

**CSS3**

SCANIA LTDA  
59.704.907/10007-67



**IMPACTA**  
EDITORAS

COD.: TE 1641/1\_WEB





**IMPACTA**  
EDITORIA

SCANAL ALTA  
59.704.907/10007-67  
**CSS3**

# Créditos

Copyright © TechnoEdition Editora Ltda.

Todos os direitos autorais reservados. Este manual não pode ser copiado, fotocopiado, reproduzido, traduzido ou convertido em qualquer forma eletrônica, ou legível por qualquer meio, em parte ou no todo, sem a aprovação prévia, por escrito, da TechnoEdition Editora Ltda., estando o contrafator sujeito a responder por crime de Violação de Direito Autoral, conforme o art.184 do Código Penal Brasileiro, além de responder por Perdas e Danos. Todos os logotipos e marcas utilizados neste material pertencem às suas respectivas empresas.

*"As marcas registradas e os nomes comerciais citados nesta obra, mesmo que não sejam assim identificados, pertencem aos seus respectivos proprietários nos termos das leis, convenções e diretrizes nacionais e internacionais."*

## CSS3

### Coordenação Geral

Marcia M. Rosa

### Diagramação

Ivã Lucino Camargo

### Coordenação Editorial

Henrique Thomaz Bruscagin

### Equipe de Apoio

Allan Maieru

### Supervisão Editorial

Simone Rocha Araújo Pereira

### Autoria

Bruno Bove Barbosa

Glaucio Daniel Souza Santos

### Revisão Ortográfica e

### Gramatical

Luciano de Souza Munhoz

Edição nº 1 | cód.: 1641/1\_WEB

Janeiro/ 2014

# Sumário

<b>Informações sobre o treinamento .....</b>	<b>9</b>
<b>Capítulo 1 – Introdução ao CSS3 .....</b>	<b>11</b>
1.1.    História.....	12
1.2.    Versões.....	13
1.2.1.    CSS1 .....	13
1.2.2.    CSS2 .....	13
1.2.3.    CSS3 .....	14
1.3.    Como funciona a linguagem CSS .....	15
1.4.    Estilos Inline .....	16
1.5.    CSS incorporado no arquivo HTML .....	17
1.6.    Incluindo um arquivo CSS externo.....	20
1.7.    Importando um arquivo CSS.....	22
1.8.    HTML5 X CSS3 .....	23
1.9.    Estágios de padronização da W3C.....	24
1.10.    Prefixos proprietários .....	26
1.11.    Modularização .....	28
Pontos principais .....	30
<b>Teste seus conhecimentos .....</b>	<b>31</b>
<b>Capítulo 2 – Seletores e Herança.....</b>	<b>35</b>
2.1.    Seletores.....	36
2.2.    Entendendo o DOM .....	36
2.2.1.    Selecionando pela classe .....	39
2.2.2.    Selecionando pelo ID.....	40
2.2.3.    Selecionando pelo elemento HTML.....	42
2.3.    Listagem de seletores da CSS3 .....	42
2.4.    Herança em CSS3 .....	50
2.5.    Especificidade e a propriedade !important.....	55
2.6.    Boas práticas .....	58
2.6.1.    Evite qualificar demais .....	58
2.6.2.    Utilize declarações CSS compactas .....	59
2.6.3.    Evite duplicar regras .....	60
2.6.4.    Condensando regras .....	61
2.6.5.    Evite !important .....	61
Pontos principais .....	62
<b>Teste seus conhecimentos .....</b>	<b>63</b>
<b>Mãos à obra!</b> .....	<b>67</b>

# CSS3

---

<b>Capítulo 3 – Posicionamento e alinhamento.....</b>	<b>71</b>
3.1.    Introdução .....	72
3.1.1.    Box Model na CSS3 .....	72
3.1.2.    Configurando largura e altura .....	74
3.1.3.    Configurando margem - Margin .....	77
3.1.3.1.    Removendo a margem padrão das páginas .....	79
3.1.4.    Configurando espaçamento - Padding .....	79
3.2.    Configurando posicionamento - Position .....	83
3.2.1.    Posicionamento inherit.....	83
3.2.2.    Posicionamento estático.....	83
3.2.3.    Posicionamento fixo.....	86
3.2.4.    Posicionamento relativo .....	89
3.2.5.    Posicionamento absoluto .....	90
3.3.    Elementos flutuantes para layout .....	93
3.3.1.    Clear.....	96
3.3.2.    Trabalhando camadas com z-index .....	96
3.4.    Flexible Box Layout.....	101
3.4.1.    Controlando o fluxo dentro de um container.....	102
Pontos principais .....	111
<b>Teste seus conhecimentos .....</b>	<b>113</b>
<b>Mãos à obra!.....</b>	<b>117</b>

