CSS avançado

Imagem em botão

- É possível com CSS trocar o escrito de um botão de formulário por uma imagem:

```
.busca{
background-image: url('lupa.png');
background-repeat: no-repeat;
border: none;
width: 20px;
height: 20px;
}
```

No html:

```
<input class="busca" type="button">
```

Bordas arredondadas

- Uma das novidades do CSS 3 foi a adição de bordas arredondadas via CSS.
- Até então, a única forma de obter esse efeito era usando imagens, o que deixava a página mais carregada e dificultava a manutenção.

Bordas arredondadas

- propriedade border-radius

```
div {
border-radius: 5px;
}
```

Podemos também passar valores diferentes para cantos diferentes do elemento:

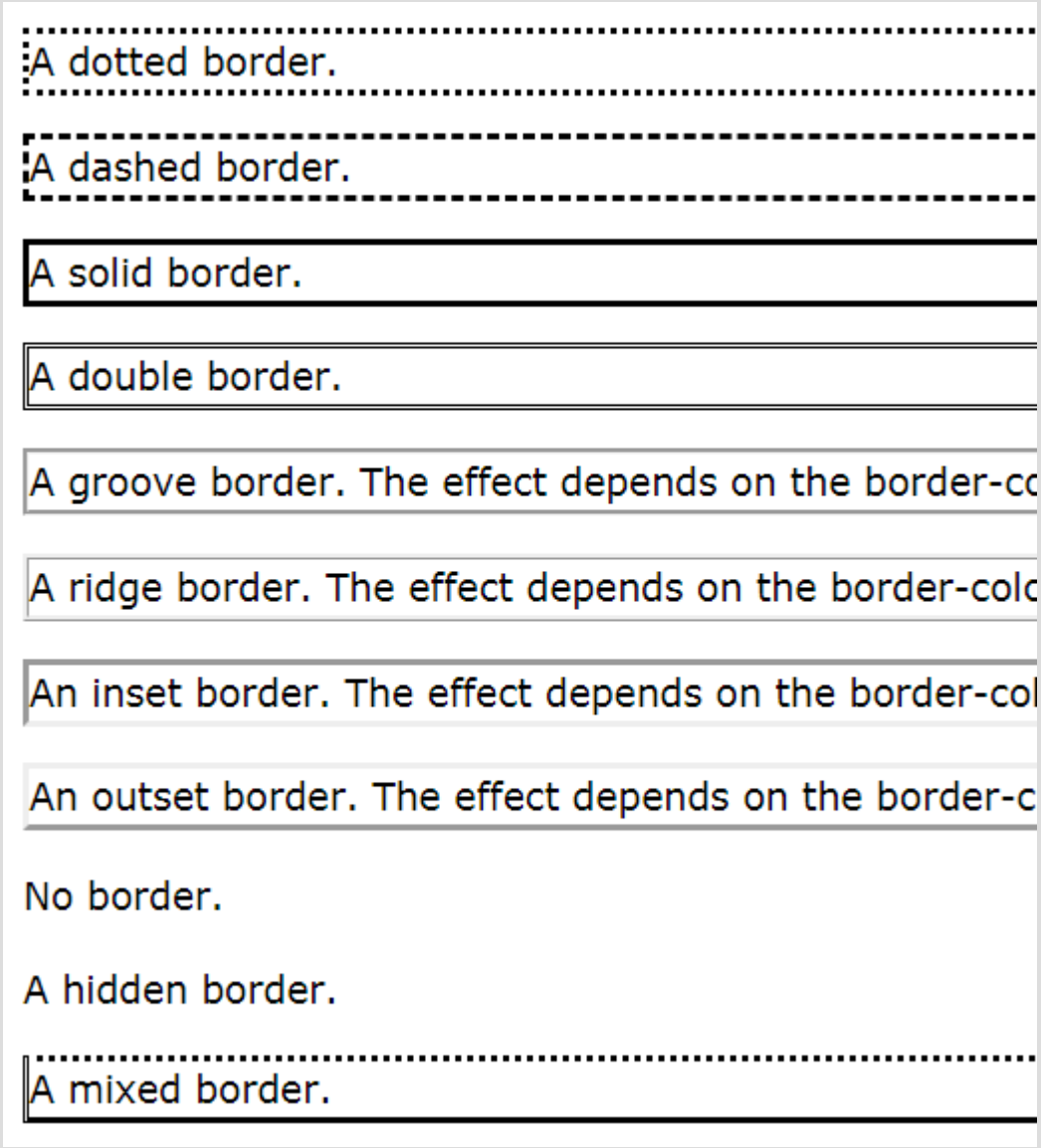
```
/* todas as bordas arredondadas com um raio de 15px */
.a {
  border-radius: 15px;
}

/* borda superior esquerda e inferior direita com 5px
borda superior direita e inferior esquerda com 20px */
.b {
  border-radius: 5px 20px;
}

/* borda superior esquerda com 5px
borda superior direita e inferior esquerda com 20px
borda inferior direita com 50px */
.c {
  border-radius: 5px 20px 50px;
}

/* borda superior esquerda com 5px
borda superior direita com 20px
borda inferior direita com 50px
borda inferior esquerda com 100px */
.d {
  border-radius: 5px 20px 50px 100px;
}
```

