*):

```
.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>
```



- 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>  
  </div>  
</body>  
</html>
```



Propriedades

display

visibility

float

position

■ Posicionamento

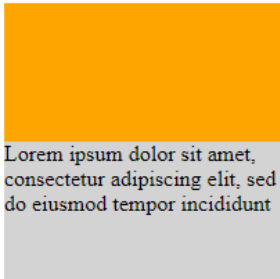
Propriedade	Exemplo	Observações
position	div{position:absolute}	Permite definir o tipo de posicionamento de um elemento. Valores: static*, relative, absolute, fixed, inherit, sticky

Posicionamento static

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

```
...  
    div{width:200px;  
        height:100px;}  
    .c1{background-color:orange;}  
    .c2{background-color:lightgray;}  
</style>  
</head>  
  
<body>  
    <div class="c1"></div>  
    <div class="c2">  
        <span>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt</span>  
    </div>  
...  

```



- Os posicionamentos **relativo; absoluto e fixo** baseiam-se em deslocamentos a partir das referências: *top, left, right, bottom*
 - Assim, nestes dois 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 <i>position</i> 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*

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <style>

    div.d1 {width:50px;height:50px}
    #box1 {background-color:lightgray;position:static}
    #box2 {background-color:orange;
           position:relative;
           top:10px;
           left:50px;}
    #box3 {background-color:darkgray;position:static}
  </style>
</head>
<body>
  <div class= "d1" id="box1"></div>
  <div class= "d1" id="box2"></div>
  <div class= "d1" id="box3"></div>
</body>
</html>
```

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



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

“box3” é posicionado tendo em conta o espaço originalmente ocupado por “box2”

Posicionamento Relativo

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

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    div.d1 {width:50px;height:50px}
    #box1 {background-color:lightgray;position:static}
    #box2 {background-color:orange;
           position:relative;
           top:30px;
           left:10px;}
    #box3 {background-color:darkgray;position:static}
  </style>
</head>
<body>
  <div class= "d1" id="box1"></div>
  <div class= "d1" id="box2"></div>
  <div class= "d1" id="box3"></div>
</body>
</html>
```

Sobreposição de elementos



O posicionamento de “box3” não considera valores de top/left de “box2” **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.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    span {position:relative;
          top:50px;
          left:20px;
          background-color:lightcoral;
          color:white;}
  </style>
</head>
<body>
  <p>Posicionamento Relativo <span>SPAN</span> sofreu um deslocamento.</p>
</body>
</html>
```

Posicionamento Relativo

sofreu um deslocamento.

SPAN

- O posicionamento relativo pode originar a sobreposição de elementos.

```
span {position:relative;
      left:60px;
      background-color:lightcoral;
      color:white;}
```

Posicionamento Relativo

sofSPAN deslocamento.

Posicionamento Absoluto

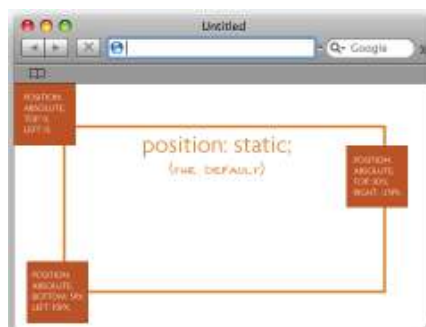
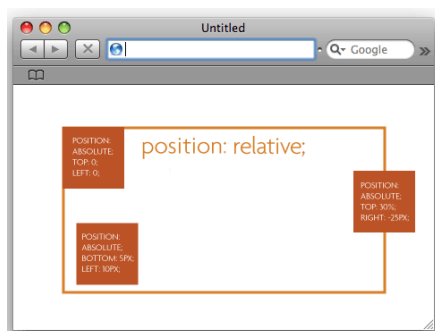
- O posicionamento absoluto é sempre efetuado relativamente ao *container* mais próximo do elemento a deslocar, tendo em consideração que:

- container \neq `<body>`

- Se o elemento a posicionar está contido em outro elemento com **position:static** então o elemento será posicionado relativamente a esse *container*

- `<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*)



<https://css-tricks.com/absolute-positioning-inside-relative-positioning/>

Posicionamento Absoluto

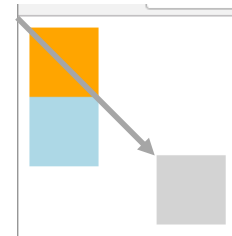
■ {position: absolute}

- 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.

```
...
div{width:50px;
    height:50px;}

.c1{background-color:orange;}
.c2{background-color:lightgray;
    position:absolute;
    top:100px;
    left:100px;
}
.c3{background-color: lightblue;}
</style>
</head>

<body>
    <div class="c1"></div>
    <div class="c2"></div>
    <div class="c3"></div>
    ...
```



Neste caso o div cuja class é c3 ocupa a posição de .c2 uma vez que este elemento foi posicionado de forma absoluta