<b>Capítulo 4 – Trabalhando com cores, backgrounds e gradientes .....</b>	<b>125</b>
4.1.    Trabalhando com cores.....	126
4.1.1.    Declaração Hexadecimal .....	126
4.1.2.    Declaração RGB.....	127
4.1.3.    Declaração RGBA.....	127
4.1.4.    Declaração HSL .....	128
4.1.5.    Declaração HSLA .....	129
4.1.6.    Declaração predefinida .....	129
4.2.    Trabalhando com backgrounds .....	130
4.2.1.    Propriedade background-clip .....	131
4.2.1.1.    border-box.....	131
4.2.1.2.    content-box .....	132
4.2.1.3.    padding-box .....	132
4.2.2.    Propriedade background-origin .....	133
4.2.2.1.    border-box.....	133
4.2.2.2.    content-box .....	134
4.2.2.3.    padding-box .....	135
4.2.3.    Propriedade background-size.....	136
4.2.3.1.    Definindo tamanho com valor fixo .....	136
4.2.3.2.    Definindo tamanho com porcentagem.....	138
4.2.3.3.    Definindo tamanho com o valor cover .....	139
4.2.3.4.    Definindo tamanho com o valor contain .....	140
4.2.3.5.    Múltiplos backgrounds.....	141
4.3.    Trabalhando com gradientes .....	142
4.3.1.    Gradiente Linear .....	143
4.3.2.    Gradiente Radial .....	144
4.3.2.1.    Gradiente Radial Circular.....	144
4.3.2.2.    Gradiente Radial Elíptico .....	146
Pontos principais .....	148
<b>Teste seus conhecimentos .....</b>	<b>149</b>
<b>Mãos à obra!.....</b>	<b>153</b>

# Sumário

<b>Capítulo 5 – Bordas e sombras .....</b>	<b>155</b>
5.1.    Bordas .....	156
5.1.1.    border-width .....	156
5.1.2.    border-style .....	157
5.1.3.    border-color .....	159
5.1.4.    border-radius .....	159
5.1.5.    border-image .....	161
5.1.5.1.    border-image-source .....	162
5.1.5.2.    border-image-slice .....	162
5.1.5.3.    border-image-width .....	163
5.1.5.4.    border-image-outset .....	164
5.1.5.5.    border-image-repeat .....	164
5.2.    Box shadow .....	165
5.2.1.    Text shadow .....	167
Pontos principais .....	169
<b>Teste seus conhecimentos .....</b>	<b>171</b>
<b>Mãos à obra! .....</b>	<b>175</b>
<b>Capítulo 6 – Trabalhando com fontes .....</b>	<b>177</b>
6.1.    Fontes.....	178
6.2.    Propriedades para fontes .....	178
6.3.    Tamanho e formatações.....	179
6.4.    A regra @font-face .....	181
6.4.1.    Tipos de fontes .....	181
6.4.2.    Usando fontes locais .....	183
6.4.3.    Usando fontes remotas .....	184
6.4.4.    Usando fontes públicas .....	187
6.4.4.1.    Usando um serviço de hospedagem de fontes .....	187
6.4.5.    Restrições com fontes pagas .....	190
6.5.    Efeito em fontes.....	191
Pontos principais .....	192
<b>Teste seus conhecimentos .....</b>	<b>193</b>
<b>Mãos à obra! .....</b>	<b>197</b>

# CSS3

---

<b>Capítulo 7 – Produzindo um layout responsivo .....</b>	<b>199</b>
7.1.    Introdução .....	200
7.2.    Etapas da criação de um layout .....	200
7.2.1.    Briefing .....	201
7.2.2.    Wireframe .....	202
7.2.3.    Dimensões .....	204
7.2.4.    Viewport .....	204
7.2.4.1.    A regra @viewport .....	205
7.3.    Web design responsivo .....	206
7.3.1.    Media queries .....	208
7.3.1.1.    Projetando para desktop .....	210
7.3.1.2.    Projetando para TV e dispositivos de alta definição .....	215
7.3.1.3.    Projetando para tablet .....	215
7.3.1.4.    Projetando para smartphone .....	220
Pontos principais .....	237
<b>Teste seus conhecimentos .....</b>	<b>239</b>
<b>Mãos à obra! .....</b>	<b>243</b>
<b>Capítulo 8 – Transições em CSS3 .....</b>	<b>245</b>
8.1.    Introdução .....	246
8.2.    transition-property .....	249
8.3.    transition-duration .....	249
8.4.    transition-timing-function .....	250
8.5.    transition-delay .....	250
8.6.    transition .....	251
Pontos principais .....	254
<b>Teste seus conhecimentos .....</b>	<b>255</b>

## Informações sobre o treinamento

---

Para que os alunos possam obter um bom aproveitamento do curso **CSS3**, é imprescindível que eles tenham participado do nosso curso de **HTML5 Fundamentos**, ou possuam conhecimentos equivalentes.



# Introdução ao CSS3

1

- ✓ História;
- ✓ Versões;
- ✓ Como funciona a linguagem CSS;
- ✓ Estilos Inline;
- ✓ CSS incorporado no arquivo HTML;
- ✓ Incluindo um arquivo CSS externo;
- ✓ Importando um arquivo CSS;
- ✓ HTML5 X CSS3;
- ✓ Estágios de padronização da W3C;
- ✓ Prefixos proprietários;
- ✓ Modularização.



**IMPACTA**  
EDITORA

## 1.1. História

A evolução da tecnologia, aliada à crescente demanda por necessidades antes pouco exploradas, tornam o trabalho de desenvolvimento tanto de sites como de aplicações web, uma tarefa que exige um aprendizado constante e intenso. Isso se deve ao fato de inúmeros novos dispositivos estarem disponíveis no mercado a cada dia, em variados formatos e tipos.