Bordas arredondadas

- Os estilos de borda disponíveis são:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Bordas arredondadas



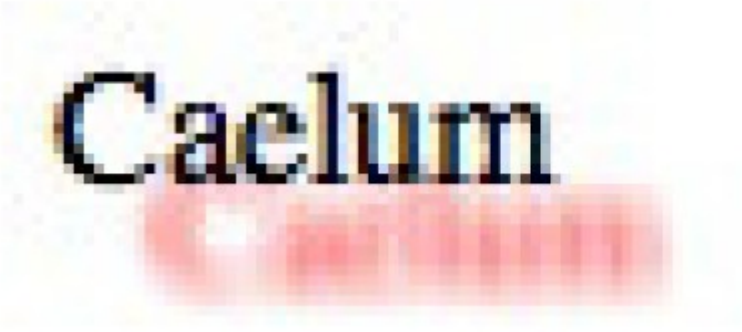
Sombras em textos

- Outro efeito do CSS 3 é o suporte a sombras em textos com text-shadow.
- Sua sintaxe recebe o deslocamento da sombra e sua cor:

```css
p {
  text-shadow: 10px 10px red;
}
```

Ou ainda pode receber um grau de espalhamento (blur):

```css
p {
  text-shadow: 10px 10px 5px red;
}
```



É possível até passar mais de uma sombra ao mesmo tempo para o mesmo elemento:

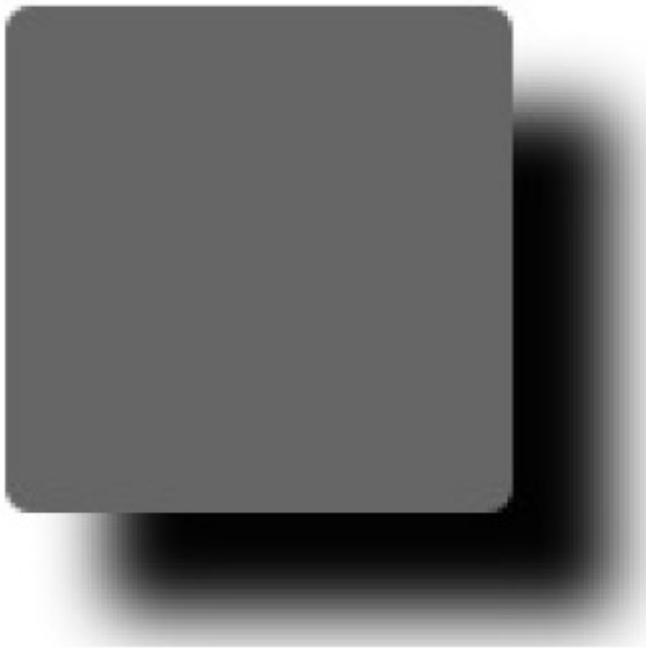
```
p {
  text-shadow: 10px 10px 5px red, -5px -5px 4px red;
}
```

Além de sombras em texto, podemos colocar sombras em qualquer elemento com `box-shadow`. A sintaxe é parecida com a do `text-shadow`:

```
box-shadow: 20px 20px black;
```

