

Programação II




Cascading Style Sheets

Sumário

- Definição
- Formas de Inserção
- Sintaxe
- Seletores
- Pseudo-classes
- Propriedades de:
 - Texto
 - Parágrafo
 - Listas
 - Fundos
 - Box model
 - Tabelas
 - Formulários
 - Visibilidade
- Unidades de Medida
- Cores
- Web fonts e importação de fontes
- Herança e Cascadeamento
- Estilização nativa e CSS Reset
- Fluxo da página
- Float, Clear e Overflow
- Posicionamento
- Media Queries
- Outros efeitos

Folhas de Estilo - CSS

- A especificação de folhas de estilo em cascata CSS (*Cascading Style Sheets*) foi introduzida para oferecer uma maneira melhor de estilizar elementos HTML, garantindo a separação entre o conteúdo e a formatação visual.
 - **HTML** → estrutura e conteúdo
 - **CSS** → formatação
- A HTML5 tornou obsoleta a formatação até então feita dentro do próprio HTML:
 - Tags: ``, `<center>`, `<big>`, `<basefont>`, `<strike>`, `<tt>`
 - Atributos: `color`, `bgcolor`, `align`, `valign`, `border` **deprecated**
- Uma folha de estilos consiste em uma ou mais regras de estilo aplicáveis aos elementos HTML, que podem ser linkadas ou embutidas nas páginas.
 - Estilo: características visuais dos elementos, como tamanho, estilo e cor de fontes, alinhamento, bordas, posicionamento, etc.
- Além de garantir a padronização das páginas, é um recurso que poupa tempo e trabalho, tanto na hora de confeccionar, quanto na de atualizar o site.
 - Cada regra pode ser aplicada a diversos elementos em uma página ou conjunto de páginas. Desta forma, a aparência permanece homogênea e as eventuais modificações são feitas apenas na regra.

Formas de Inserção

- O recurso CSS pode ser adicionado a uma página HTML das seguintes maneiras:
 1. **estilos *inline***, especificados individualmente no atributo **style** de elementos HTML específicos;
 2. **estilos incorporados** à página, com regras de estilo definidas no elemento **<style>** dentro da seção **<head>** do documento;
 3. **arquivo .css externo**, vinculado a uma ou mais páginas.
- A maneira mais recomendada é a terceira, pois desta forma um mesmo arquivo CSS pode ser compartilhado entre vários documentos HTML, além de se manter totalmente separados o conteúdo e a formatação.

1 – Estilos *inline*

- Declara o formato de um elemento específico, de forma individual.
 - Utiliza o atributo **style**.
- Este método perde muitas das vantagens de folhas de estilo, pois mistura o conteúdo com a apresentação (evite!).

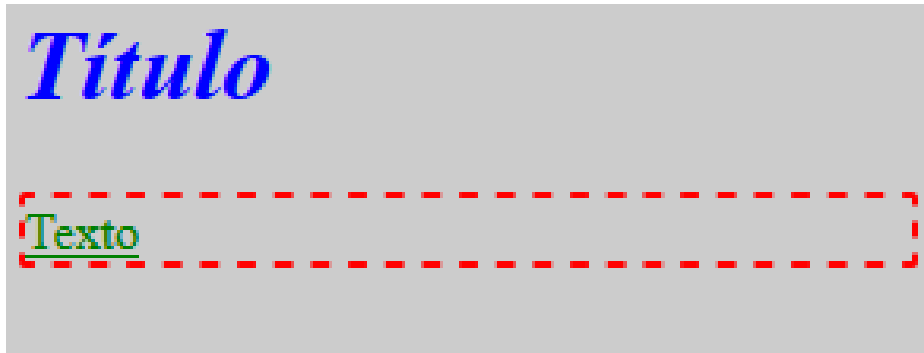
```
<h1 style="color: blue; font-style: italic">  
    Título em azul e itálico  
</h1>  
<p style="color: #00CC00; margin: 5px;">  
    Aqui um parágrafo de cor verde e com 5px nas 4 margens.  
</p>
```

Na definição de estilos, usa-se dois-pontos (:) ao invés de igual (=) para separar o nome da propriedade de seu valor.

Várias propriedades podem ser aplicadas num mesmo atributo style, desde que separadas por ponto-e-vírgula (;)

2 – Estilos internos (incorporados)

- Uma folha de estilo é dita incorporada ou interna quando as regras CSS estão declaradas no cabeçalho do documento HTML.
 - Utiliza a tag `<style>` no cabeçalho.
- Ideal para ser aplicada a uma única página.
- Cada regra pode se aplicar a um ou mais elementos dentro da mesma página.



```
...
<head>
...
<style>
.estilo {
  color:green;
  text-decoration: underline;
  border:2px dashed red;
}
body {
  background: #CCCCCC;
}
h1 {
  color: blue;
  font-style: italic;
}
</style>
</head>
<body>

  <h1>Título</h1>
  <p class="estilo">Texto</p>

...

```

3 – Estilos Externos

- Uma folha de estilo é externa quando as regras CSS estão declaradas em um arquivo de texto plano separado do documento HTML, o qual possui a extensão **.css**.
 - Utiliza a tag **<link>** no cabeçalho do documento.