Atualmente os dispositivos móveis estão ganhando muito espaço no tempo do usuário, por sua comodidade e disponibilidade imediata. É mais cômodo abrir um tablet num clique de botão do que ligar um notebook para ler e-mails e realizar rotinas comuns.

Para atender essa demanda a W3C (World Wide Web Consortium), que é um consórcio que determina os padrões da internet, padronizou e continua trabalhando em uma linguagem para modernos sites e aplicações web. Essa linguagem é o HTML, que atualmente está em sua versão 5.

Desde seu surgimento, o HTML sempre exibiu as informações para o usuário de uma forma amigável, mas para formatar cores, tamanhos, espaços e posições, era necessária uma tecnologia própria para tal. A W3C então criou uma linguagem de estilos para formatar espaços, texto, suas cores e tamanhos, conhecida como folhas de estilo em cascata, que vem do acrônimo em inglês **CSS – Cascading Style Sheet**.

## 1.2. Versões

É importante mencionar que, quando foi lançada a linguagem HTML em sua primeira versão, o criador da World Wide Web Sir Tim Berners-Lee, acreditava que o navegador do usuário é que deveria ser responsável por apresentar o conteúdo da forma esperada pelo usuário. Evidentemente não foi a melhor solução, uma vez que cada navegador tinha sua própria interpretação de formatação de conteúdo.

### 1.2.1. CSS1

Com o objetivo de padronizar as formatações, foi lançada em 17 de dezembro de 1996 como recomendação oficial a CSS1 (<http://www.w3.org/TR/CSS1/>), que conseguia formatar conteúdo, cores e outros recursos básicos.

### 1.2.2. CSS2

Em 1998 a web já estava mais madura e as novas necessidades da linguagem de formatação já estavam muito claras. É lançada então a **CSS2**, que possuía ferramentas úteis para o desenvolvimento web, tais como recursos para formatação de um arquivo para impressão, formatação completa de fontes, posicionamento, novos tipos de seleção de conteúdo denominados seletores, novo formato de posicionamento de conteúdo chamado box model, e muito mais. Em junho de 2011, a linguagem é oficialmente atualizada para **CSS 2.1**.

## 1.2.3. CSS3

Como mencionado anteriormente, com a rápida evolução da tecnologia e sua consequente criação de novos dispositivos para acessar a web, foi necessária mais uma atualização, e desde 29 de setembro de 2011 os seletores do CSS entraram em uma nova versão chamada Seletores Nível 3. Embora a linguagem CSS seja a mesma que a anterior, e tenha ganhado novos (e antes impensados) recursos fundamentais na tecnologia, como animações, transformações, nova estrutura de posicionamento e novos seletores, tudo isso fez com que a linguagem passasse a ser chamada de **CSS3** não somente por essas novidades, mas por ser o terceiro nível de progresso da especificação.

 CSS3 se refere ao terceiro nível da especificação da linguagem CSS e não à versão de seus seletores.

## 1.3. Como funciona a linguagem CSS

A linguagem CSS é uma linguagem de estilos, e é dividida em uma estrutura de três partes: “Quem”, “O quê” e “Como”.

- **Quem:** Para aplicar um efeito em um documento web, é necessário determinar quem desejamos formatar; é possível formatar qualquer objeto de um site ou aplicativo web, como um título, uma imagem, um texto, por exemplo.
- **O quê:** Após escolhermos quem deverá ser formatado, é necessário determinar o que desejamos formatar neste objeto; se o objeto escolhido for um texto, podemos formatar cor, tamanho, posicionamento, transparência e muito mais.
- **Como:** Sabendo que objeto queremos formatar e o que desejamos formatar nele, agora é necessário especificar o valor desta formatação; por exemplo, se escolhemos como objeto um título `<h1>` de um documento HTML, e desejamos formatar a cor deste título, podemos escolher cinza como valor da formatação.

O nome técnico que damos a esse processo é descrito na estrutura a seguir, denominada **Declaração de Estilo**, que é o formato utilizado pela linguagem CSS:

### QUEM{O QUÊ:COMO}

- **Seletor:** “Quem” representa o seletor, que é o elemento da linguagem HTML que queremos selecionar.
- **Propriedade:** “O quê” representa a propriedade que desejamos formatar, como a cor da fonte de um texto.
- **Valor:** “Como” representa o valor da propriedade declarada, por exemplo `#777`, que representa uma cor em formato hexadecimal.

```
Seletor{Propriedade:Valor}
```

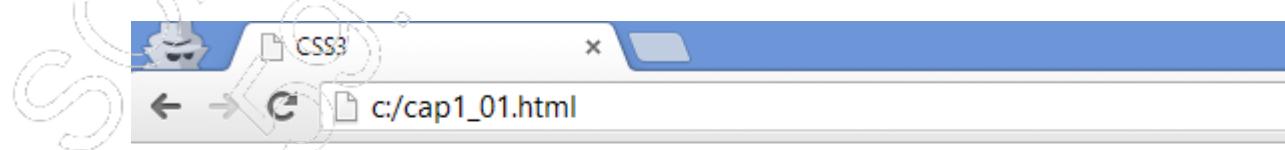
## 1.4. Estilos Inline

Para aplicarmos os estilos CSS, podemos criar um atributo **style** no elemento HTML que desejamos formatar, e então esse estilo será aplicado exclusivamente a esse elemento.