Podemos ainda passar um terceiro valor com o blur:

```
box-shadow: 20px 20px 20px black;
```



Diferentemente do `text-shadow`, o `box-shadow` suporta ainda mais um valor que faz a sombra aumentar ou diminuir:

```
box-shadow: 20px 20px 20px 30px black;
```

Por fim, é possível usar a keyword **inset** para uma borda interna ao elemento:

```
box-shadow: inset 0 0 40px black;
```

Opacidade a um elemento

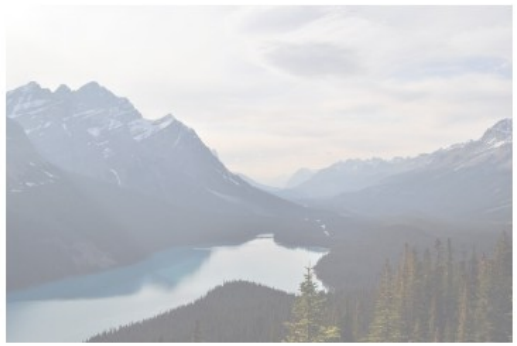
Veja esse exemplo com um parágrafo por cima de uma imagem:

É simples com a `opacity` que recebe um valor decimal entre 0 e 1:

```
p {
  opacity: 0.3;
}
```

Opacidade

- Opacidade é bastante utilizado para que a imagem fique sempre transparente a menos que usuário passe o mouse em cima dela:


Example

```
img {
  opacity: 0.5;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}

img:hover {
  opacity: 1.0;
  filter: alpha(opacity=100); /* For IE8 and earlier */
}
```

Gradientes

- O CSS3 traz também suporte à declaração de gradientes sem que precisemos trabalhar com imagens.
- Além de ser mais simples, a página fica mais leve e a renderização fica melhor por se adaptar a todo tipo
- de resolução.

Existe suporte a gradientes lineares e radiais, inclusive com múltiplas paradas. A sintaxe básica é:

```
.linear {
  background: linear-gradient(white, blue);
```

- Podemos ainda usar gradientes com angulações diferentes e diversas paradas de cores:

```
.radial {
  background: radial-gradient(white, blue);
}
```

Podemos ainda usar gradientes com angulações diferentes e diversas paradas de cores:

```
.gradiente {
  background: linear-gradient(45deg, #F0F9FF 0%, #CBEBFF 47%, #A1DBFF 100%);
}
```

A versão atual da especificação suporta um primeiro argumento que indica a direção do gradiente:

```
.linear {
  background: linear-gradient(to bottom, white, blue);
}
```

- Obs: pode haver diferenças dependendo da versão do browser

Transition no css

- Isso funciona, mas o elemento é deslocado de uma vez quando passamos o mouse.

Por exemplo: temos um elemento na posição `top:10px` e, quando passarmos o mouse em cima (hover), queremos que o elemento vá para `top:30px` . O CSS básico é:

```
#teste {
  position: relative;
  top: 10px;
}

#teste:hover {
  top: 30px;
}
```

```
#teste:hover {
  transition: top 2s;
}
```

Por padrão, a animação é linear, mas temos outros tipos para animações mais suaves:

- `linear` - velocidade constante na animação;
- `ease` - redução gradual na velocidade da animação;
- `ease-in` - aumento gradual na velocidade da animação;
- `ease-in-out` - aumento gradual, depois redução gradual na velocidade da animação;
- `cubic-bezier(x1,y1,x2,y2)` - curva de velocidade para animação customizada (avançado)

```
#teste:hover {
  transition: top 2s ease;
}
```

CSS Transforms

- Com essa nova especificação, agora é possível alterar propriedades visuais dos elementos antes impossíveis. Por exemplo, agora podemos alterar o ângulo de um elemento, mostrá-lo em uma escala maior ou menor que seu tamanho padrão ou alterar a posição do elemento sem sofrer interferência de sua estrutura.