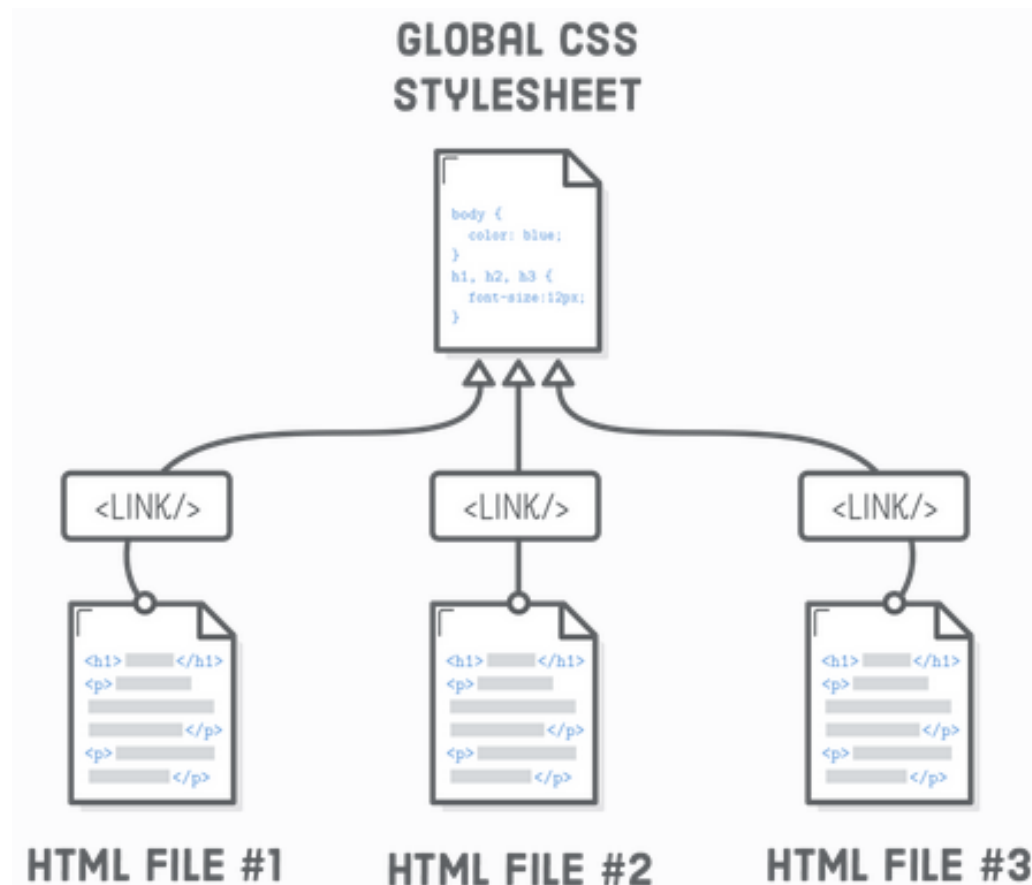
```
<!-- página HTML -->
<head>
...
<link rel="stylesheet" href="estilo.css" type="text/css">
...
</head>
```

 não é mais necessário

```
/* arquivo estilos.css */
p {
  font-size: 20pt;
  color: #ff0000;
}
h1 {
  font-family: arial, sans-serif;
}
```

3 – Estilos Externos

- Uma folha de estilo externa é ideal para ser aplicada a várias páginas. Pode-se mudar a aparência de um site inteiro, mudando apenas o arquivo CSS.



Sintaxe

- Para estilos *inline*, basta declarar os pares **propriedade: valor;** dentro do atributo **style** do elemento desejado.
- Para estilos *incorporados* ou *externos*, cada regra de estilo consiste em um seletor, seguido por um conjunto de propriedades e valores aplicáveis aos elementos referenciados por esse seletor.

seletor {
propriedade: valor;
...
}

**Declaração
(seguida de ;)**

Regra CSS

Os seletores mais utilizados são:

- nome de um elemento (tag)
- .classe
- elemento.classe
- #id
- elemento[atributo="valor"]
- [atributo="valor"]

Lista com todos os seletores:

https://www.w3schools.com/cssref/css_selectors.asp

Validador de sintaxe CSS:

<https://jigsaw.w3.org/css-validator/>

Sintaxe - Exemplos

```
h1 {  
    font-size: x-large;  
    color: green;  
}
```

A cor verde e o tamanho extra-grande se aplicarão a todos os elementos **h1** presentes no documento HTML

```
h2 {  
    font-size: large;  
    color: orange;  
}
```

Os conteúdos de todos os elementos **h2** terão fonte grande e cor laranja

```
.note {  
    font-size: small;  
    color: blue;  
}
```

Elementos que contiverem o atributo **class="note"** terão fontes pequenas em azul (note o caractere **.** antes do nome do estilo)

```
#footer {  
    font-family: comic sans ms;  
}
```

O elemento da página que possui o atributo **id="footer"** terá fonte Comic Sans (note o caractere **#** antes do nome do estilo)

Seletor de Elemento

- Aplica-se a todas as instâncias de um elemento no documento:
`p { color: red; text-align: center; background-color: #ccc; }`
- Conjunto de vários elementos (separados por vírgula):
`p, h1, h2, h3 { text-align: justified; }`
- Todos os elementos da página (*):
`* { margin: 0px; padding: 0px; }`
- Contexto:
`ol li a { color: red; } /* somente elementos <a> que são descendentes de um item de lista ordenada */`
- Atributo:
`input[type="text"] { border: 1px solid red; } /* todos os input tipo texto */`

Seletor de ID

- Aplica-se a uma única instância do elemento no documento:

```
#topo {  
    font-style: italic;  
    color: red;  
    text-align: center;  
}
```

- Exemplo de aplicação:

```
<h1 id="topo">
```

Seletor de Classe

- Classes genéricas (aplicam-se a quaisquer elementos vinculados à classe):

`.sinopse { font-style: italic; color: lightgray; }`

`.destaque { color: red; }`

- Exemplo de aplicação:

`<p class="sinopse">`

``

Neste caso, o elemento terá as propriedades das 2 classes. Se há uma regra conflitante (como ocorre com color), aquela declarada por último na folha de estilos prevalecerá.

- Classes associadas a tags específicas:

`p.abstract { font-style: italic; left-margin: 0.5cm; }`

`p.equation { font-family: Symbol; text-align: center; }`

Pseudo-classes

- São classes predefinidas ligadas ao estado do elemento.

- Sintaxe:

`seletor:pseudo-classe{ propriedade:valor; }`

- Elemento `<a>`:

`a:link { color: #CC6699 } /* links não visitados */`

`a:visited { color: #669966 } /* links visitados */`

`a:active { color: #0000FF } /* quando o link está selecionado */`

`a:hover { color: #FFCC66 } /* quando o mouse está sobre o link */`

- Elemento `<p>`:

`div p:first-child { font-size: 12px; } /* Aplica-se ao primeiro <p> quando este for descendente de uma div */`

`p:first-line { font-style:small-caps } /* primeira linha do parágrafo */`

`p:first-letter { font-size: 200%; float: left } /* primeira letra */`

- Elemento `<input>`:

`input:focus { border: 1px solid red; } /* borda vermelha quando está com foco */`

Propriedades de Texto

- Cor:
 - color: /* nome | valor hex. | valor RGB | valor HSL */
- Fonte:
 - font-style: /* normal | italic | oblique */
 - font-weight: /* normal | bold | bolder | lighter | valor | initial | inherit */
 - font-family: /* nome completo | família (sans-serif, serif, monospace) */
 - font-size: /* % | px | pt | em | ex | small | large | etc */
 - text-transform: /* uppercase | lowercase | capitalize */
 - text-decoration: /* none | overline | underline | line-through */
 - text-shadow: /* tam. vertical, tam. horizontal, cor */

Sans-serif

Serif

Propriedades de Parágrafo

- Alinhamento:
 - `text-align: /* right | center | left | justified */`
- Espaçamento:
 - `line-height: /* altura de cada linha */`
 - `letter-spacing: /* espaçamento entre letras */`
 - `word-spacing: /* espaçamento entre palavras */`
 - `text-indent: /* indentação na primeira linha */`
- Direção:
 - `direction: /* ltr | rtl */`