Devemos utilizar este recurso com cautela, uma vez que o objetivo principal da CSS é tornar o código HTML independente do estilo.

```
1 <!doctype HTML>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>CSS3</title>
6   </head>
7   <body>
8     <main role="main">
9       <p style="font-family:arial;font-size:20px;color:#777;">
10      Formatando o conteúdo de um texto
11      com a formatação Inline
12    </p>
13  </main>
14 </body>
15 </html>
```

Veja o resultado no navegador:



Formatando o conteúdo de um texto com a formatação Inline

Desta forma, o parágrafo representado pelo elemento **<p>** está formatado com o estilo de tipo de fonte, tamanho da fonte e cor do texto escolhidos.

## 1.5. CSS incorporado no arquivo HTML

Uma forma de incluir a formatação de uma folha de estilo em um documento HTML é por meio da tag **<style>**, que permite a inclusão de um bloco de folha de estilos dentro de um documento HTML.

Observe a formatação de um título de um documento HTML.

```
1  <!doctype HTML>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title>CSS3</title>
6  <style type="text/css" rel="StyleSheet">
7      header{
8          font-family: arial;
9          font-size: 20px;
10         color:#E44D3A;
11         font-weight: bold;
12     }
13 </style>
14 </head>
15 <body>
16 <main role="main">
17     <header> Texto no cabeçalho do documento</header>
18     <p>
19         Conteúdo principal
20     </p>
21 </main>
22 </body>
23 </html>
```

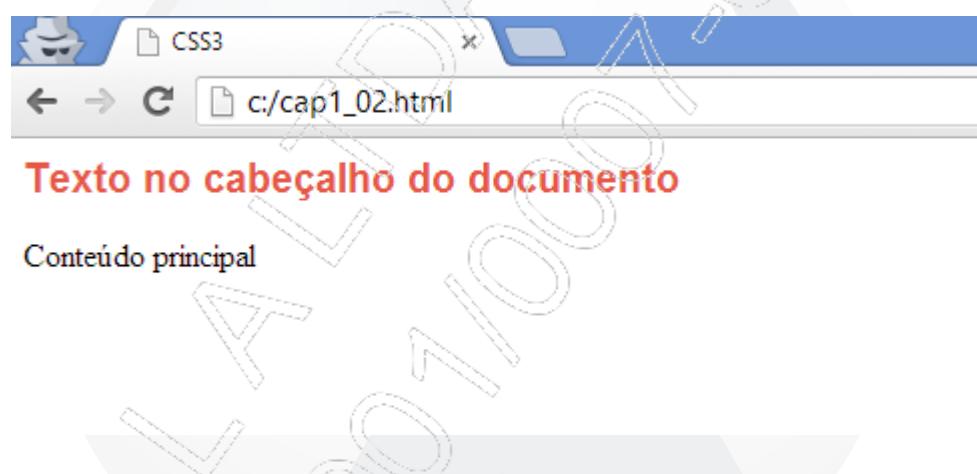
Percebemos que o exemplo em questão possui a tag **<style>** declarada no elemento **<head>**, o que significa que todo o seu conteúdo será carregado antes da exibição do documento pelo navegador.

Dentro do elemento **<style>**, percebemos o atributo **type** com o valor “**text/css**”, informando ao navegador em conjunto com o atributo **rel**, qual será a linguagem de estilo que será adotada por este documento, e o tipo “**text/css**” será sempre o padrão, o que torna esse atributo opcional.