Translate

```css
.header {
  /* Move o elemento no eixo horizontal */
  transform: translateX(50px);
}

#main {
  /* Move o elemento no eixo vertical */
  transform: translateY(-20px);
}

footer {
  /* Move o elemento nos dois eixos (X, Y) */
  transform: translate(40px, -20px);
}
```

Rotate

```
#menu-departamentos {
  transform: rotate(-10deg);
}
```

Scale

```
#novidades li {



  /* Alterar a escala total do elemento */
  transform: scale(1.2);
}

#mais-vendidos li {
  /* Alterar a escala vertical e horizontal do elemento */
  transform: scale(1, 0.6);
}
```

Skew

```
footer {
  /* Distorcer o elemento no eixo horizontal */
  transform: skewX(10deg);
}

#social {
  /* Distorcer o elemento no eixo vertical */
  transform: skewY(10deg);
}
```

É possível combinar várias transformações no mesmo elemento, basta declarar uma depois da outra:

```
html {
  transform: rotate(-30deg) scale(0.4);
}
```

MediaTypes

- Media Types servem para direcionar um determinado CSS para um determinado tipo de meio de acesso.
- Seja ele na tela do seu monitor, na sua impressora, no seu PDA, no seu sintetizador de voz, celular, smartphone, microondas etc e etc…
- Não importa com qual dispositivo o usuário esteja acessando seu site, ele deve ser bem apresentado.

MediaTypes

- O W3C criou uma forma para fazermos isso com a maior facilidade. Você pode criar um CSS específico para cada tipo de meio de acesso, com a mesma facilidade que você cria um CSS para ser visto em um Desktop.

- Você pode personalizar um site para ser visto em um Smartphone ou até mesmo quando o visitante imprimir uma página de texto do seu site. Utilizando o caso da impressão: não é interessante para o visitante, que o menu, banners e outros elementos do site sejam impressos no papel.

MediaTypes

que podem ser declarados ao se invocar
um arquivo CSS:

```
<link rel="stylesheet" href="site.css" media="screen" />
<link rel="stylesheet" href="print.css" media="print" />
<link rel="stylesheet" href="handheld.css" media="handheld" />
```

MediaTypes

Outra forma de declarar os *media types* é separar as regras dentro do próprio arquivo CSS:

```
@media screen {
  body {
    background-color: blue;
    color: white;
  }
}

@media print {
  body {
    background-color: white;
    color: black;
  }
}
```

ViewPort

- Podemos manipular o viewport dos browsers para podermos entregar um website mais adequado e confortável para os usuários.
- Essa manipulação era feita diretamente via uma metatag no head do documento, algo assim:

- <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0">

ViewPort

- Necessário quando se tem uma resolução gigante em aparelhos com a tela “relativamente” pequena, com 320×480.
- Lembre-se:resolução é uma coisa, o tamanho da tela é outra.
- Um iPhone tem uma resolução gigante (para mobiles, claro), de algo em torno de 900×640, mas o tamanho da tela é de 320×480.
- É por isso que quando você abre um website sem manipulação de viewport, ele aparece miniaturizado. Por que embora você esteja vendo o site em uma tela pequena, o site aparece como se estivesse numa resolução alta.

ViewPort

- Quando manipulamos o viewport do browser, nós “diminuímos” essa resolução.

```
@viewport {
  width: device-width;
  zoom: 1;
}
```

ViewPort

- Depois de determinar o viewport, você pode definir suas media queries normalmente, também diretamente de dentro do seu código CSS.

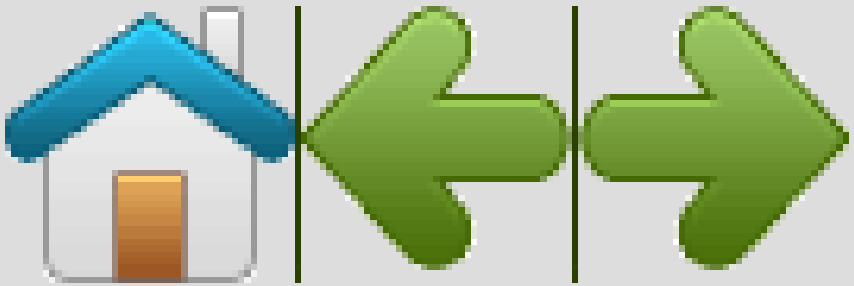
```
@viewport {
  width: device-width;
  zoom: 1;
}

@media screen and (min-width: 400px) {
  div { color: red; }
}

@media screen and (max-width: 400px) {
  div { color: green; }
}
```