Propriedades de Listas

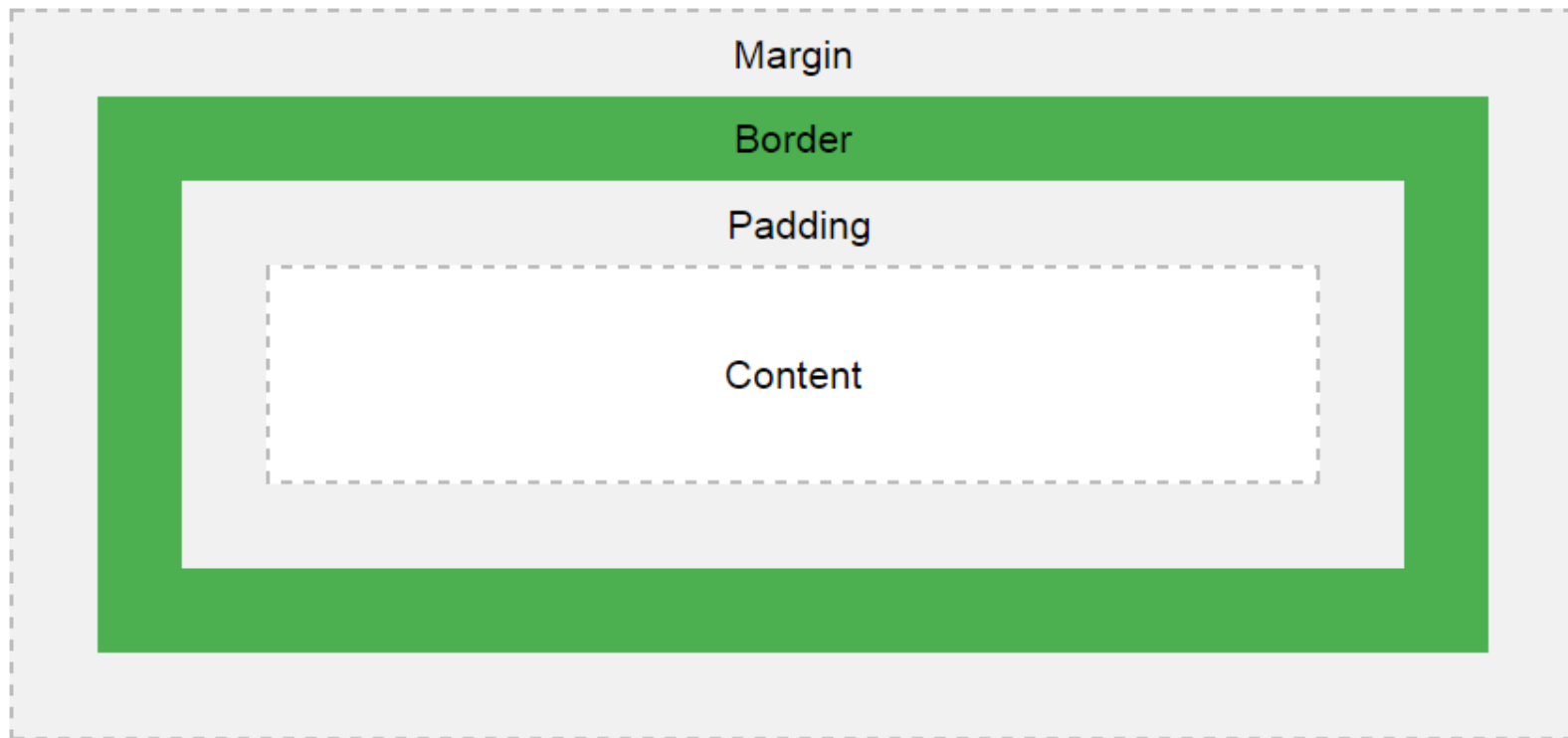
- Tipo de marcador:
 - `list-style-type: /* none | square | circle | upper-roman | lower-roman | upper-alpha | lower alpha */`
- Imagem como marcador:
 - `list-style-image: /* url('caminho do arquivo'); */`
- Posição:
 - `list-style-position: /* inside | outside */`

```
ul {  
    background: #3399ff;  
    padding: 10px;  
}  
  
ul li {  
    background: #cce5ff;  
    list-style-type: none;  
    margin: 5px;  
}
```



Box Model

- Todos os elementos HTML são renderizados como “boxes” (caixas).
 - “block boxes” ou “inline boxes”
- O Box Model é um conjunto de regras que determina as dimensões de cada elemento, por meio das seguintes propriedades: margens (*margin*), bordas (*border*), preenchimento (*padding*) e conteúdo (*content*).



Propriedades do “Box Model”

- Margens

- *shorthand*:

- `margin: /* auto | valor absoluto | % | inherit */`

Margin: auto centraliza o elemento na tela
(width deve estar definido explicitamente)

- Margens individuais:

- `margin-left`
 - `margin-right`
 - `margin-top`
 - `margin-bottom`

```
div {  
    width: 320px;  
    padding: 10px;  
    border: 5px solid gray;  
    margin: 0;  
}
```

- Padding

- *shorthand*:

- `padding: /* valor absoluto | % | inherit */`

- Paddings individuais:

- `padding-left`
 - `padding-right`
 - `padding-top`
 - `padding-bottom`

Propriedades do “Box Model”

- Bordas:
 - border
 - border-left
 - border-right
 - border-top
 - border-bottom
 - border-color: /* nome | cód. hex. | cód. rgb | transparent */
 - border-width: /* thin | medium | thick | tamanho (px, pt, cm,...) */
 - border-style: /* solid | dotted | dashed | etc. */
- A propriedade **border** pode condensar os valores das demais (*shorthand* ou atalho):
 - border: 5px solid red;
- A propriedade **border-radius** especifica cantos arredondados

```
p.round3 {  
    border: 2px solid red;  
    border-radius: 12px;  
}
```

Propriedades de Fundos

- Fundos:
 - background-color: /* nome | cód. hex. | cód. rgb | cód. hsl */
 - background-image: /* url('caminho do arquivo'); */
 - background-repeat: /* no-repeat | repeat | repeat-y | repeat-x */
 - background-attachment: /* fixed | scroll | local */
 - background-position: /* right top | left top, etc */
 - background-clip: /* border-box | padding-box | content-box */
 - background-size: /* dimensões da img no box */
- A propriedade background pode condensar os valores das demais (*shorthand* ou atalho):
 - background: #ffffff url("img_tree.png") no-repeat right top;
- O background se aplica ao conteúdo e ao *padding*, mas não às margens (estas são sempre transparentes).