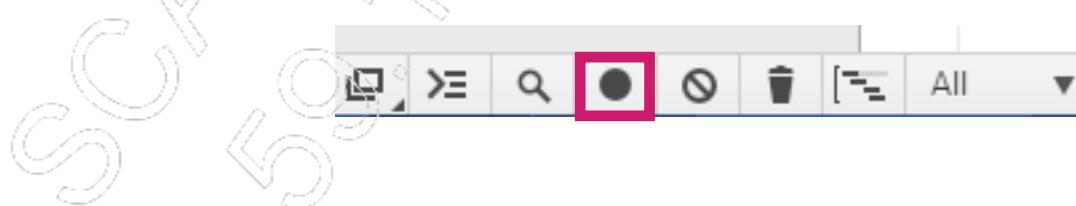
## CSS3

---

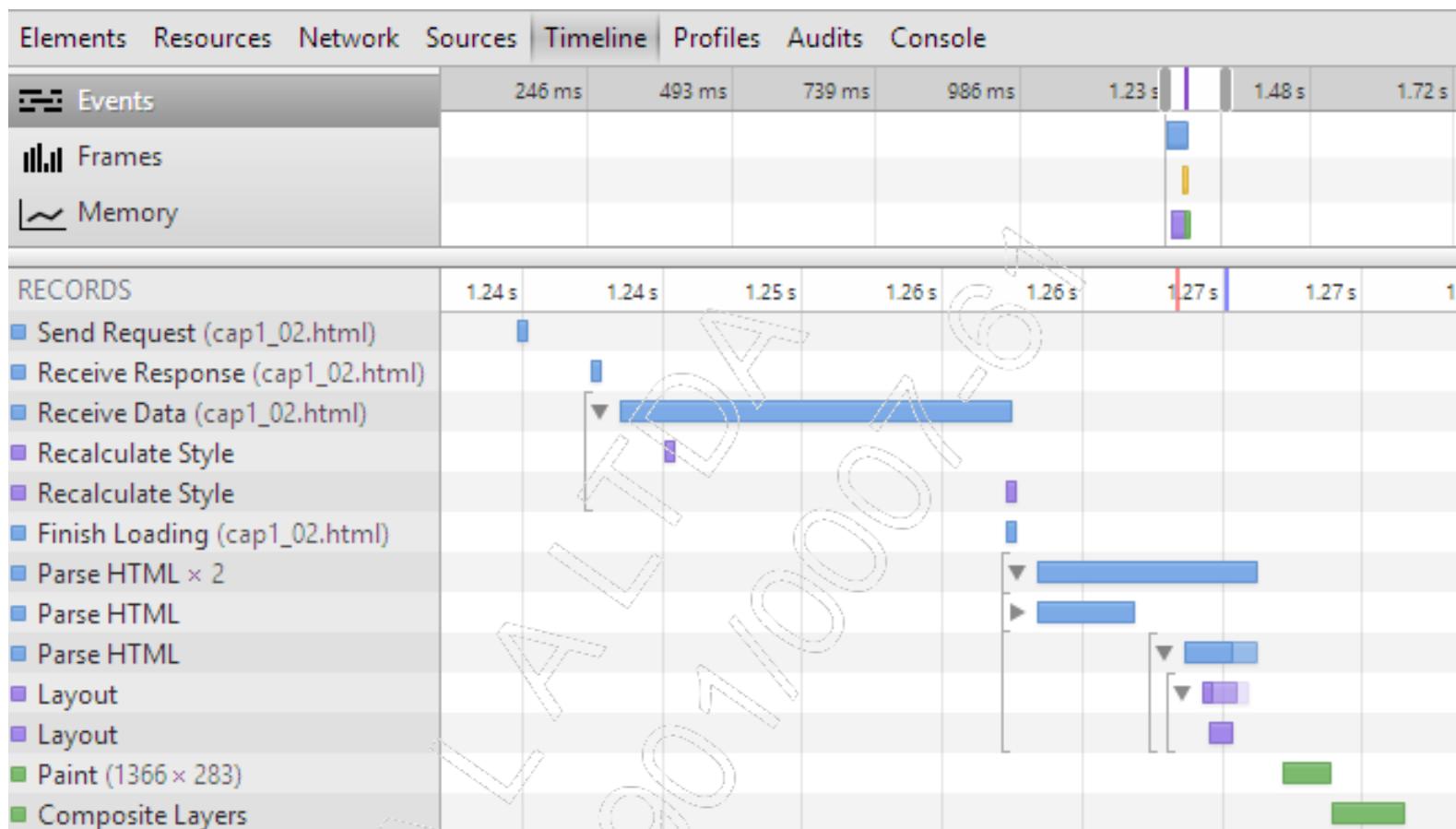
Vejamos o resultado do código acima, que utilizou o seletor **header** para selecionar a tag `<header>` e aplicar a ela as declarações de tipo, tamanho e cor de fonte, bem como a aplicação de negrito no texto.



Caso estejamos utilizando o navegador Google Chrome podemos, por meio da tecla de atalho CTRL + SHIFT + J, abrir a ferramenta **Developer Tools** do Google Chrome. Selecionando a aba **Timeline**, podemos clicar no símbolo de gravação em destaque na imagem que segue:



E depois de ativada a gravação da Timeline, basta pressionar F5 para recarregar a página, e analisar o tempo de carregamento do documento e como o navegador executou a interpretação do HTML.



Percebemos que o maior tempo para exibir o documento foi do download do arquivo, e da interpretação do HTML por parte do motor do navegador. Então os estilos são analisados e o layout é enfim renderizado de forma contínua e sequente.

Temos uma vantagem em utilizar o elemento `<style>` incorporado no próprio documento. Cada vez que for preciso alterar o CSS, o elemento não ficará armazenado em cache, e será recarregado toda vez que o arquivo HTML também for carregado. Por este motivo, em ambiente de desenvolvimento, pode ser útil que primeiro seja utilizada a folha de estilo incorporada ao arquivo HTML, e após os ajustes necessários, o mesmo seja alocado como um arquivo externo, para possível utilização em outros arquivos HTML.

## 1.6. Incluindo um arquivo CSS externo

A forma mais comum para trabalharmos com arquivos de folhas de estilos CSS é através da inclusão de um arquivo externo, o que é possível por meio do elemento `<link>`.