CSS Image Sprites

- Sprite é uma coleção de imagens colocadas numa única imagem
- Motivação: muitas imagens demora mais pra carregar e gera várias requisições ao servidor

CSS Image Sprites

```
#home {
  width: 46px;
  height: 44px;
  background: url(img_navsprites.gif) 0 0;
}
```



Contadores css

- São como variáveis
- Utiliza as propriedades:
  - counter-reset – Cria / reseta um contador
  - counter-increment - incrementa
  - content – insere conteúdo
  - counter() or counters() function – Adiciona o valor do contador a um elemento

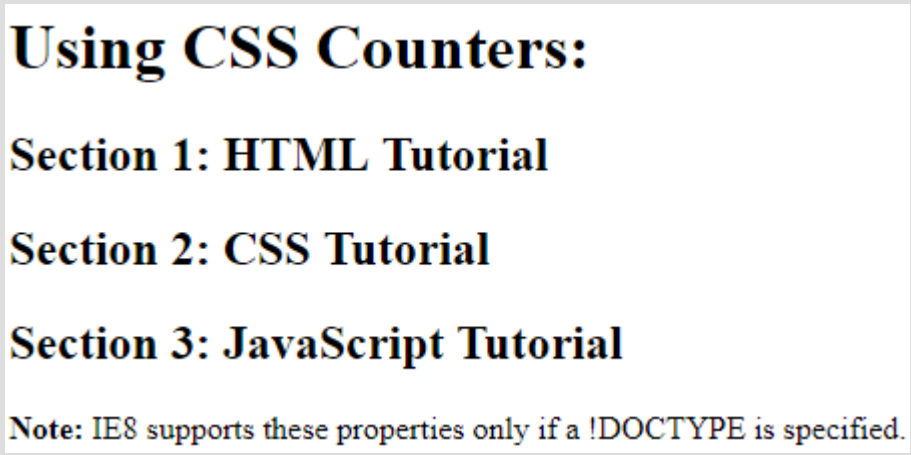
Contadores css

- Exemplo: criar um contador no body; incrementar a cada tag <h2> e escrever "Section <value of the counter>:" ao início de cada elemento:

```
body {
  counter-reset: section;
}

h2::before {
  counter-increment: section;
  content: "Section " counter(section) ": ";
}
```

Contadores css

- Resultado:



Contadores aninhados

- Este exemplo cria além do contador para <h2> um contador também para <h1> aninhados

```
body {
  counter-reset: section;
}

h1 {
  counter-reset: subsection;
}

h1::before {
  counter-increment: section;
  content: "Section " counter(section) ". ";
}

h2::before {
  counter-increment: subsection;
  content: counter(section) "." counter(subsection) " ";
}
```

Contadores aninhados

- Resultado

Section 1. HTML tutorials:

1.1 HTML Tutorial

1.2 CSS Tutorial

Section 2. Scripting tutorials:

2.1 JavaScript

2.2 VBScript

Section 3. XML tutorials:

3.1 XML

3.2 XSL

Listas numeradas

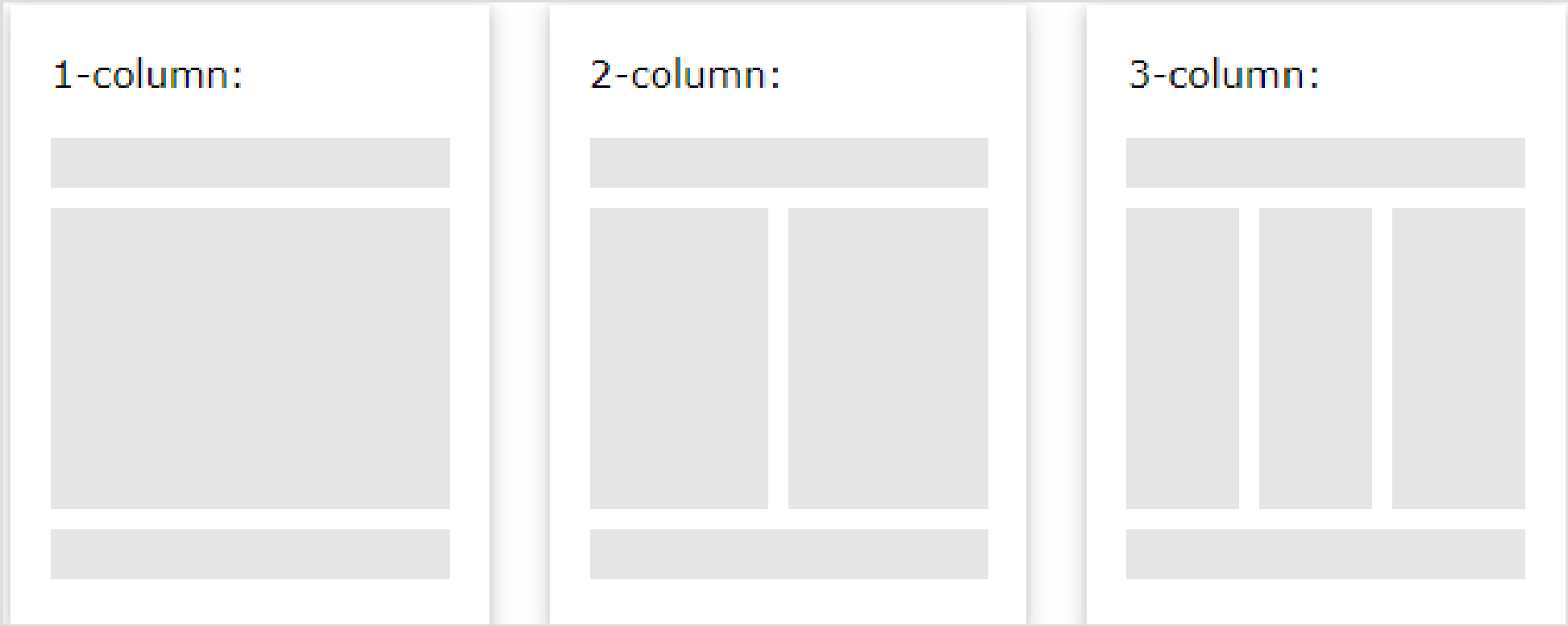
```
ol {
  counter-reset: section;
  list-style-type: none;
}

li::before {
  counter-increment: section;
  content: counters(section,".") " ";
}
```

1 item
2 item
    2.1 item
    2.2 item
    2.3 item
        2.3.1 item
        2.3.2 item
        2.3.3 item
    2.4 item
3 item
4 item

1 item
2 item

Layout com CSS

- Existem milhares de maneiras diferentes de se criar um layout:



Layout css

Um layout de
3 colunas
Que muda para
1 coluna em
telas
menores

```css
/* Create three equal columns that floats next to each other */
.column {
  float: left;
  width: 33.33%;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other on smaller screens (600px wide or less) */
@media screen and (max-width: 600px) {
  .column {
    width: 100%;
  }
}
```

Layout css

- Dica:
  - Para criar um layout de 2 colunas, mude o valor de width para 50%. Para criar um layout de 4 colunas, use 25%, etc.

Layout com colunas desiguais

- A maior parte do espaço é reservada para o conteúdo principal
- É possível mudar o width como desejar, desde que atinja 100%

```css
.column {
  float: left;
}

/* Left and right column */
.column.side {
  width: 25%;
}

/* Middle column */
.column.middle {
  width: 50%;
}

/* Responsive layout - makes the three columns stack on top of
each other instead of next to each other */
@media screen and (max-width: 600px) {
  .column.side, .column.middle {
    width: 100%;
  }
}
```

Medidas em css

- São de dois tipos:
  - Absolutas
  - Relativas

- Absolutas:
  - Não recomendada para telas, pois variam muito de tamanho
  - Pode ser usada por exemplo para impressão
- Relativas:
  - Relativas a outra medida

Medidas css

- Medidas fixas:

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)

- Pixel é relativa ao dispositivo

Medidas css

- Relativas:

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
%	Relative to the parent element