Propriedades do “Box Model”

- Elementos de bloco (*block boxes*):
 - Sempre aparecem abaixo do elemento de bloco anterior;
 - A largura padrão é configurada automaticamente, baseada na largura de seu *container* (elemento pai). Se não há elemento *container*, os elementos de bloco terão sempre a largura da janela do navegador;
 - A altura padrão dos elementos de bloco é baseada no seu conteúdo.
- Elementos *inline* (*inline boxes*):
 - Não se destinam a determinar layouts, mas a estilizar conteúdos dentro de um bloco;
 - A largura padrão é baseada em seu conteúdo (e não na largura do elemento pai);
 - Não afetam o espaçamento vertical;
 - Ignoram completamente as margens top e bottom; o padding top e bottom, quando presente, extrapola a altura do elemento, sem no entanto afetar o layout vertical da página.



“BLOCK BOX”



“INLINE BOXES”

Definindo explicitamente a Altura / Largura

- Altura:
 - height: /* auto | valor absoluto | valor relativo */
- Largura:
 - width: /* auto | valor absoluto | valor relativo */
 - max-width: /* none | valor absoluto | valor relativo */
- **max-width** define a largura máxima de um elemento, isto é, mesmo em uma janela de tamanho maior, o elemento não se expandirá além da sua largura original. Permite ainda que o elemento seja reduzido caso o espaço disponível na janela seja menor do que a largura original (evita o aparecimento da barra de rolagem).
- Tanto a largura quanto a altura se referem à área de conteúdo do box, isto é, sem contar *padding*, *border* e *margin*.
 - Isso pode se tornar um problema, por exemplo, quando as somas das larguras dos elementos extrapola 100%.
 - Este comportamento pode ser mudado com a propriedade **box-sizing: border-box**. Assim, o espaço para *padding*, *border* e *margin* será descontado da área do conteúdo.

Propriedades de Tabela

- Bordas:
 - border: /* já visto; aplicar a table, th, tr e td */
 - border-collapse: /* collapse */
 - border-spacing: /* valor absoluto horiz. e valor absoluto vert. */
- Alinhamento:
 - text-align /* já visto; alinhamento horizontal */
 - vertical-align: /* top | bottom | middle */
- Linhas com hover:
 - tr:hover { background-color: #f5f5f5; }
- Tabelas zebradas (odd: ímpar; even: par):
 - tr:nth-child(odd){ background-color: #f2f2f2 }

```
table {  
    border-collapse: collapse;  
}  
table, td, th {  
    border: 1px solid black;  
}
```


Propriedades de Formulários

- Diversas opções já vistas podem ser utilizadas para estilização de formulários:
 - **border**: define as bordas dos campos (todas ou individualmente)
 - **padding**: define espaço entre o conteúdo e a borda
 - **margin**: define margens externas nos campos
 - **background-color**: cor de fundo
 - **color**: cor do texto
 - **:hover** e **:focus**, para mudar alguma característica quando o mouse passa sobre o campo ou quando o mesmo é selecionado, respectivamente
 - **background-image**, **background-position** e **background-repeat**: para adicionar ícones dentro das caixas de texto.
- Podemos utilizar seletores de atributos para facilitar a aplicação de estilos a todos os campos do formulário:
 - `input[type=text] { ... }`
 - `input[type=submit] { ... }`

Propriedades de Visibilidade

- A principal propriedade para controle de layout é **display**, que especifica como (e se) um elemento será mostrado.
 - Todos os elementos possuem um valor default (a maioria é *inline* ou *block*). Isso pode ser redefinido alterando-se a propriedade display
- Ocultando elementos:
 - Ao ocultar um elemento com **display: none**, este não será mostrado, e a página se comportará como se ele não existisse (os elementos seguintes ocuparão o espaço)
 - Ao ocultar com **visible: hidden**, o elemento não é mostrado, mas o espaço que seria ocupado por ele permanece alocado.

- display

- block
- inline
- inline-block
- list-item
- none

display: inline-block;
comporta-se como inline,
mas aceita medidas de
altura e largura

display: none
afeta o layout

- visibility

- visible
- hidden

visibility: hidden
não afeta o layout

Unidades de Medida (distância)

- Absolutas:
 - **px** (pixels), **pt** (pontos; 1pt = 1/72in), **pc** (1pc = 12pt),
 - **cm** (centímetro), **mm** (milímetro), **q** (¼ de milímetro), **in** (polegada – 2.54cm ou 96px),
- Relativas:
 - **%** (porcentagem – relativa à largura do elemento pai)
 - Para fontes:
 - **em** (tamanho atual da fonte do elemento em questão),
 - **ex** (altura da letra “X” da fonte padrão do navegador),
 - **ch** (largura mais espaçamento do dígito 0),
 - **rem** (relativa ao tamanho da letra “M” do elemento raiz do documento)
 - Porcentagem da *viewport* (tamanho da janela do navegador):
 - **vw, vh, vmin, vmax**
 - A unidade é opcional quando o valor é 0 (zero)

Recomendadas
para uso em **telas**:

px
em
%

Unidades de Medida (outras)

- Duração
 - **s** (segundo)
 - **ms** (milisegundo)
- Ângulos
 - **deg** (graus – 360 em um círculo)
 - **grad** (grados – 400 em um círculo)
 - **rad** (radianos – 2π rad em um círculo)
 - **turn** (voltas – 1 em um círculo)
- Resolução
 - **dpi** (dots per inch – pontos por polegada)
 - **dpcm** (dots per cm – pontos por centímetro)
 - **dppx** (dots per pixel – pontos por pixel)

Cores

- O HTML e o CSS suportam um conjunto de 140 nomes de cores:
 - https://www.w3schools.com/colors/colors_names.asp
- As cores também podem ser especificadas usando valores RGB, Hexadecimais, HSL, bem como RGBA e HSLA.
- Exemplo: cor "Tomato":
 - RGB (*Red, Green, Blue*): **rgb(255, 99, 71)**
 - Hexadecimal: **#ff6347**
 - HSL (*Hue, Saturation, Lightness*): **hsl(9, 100%, 64%)**
- Exemplo: cor "Tomato", com 50% de transparência:
 - RGBA (*Red, Green, Blue, Alpha*): **rgba(255, 99, 71, 0.5)**
 - HSLA (*Hue, Saturation, Lightness, Alpha*): **hsla(9, 100%, 64%, 0.5)**

Cores em RGB / Hexadecimal

- RGB (*Red, Green, Blue*) permite especificar até 16 milhões de cores como combinação de diferentes intensidades de vermelho, verde e azul.
 - o menor valor que pode ser dado a cada uma das cores é 0 (em HEX: 00) e o maior valor é 255 (em HEX: FF).
- Os valores podem ser especificados em decimal:
 - `rgb(0, 0, 0); /* preto */`
 - `rgb(255, 255, 255); /* branco */`
 - `rgb(255, 0, 0); /* vermelho puro */`
- Ou em hexadecimal:
 - `#000000 /* preto */`
 - `#FFFFFF /* branco */`
 - `#FF0000 /* vermelho puro */`
- Variações de cinza são obtidos usando os mesmos valores para vermelho, verde e azul.