- Arquivo: `cap1_03.html`

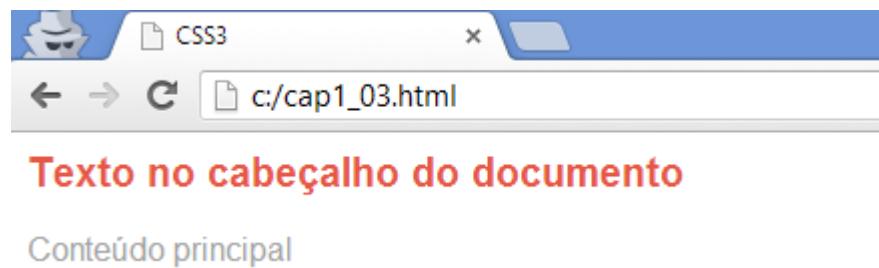
```
1 <!doctype HTML>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>CSS3</title>
6     <link rel="StyleSheet" href="css/estilo1.css" type="text/css">
7   </head>
8   <body>
9     <main role="main">
10       <header> Texto no cabeçalho do documento</header>
11       <p>
12         Conteúdo principal
13       </p>
14     </main>
15   </body>
16 </html>
```

O atributo `rel` determina a relação do arquivo com o documento atual;  
O atributo `type` determina o tipo de linguagem do arquivo em questão;  
O atributo `href` determina a localização do arquivo a ser incluído;  
O atributo `media` determina em que tipo de mídia esse arquivo deve ser carregado, por exemplo `print`, para impressão.

- Arquivo: `css/estilo1.css`

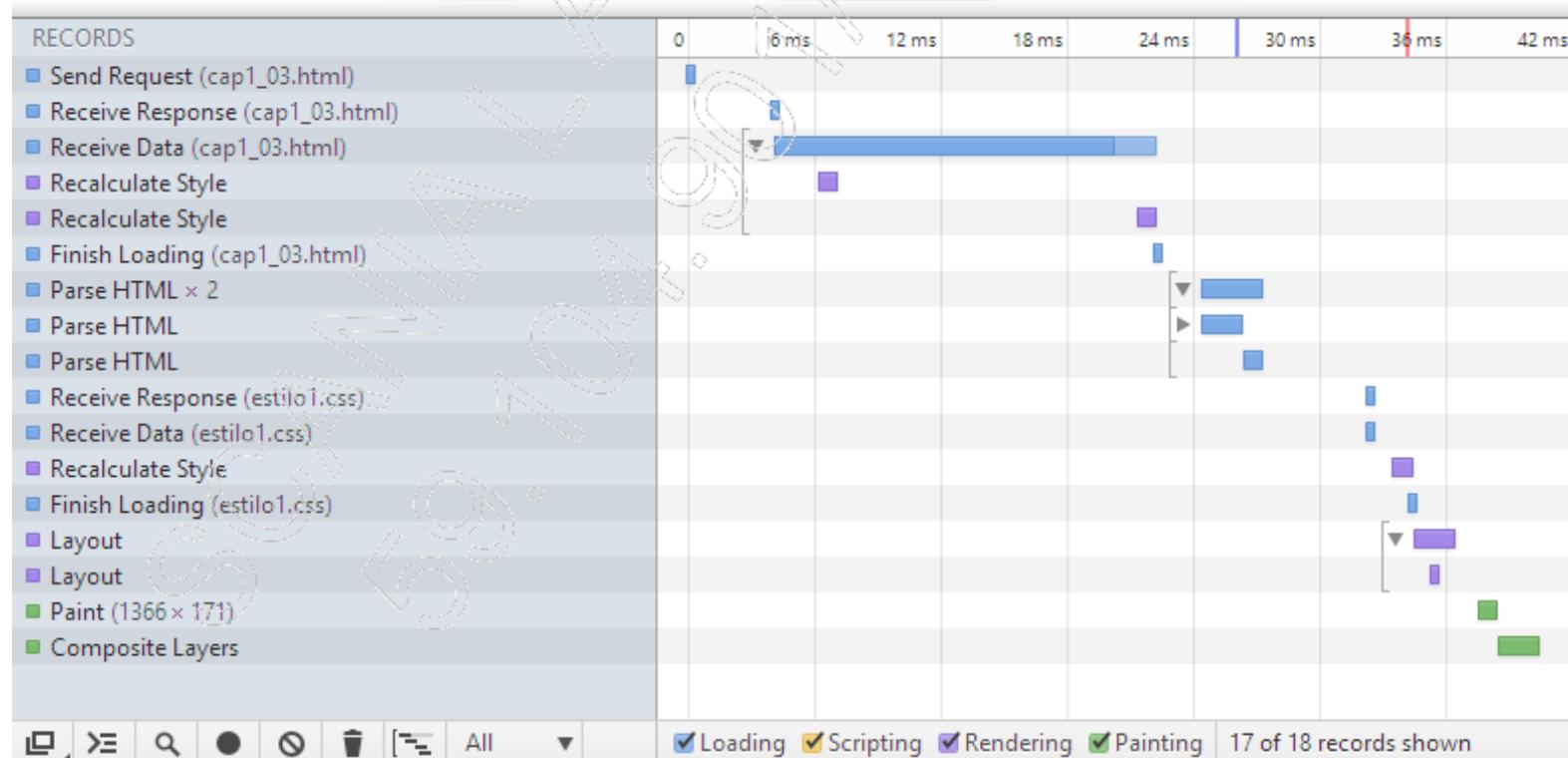
```
1 @charset "UTF-8";
2
3 header{
4   font-family: arial;
5   font-size: 20px;
6   color:#E44D3A;
7   font-weight: bold;
8 }
9 p{
10   color:#999;
11   font-family: arial;
12 }
```

Produzindo o resultado a seguir:



Observe que o arquivo **estilo1.css** possui uma declaração de **charset** como UTF-8 para evitar problemas de acentuação.