Posicionamento Absoluto

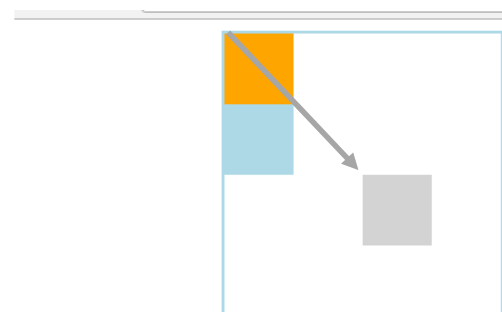
```
#global{ position:absolute;
width:200px;
height:200px;
left:150px;
border:2px solid lightblue;}

div{width:50px;
    height:50px;}

.c1{background-color:orange;}
.c2{background-color:lightgray;
    position:absolute;
    top:100px;
    left:100px;}
.c3{background-color: lightblue;}
</style>
</head>

<body>
    <div id="global">
        <div class="c1"></div>
        <div class="c2"></div>
        <div class="c3"></div>
    </div>
    ...
```

Referência de posicionamento diferente de <body>



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

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, ...)

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #div1 {width:200px;
           height:100px;
           position:fixed;
           top:100px;
           left:100px;
           background-color:lightcoral;color:white;text-align:center;}
    #div2 {float:right;width:100px;}
  </style>
</head>
<body>
  <div id="div1">FIXED</div>
  <div id="div2"><br/>...</div>
</body>
</html>
```



Posicionamento *sticky*

■ {position:*sticky*}

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

```
div.sticky {
  position: sticky;
  top: 100px;
  padding: 5px;
  background-color: lightcoral;
  border: 2px solid lightcoral;
  color:white;
}

</style>

</head>

<body>

<br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br>

<div class="sticky">Sticky</div>

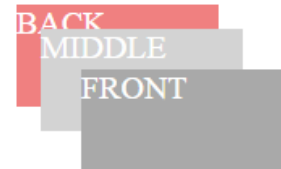
<br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br>
```



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>)

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {width:100px;height:50px;color:white}
    #div1 {background-color:lightcoral;
      z-index:1;}
    #div2 {background-color:lightgray;
      z-index:2;
      position:absolute; top:20px;left:20px;}
    #div3 {background-color:darkgray;
      z-index:3;
      position:absolute; top:40px;left:40px;}
  </style>
</head>
<body>
  <div id="div1">BACK</div><div id="div2">MIDDLE</div><div id="div3">FRONT</div>
</body>
</html>
```



Exemplo

Funcionamento submenu (display/position)

■ 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.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

- estilos.css : efeito hover

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

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

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

Dropdown

Link 1

Link 2

Link 3

Layouts

Tipos

- Existem diferentes tipos de *layouts*:

- *fixed layout*

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

- *fluid layout*

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

- *elastic layout*

- As dimensões são definidas em **em** existindo um redimensionamento dos conteúdos que é proporcional ao tamanho do texto.

- *hybrid layout*

- Combina áreas de dimensões fixas e escaláveis.

fixed layout

- São criados com uma largura fixa em pixéis (px)

- este é o layout tradicional uma vez que a esmagadora maioria dos acessos era efetuada a partir de *desktops*, atualmente **esta situação já não se verifica**

- ex: 1024 x 768 (width: 960px)

- deve ser definido como vai ser alinhado relativamente à janela do browser:

- à esquerda, deixando uma margem **variável** do lado direito (resulta num layout muito desequilibrado);
- centrado, garantido margens (esquerda/direita) de largura igual, o que melhora o enquadramento do conteúdo

Vantagens	Desvantagens
Fácil de implementar	O conteúdo do lado direito fica invisível se a janela do browser tiver uma dimensão inferior à largura definida
Tem um comportamento previsível para as resoluções que foi criado.	Pode existir uma quantidade enorme de espaço vazio em monitores com uma resolução elevada
	Com valores de <i>font-size</i> elevados o <i>line length</i> (número de palavras ou caracteres por linha) pode ser muito reduzido.

fixed Layout (baseado na propriedade float)

```
#wrapper{width:960px;
margin:0px auto;
font-size:25px;}

#header,#footer{width:960px;
height:75px;
padding:10px;
background-color: lightgray;}

#nav{float:left;
width:200px;
height:500px;
padding:10px;
margin:10px;
background-color:orange;}

#main{float:left;
width:670px;
height:500px;
padding:10px;
margin:10px;
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>
```



fluid (liquid) layout

- As colunas contidas na página ajustam-se ao espaço disponível na janela do browser
 - layout versátil que se ajusta a diferentes dimensões da janela do browser, e como tal é um elemento chave do *responsive web design* (adaptar conteúdos a diferentes *viewports*)
 - As dimensões são definidas em %

Vantagens	Desvantagens
Ajustam o conteúdo ao espaço de visualização disponível;	Em monitores de grandes dimensões o <i>line length</i> pode tornar-se excessivo e prejudicar a leitura;
Permite a correta visualização em maior variedade de dispositivos (<i>desktops, mobile, ...</i>);	É menos previsível que os layouts fixos, nomeadamente para dimensões extremas dos browsers;
Evita margens muito grandes (espaço vazio);	Pode ser mais difícil de implementar que o layout anterior.
Evita a necessidade de recorrer a <i>scrollbars</i> horizontais.	

fluid Layout (baseado na propriedade float)

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