Cores em HSL

- HSL (*Hue*, *Saturation*, *Lightness*) especifica cor em termos de matiz, saturação e luz.
 - *Hue* (matiz) é um grau no círculo cromático, onde 0 é vermelho, 120 é verde e 240 é azul.
 - *Saturation* (saturação) é a intensidade da cor. É um percentual onde 100% equivale à cor pura, sem adição de cinza, e 0% equivale a um tom totalmente cinza (não se vê mais a cor original).
 - *Lightness* (luz) é um percentual onde 0% é preto (sem luz) e 100% é branco (luz total).
- Exemplos:
 - `hsl(0, 100%, 50%)` /* vermelho puro */
 - `hsl(0, 100%, 0%)` /* preto – matiz vermelho, mas com 0 de luz */
 - `hsl(0, 100%, 100%)` /* branco – matiz vermelho, mas com luz total */

Web fonts

- Tradicionalmente, as fontes declaradas na propriedade **font-family** são buscadas no computador do usuário para exibição.
 - Se a fonte não está instalada no computador, o navegador utiliza sua fonte padrão.
- O recurso *web fonts* permite a utilização de fontes que não estejam instaladas no computador do usuário. O arquivo da fonte fica armazenado no servidor e será baixado automaticamente quando necessário.
- Isto pode ser feito por meio da regra CSS3 `@font-face`. Basta atribuir um nome interno para a fonte e indicar o caminho para o arquivo.
 - Este nome interno pode então ser referenciado na propriedade `font-family` dos elementos.

```
@font-face {  
  font-family: "Minha fonte";  
  src: url(minhafonte.ttf);  
}
```

```
div {  
  font-family: "Minha fonte";  
}
```


Importando fontes

- Outra alternativa é importar fontes de um repositório externo, como o Google Web Fonts (<https://fonts.google.com/>).
- Para importar, devemos referenciar a URL da fonte em uma tag `<link>` no cabeçalho do documento. Depois disto o nome da fonte já pode ser referenciado na propriedade `font-family` dos elementos.
- O exemplo abaixo importa a fonte Hi Melody:

```
<link href="https://fonts.googleapis.com/css?family=Hi+Melody"
rel="stylesheet">
```

- Utilização:

```
font-family: "Hi Melody";
```

Herança e Cascadeamento

- Algumas propriedades definidas em um elemento são herdadas (efeito cascata) por seus elementos filhos:
 - `body { color: blue; } /* define a fonte como azul para todos os elementos da página */`
- As propriedades mais específicas sobrescrevem as herdadas:
 - `body { color: blue; } /* todo o texto da página será em azul ... */`
 - `p { color: black; } /* ... mas os parágrafos terão a cor preta ... */`
 - `p.destaque { color: red; } /* ... e parágrafos da classe destaque terão fonte vermelha */`
- Propriedades que não são herdadas automaticamente do elemento pai (como width, height, padding, margin) podem ser herdadas explicitamente, utilizando-se o valor **inherit** na referida propriedade do elemento filho:
 - `width: inherit; /* herda a largura do elemento pai */`

Herança e Cascadeamento

- Sabemos que podemos ter estilos inline, incorporados ou externos.
- Quando uma propriedade foi definida para um mesmo seletor em diferentes folhas de estilo (ou em mais de uma regra dentro da mesma folha), o valor da última regra lida será utilizado.
 - É possível forçar uma propriedade ao declará-la como **!important**:
`p.destaque { color: red !important; } /* mesmo que p.destaque seja redefinido posteriormente, a cor vermelha permanecerá */`
- Em último caso, quando uma propriedade não foi definida em nenhuma folha de estilos, será adotado o estilo padrão do navegador.
- Portanto, o navegador adota a seguinte ordem de prioridade:
 - 1) Regras declaradas como **!important**;
 - 2) Estilo inline (mais “forte”, sobrescreve os demais);
 - 3) Estilo incorporado ou externo (qual vem por último no <head>);
 - 4) Folha de estilo padrão do navegador do usuário.