O tempo de carregamento de um arquivo CSS externo faz com que o tempo de processamento da página seja maior. No entanto, esse tempo é compensado uma vez que o arquivo CSS será utilizado em todo o site ou aplicação web, e logo as próximas páginas já terão o arquivo CSS carregado em cache.



## 1.7. Importando um arquivo CSS

Por meio da diretiva **@import** é possível carregar um arquivo CSS dentro de outro arquivo CSS, tratando-o como um arquivo de inclusão.

No exemplo a seguir, o arquivo **cap1\_04.html** irá carregar o arquivo **estilo2.css**, que por sua vez irá aplicar a folha de estilos ao documento em questão. Também neste arquivo será incluído o arquivo **estilo3.css**.

```
1 <!doctype HTML>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>CSS3</title>
6     <link rel="StyleSheet" href="css/estilo2.css" type="text/css">
7   </head>
8   <body>
9     <main role="main">
10       <header> Texto no cabeçalho do documento</header>
11       <p>
12         Conteúdo principal
13       </p>
14       <p class="texto-destaque">
15         Aplicando um estilo com base no arquivo estilo3.css <br/>
16         que está sendo carregado dentro do arquivo estilo2.css
17       </p>
18     </main>
19   </body>
20 </html>
```

- Arquivo: **estilo2.css**

```
1 @charset "UTF-8";
2
3 @import url('estilo3.css');
4▼ header{
5   font-family: arial;
6   font-size: 20px;
7   color:#E44D3A;
8   font-weight: bold;
9 }
10▼ p{
11   color:#999;
12   font-family: arial;
13 }
```

Observe que o arquivo de estilos está utilizando a diretiva **@import** em conjunto com a função **url**, para passar o parâmetro do arquivo a ser carregado.

Agora o arquivo **estilo3.css** que está sendo carregado como arquivo de inclusão no arquivo **estilo2.css**:

```
1 @charset "UTF-8";
2
3 .texto-destaque{
4     font-family: arial;
5     font-size: 18px;
6     color:#34f;
7     font-weight: bold;
8 }
```

Veja o resultado:



Texto no cabeçalho do documento  
Conteúdo principal  
Aplicando um estilo com base no arquivo estilo3.css  
que está sendo carregado dentro do arquivo estilo2.css

## 1.8. HTML5 X CSS3

É importante mencionar que, inicialmente, a W3C tratava o CSS3 como uma tecnologia abaixo do HTML5, porém essa implementação agora é independente. Sendo assim, o nível de evolução da linguagem CSS independe das revisões e padronizações do HTML5 e suas tecnologias.

O CSS3 se tornou poderoso a ponto de se converter numa tecnologia independente, embora ainda precise de uma estrutura HTML para ser aplicada.

## 1.9. Estágios de padronização da W3C

Quando aprendemos alguns recursos novos de HTML5 ou CSS3, é importante saber em qual grau de padronização tal tecnologia se encontra. Isso se deve ao fato do processo de padronização de uma tecnologia, que começa como um rascunho, ir sofrendo alterações, até se tornar uma recomendação oficial do W3C.

Temos a seguir uma tabela das etapas de padronização utilizada pela W3C para padronizar a linguagem HTML5, CSS3 e outras tecnologias.