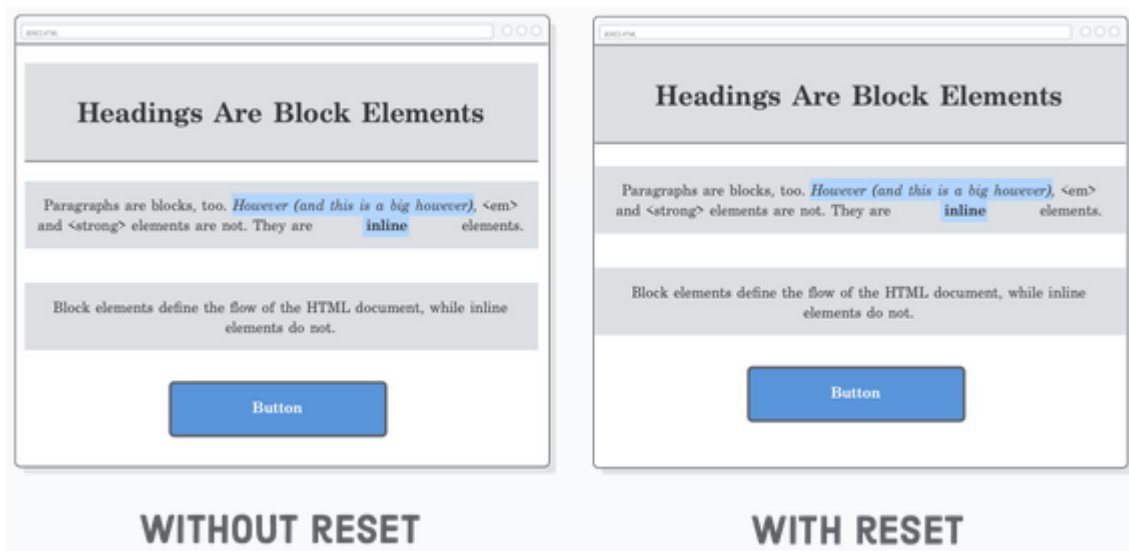
Estilização nativa e CSS Reset

- Quando não definimos a estilização de um elemento, é adotada a estilização nativa definida por uma folha de estilos interna do navegador. Porém, as regras constantes na folha de estilos nativa não são padronizadas e cada fabricante adota os valores de CSS que bem entende.
- Embora na prática não sejam valores absurdamente diferentes, são diferentes o suficiente para causar inconsistências de renderização entre navegadores.
- Em 2007, Eric Meyer publicou uma folha de estilos capaz de sobrescrever a folha de estilos nativa, a qual foi chamada de CSS Reset (<http://meyerweb.com/eric/tools/css/reset/>). A intenção é setar um valor básico para todas as características do CSS, sobrescrevendo totalmente os estilos padrão do navegador.
- Com base nesta ideia outras CSS Reset surgiram:
 - **Normalize:** <http://necolas.github.io/normalize.css/>
 - **HTML5 Doctor CSS Reset:** <http://cssreset.com/scripts/html5-doctor-css-reset-stylesheet/>
 - **YUI3 Reset CSS:** <https://yuilibrary.com/yui/docs/cssreset/>

Estilização nativa e CSS Reset

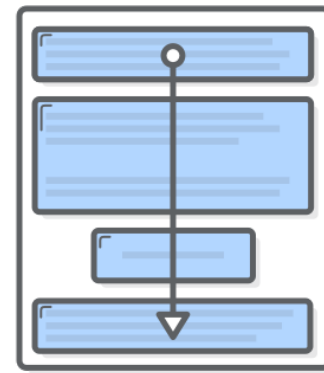
- Mesmo que opte por não utilizar uma folha de estilo específica para CSS Reset, pode ser útil reconfigurar as margens, *padding* e *box-sizing* de todos os elementos, inserindo a regra abaixo no início de sua folha de estilos (compare o antes e depois na figura):

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

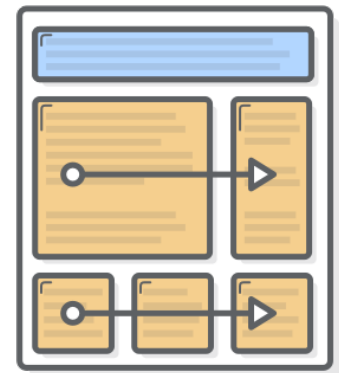


Fluxo da página

- Os elementos HTML são exibidos no navegador de acordo com a ordem de leitura do idioma da página (em geral, da esquerda para a direita, de cima para baixo).
- O navegador organiza os elementos na página iniciando pelo topo do arquivo, exibindo os elementos que encontrar:
 - Os elementos de bloco possuem **fluxo vertical**, sendo cada um adicionado verticalmente, um abaixo do outro
 - p, header, footer, section, div, form, h1, h2, ...
 - Os elementos de linha têm **fluxo horizontal**, do canto superior esquerdo até o canto inferior direito
 - img, span, a, texto em geral, ...



VERTICAL FLOW

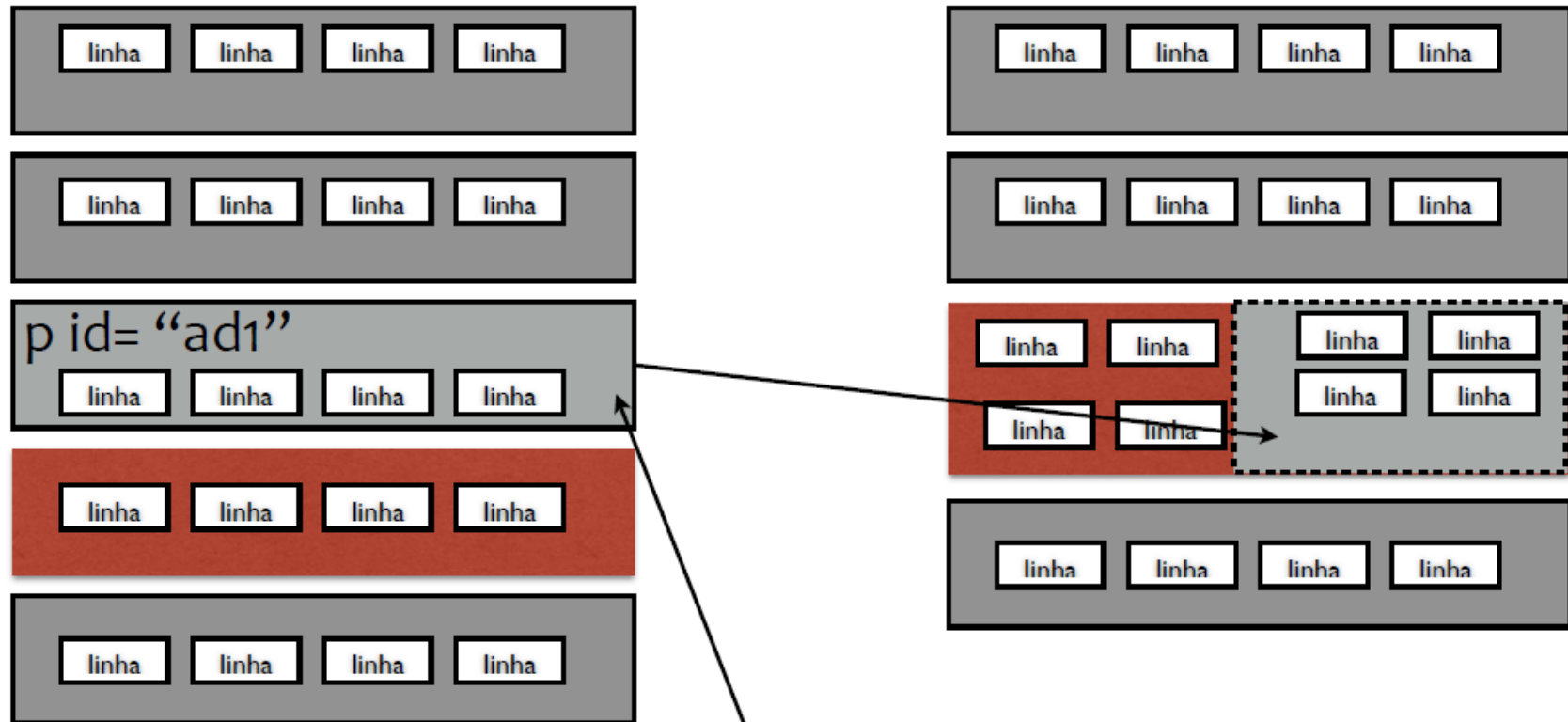


HORIZONTAL FLOW

Float

- A propriedade float nos permite controlar o posicionamento vertical de um elemento.
- Um elemento de bloco com propriedade **float** será retirado do fluxo vertical do documento e movido para a esquerda ou direita, o que faz com que o conteúdo abaixo dele flua ao seu redor (ao invés de posicionar-se abaixo dele).
 - Os elementos de bloco subsequentes seguirão o fluxo normal, ignorando o elemento flutuante; os elementos de linha serão arranjados considerando o elemento flutuante.
- A propriedade **float** tem quatro valores válidos:
 - **left**: o elemento flutua à esquerda de seu elemento pai
 - **right**: o elemento flutua à direita de seu elemento pai
 - **none**: o elemento não flutua (*default*)
 - **inherit**: herda o valor de float de seu elemento pai
- Quando há vários elementos flutuando na mesma direção, estes serão “empilhados” horizontalmente, isto é, dispostos lado a lado. Por este motivo, a propriedade **float** é bastante utilizada na composição de layouts de páginas com várias colunas.

Float



```
#ad1{
```

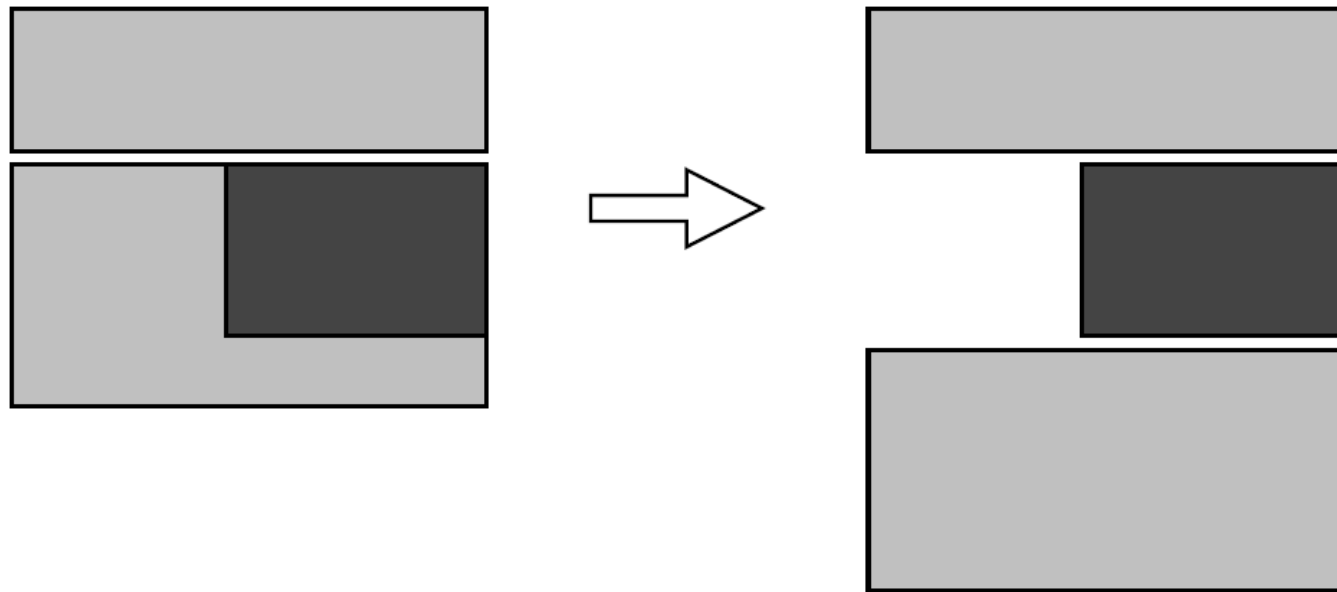
```
width: 200px;
```

```
float:right;
```

```
}
```


Clear

- Às vezes, é indesejável que um elemento flutue.
 - Por exemplo, um cabeçalho que introduz uma nova seção de um artigo não deve aparecer ao lado de uma imagem da seção anterior.
- A propriedade **clear** diz a um elemento de bloco para ignorar qualquer float que apareça antes dele.
 - Ao invés de flutuar ao redor de um elemento flutuante anterior, um elemento com clear sempre aparece após os elementos flutuantes.
 - É como forçar um elemento de bloco de volta ao fluxo padrão vertical da página.
- Valores possíveis: left (esquerda), right (direita), both (ambos), none (padrão) e inherit.

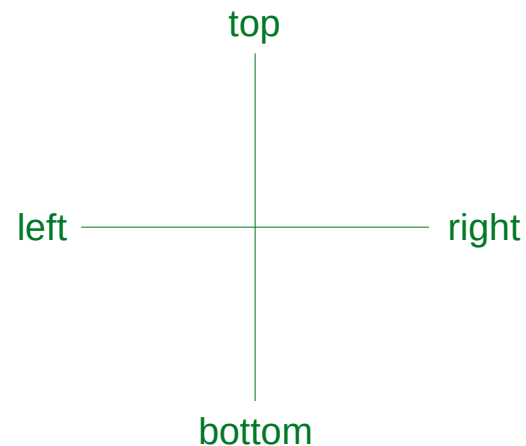


Overflow

- Quando um elemento possui tamanho insuficiente para o comportar o conteúdo, a propriedade **overflow** especifica como o navegador deve proceder:
 - **visible**: (default) o conteúdo excedente é renderizado para fora do box de seu container;
 - **hidden**: o conteúdo excedente é cortado e não será visível;
 - **scroll**: uma barra de rolagem é adicionada para permitir acesso ao conteúdo excedente que não está visível;
 - **auto**: adiciona barra de rolagem somente quando necessário.
- Quando se trata de layouts com elementos flutuantes (float), é importante destacar que estes elementos não contribuem para a altura de seus containers, quando esta é calculada automaticamente (isto é, quando a altura não é definida de modo fixo).
 - Por este motivo, muitas vezes precisamos forçar isto adicionando **overflow: hidden** ao container. Assim, o elemento container reconhecerá a altura de todos os elementos float ali contidos.

Posicionamento

- Para alterar o posicionamento padrão dos elementos, isto é, a posição original de acordo com o fluxo do documento, existem as propriedades de coordenadas **top**, **bottom**, **left** e **right**.
 - Porém, estas funcionam apenas se definirmos previamente a propriedade **position** do elemento
 - O resultado da aplicação das coordenadas também difere dependendo do valor de **position**
- A propriedade position determina qual é o modo de posicionamento de um elemento. Pode receber os seguintes valores:
 - static
 - relative
 - absolute
 - fixed
 - sticky



Posicionamento

- position: static
 - Posicionamento padrão para todos os elementos.
 - Um elemento com posição **static** permanece sempre em seu local original no documento, isto é, sua posição é dada automaticamente pelo fluxo estático da página (por padrão ele é renderizado logo após seus irmãos).
 - Ignora as coordenadas (left, right, top e bottom), quando definidas.
 - O tamanho do seu elemento pai leva em conta o tamanho do elemento static.

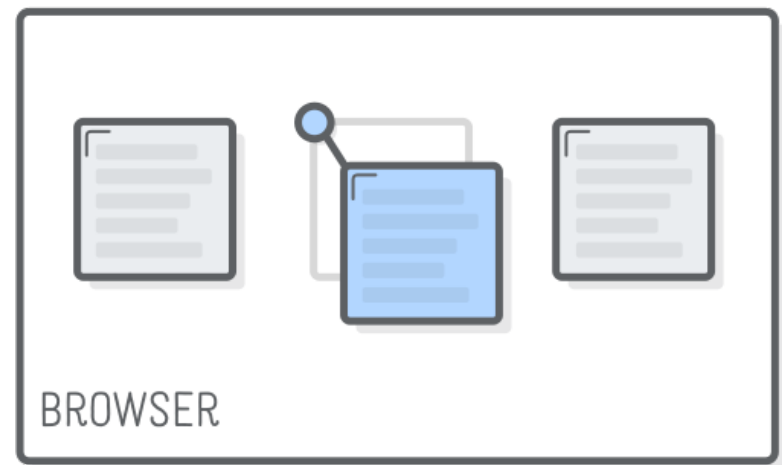
```
#topo
{
  position: static;
  color:red;
  text-align:center;
  background-color: #ccc;
}
```

```
p
{
  position:static;
  background-color: #ccc;
}
```

Posicionamento

- position: relative
 - Por padrão comporta-se como static, porém pode ser movido em relação à sua posição normal através da definição das coordenadas (top, bottom, left e right).
 - Este deslocamento não afeta os demais elementos da página. O espaço vago (que seria ocupado pelo elemento em sua posição original no fluxo) não é ocupado por outros elementos.
 - O tamanho do seu elemento pai leva em conta o tamanho do elemento relative, porém sem levar em conta seu deslocamento.

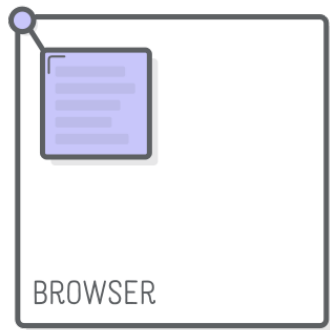
```
.logotipo {  
    position: relative;  
    top: 50px;  
    left: 25px;  
}
```



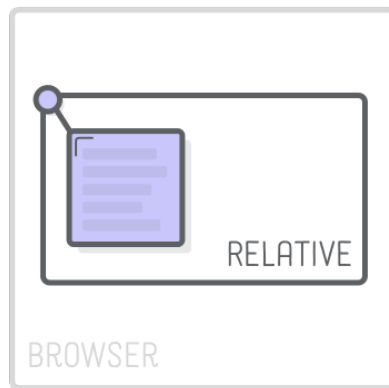
RELATIVE POSITIONING

Posicionamento

- position: absolute
 - Assemelha-se ao relative, porém, o deslocamento não se dá em relação à posição original do próprio elemento, mas em relação ao seu elemento ancestral mais próximo cujo posicionamento seja diferente de static, ou, em relação ao body da página, se não houver elemento ancestral posicionado.
 - O elemento é removido do fluxo do documento e o espaço vago é utilizado pelos elementos seguintes no fluxo.
 - Obedece às coordenadas (left, right, top, bottom) de acordo com o tamanho e posição desse elemento pai (ou body), a partir de seu canto superior esquerdo.
 - Seu tamanho não conta para calcular o tamanho do elemento pai.



ABSOLUTE POSITIONING



RELATIVELY ABSOLUTE POSITIONING

```
#menu{  
  position:absolute;  
  left: 30px;  
  top: 30px;  
  background: #bbb;  
}
```

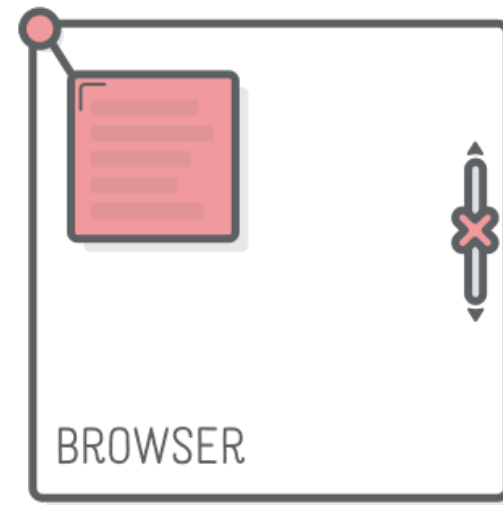
```
#principal{  
  position: relative;  
  width: 1240px;  
  height:430px;  
  background: #000;  
}
```

Se #menu estiver dentro do elemento #principal, o deslocamento se dará relativamente a este último. Senão, se dará em relação ao body da página.

Posicionamento

- position: fixed
 - Toma como referência a porção visível do documento no navegador (*viewport*), e mantém essa posição inclusive quando há rolagem na tela.
 - Assim como absolute, o elemento é removido do fluxo do documento e o espaço vago é utilizado pelos elementos seguintes no fluxo.
 - Pelo menos uma configuração de posicionamento vertical (left ou right) e uma horizontal (top ou bottom) é obrigatória.

```
#fixo{  
    position:fixed;  
    top: 0px;  
    right: 40px;  
}
```



FIXED POSITIONING

Posicionamento

- position: sticky
 - A posição do elemento é baseada na posição atual de rolagem da tela, alternando entre “relative” e “fixed”
 - É relative até atingir a posição de rolagem na viewport, quando então se torna fixa, voltando a ser relative quando seu container não for mais visível na viewport.
 - É necessário indicar pelo menos uma coordenada para que este posicionamento funcione corretamente.
 - Ainda não é plenamente suportada em todos os navegadores, de modo que é conveniente definir também a propriedade com o prefixo webkit.

```
div.sticky {  
    position: -webkit-sticky;  
    position: sticky;  
    top: 0;  
    padding: 5px;  
    background-color: #cae8ca;  
}
```


Outros recursos CSS3

- Box-shadow e Text-shadow
 - definição de efeitos sobreados em elementos ou textos
- Gradientes
 - efeitos de transição suaves entre duas cores.
- Transforms
 - efeitos de transformação 2D e 3D (girar, inclinar, rotacionar)
- Transitions
 - permite mudar os valores das propriedades de um elemento de forma gradual e com um tempo de duração.
- Animações
 - possibilita que um elemento mude gradativamente de um estilo para outro.