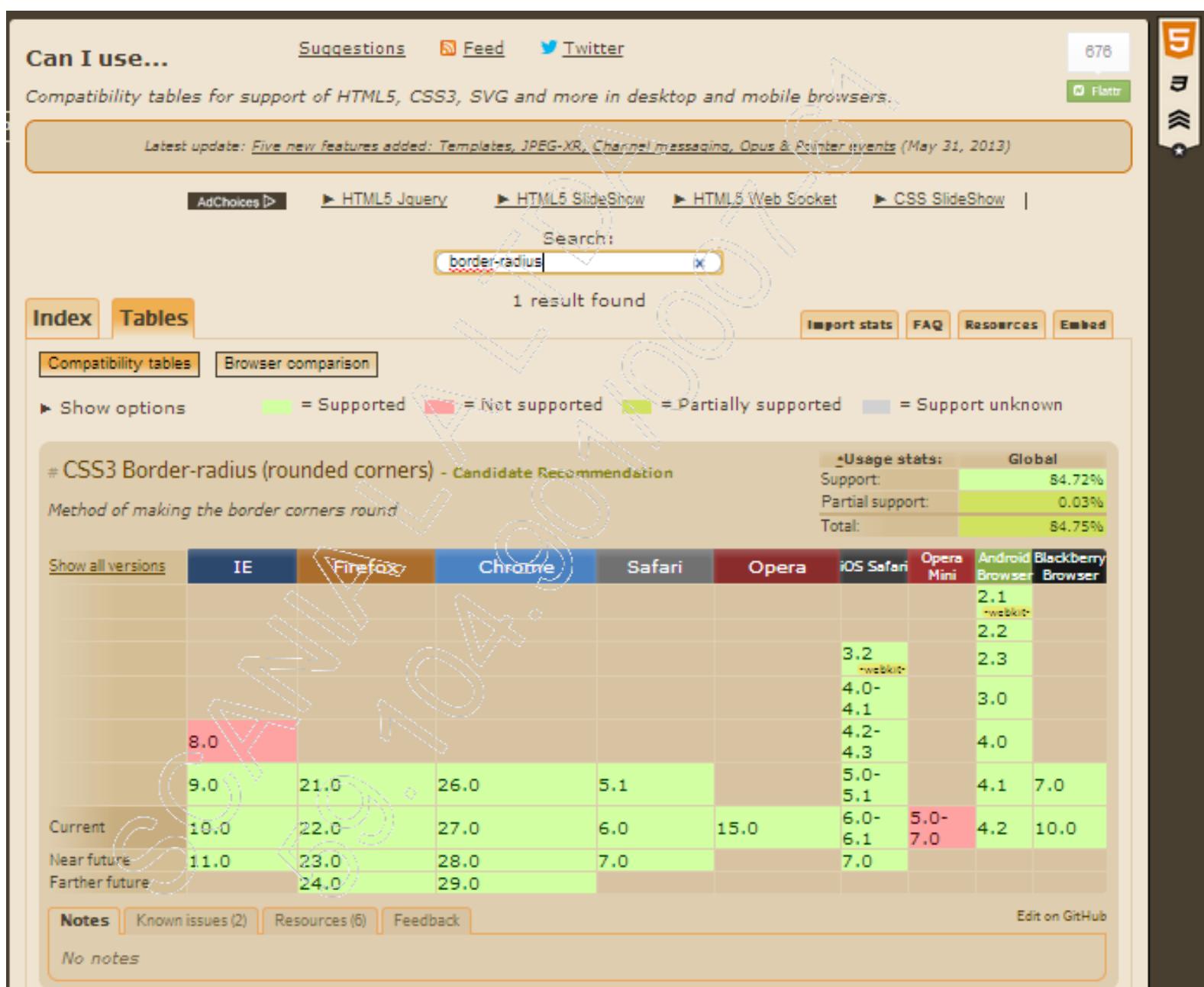
FASE	DESCRIÇÃO
WD	A primeira fase é uma ideia de um novo recurso, chamada de <b>Working Draft</b> , que significa “rascunho de trabalho”. Um esboço para melhorias é criado, e correções são feitas por parte da comunidade. Mudanças ou até mesmo o cancelamento do rascunho podem ocorrer nesta etapa.
LC	Quando uma ideia já está madura o suficiente para ser analisada como uma recomendação, é feita uma última chamada para os que ainda desejam aplicar sugestões ou mudanças nesta ideia. Esta última chamada em inglês é traduzida como <b>Last Call</b> .
CR	Quando todas as correções e ajustes são realizados, então chega a fase de <b>Candidate Recommendation</b> , que significa estar próxima de sua versão final. Embora mudanças possam ocorrer nessa fase, elas são mínimas e tratadas como ajustes.
PR	O penúltimo estágio é o <b>Proposed Recommendation</b> , significando que as implementações já foram testadas, e serão submetidas para homologação por parte do W3C.
REC	Quando a especificação atinge o status <b>W3C Recommendation</b> , significa que ela foi finalizada e aprovada pelos membros e diretores do W3C.

Quando um novo recurso está sendo criado e padronizado pela W3C, é importante ficarmos atentos em qual estágio da recomendação ele se encontra, uma vez que muitas mudanças podem ocorrer até chegarem à fase REC, caso ele esteja em WD, por exemplo.

É importante salientar também que os navegadores mais atuais, bem como suas versões para desenvolvedores, muitas vezes já implementam uma tecnologia mesmo que esta ainda não esteja completamente aprovada e homologada.

Existem alguns serviços que ajudam a verificar em qual estágio está determinada tecnologia e em qual navegador esse recurso está ativo. Uma referência bem conhecida e muito utilizada foi criada por Paul Irish ([www.caniuse.com](http://www.caniuse.com)).

Ao procurarmos por **border-radius**, por exemplo, que é um recurso da CSS3 que coloca cantos arredondados em elementos HTML5, notamos quais navegadores o suportam e quais não o suportam:



Apenas no Internet Explorer 8.0 e Opera Mini este recurso não está disponível, porém funciona normalmente nos outros navegadores.

## 1.10. Prefixos proprietários

Para utilizarmos alguns dos muitos recursos da CSS3 que ainda não estão homologados como recomendação W3C, faz-se necessário a utilização dos chamados **prefixos proprietários**.

Os prefixos proprietários atuam como uma referência ao motor de renderização dos navegadores, permitindo que determinado recurso seja utilizado apenas pelo navegador para o qual este recurso foi declarado.

Por exemplo, os navegadores Google Chrome e Apple Safari utilizavam em 2013 o motor de renderização criado pela Apple, mantido como um projeto de código aberto chamado **WebKit**. O Google anunciou posteriormente que utilizaria uma versão própria do WebKit chamada **Blink**.

Caso seja necessário aplicar uma propriedade da CSS3 que ainda não está oficialmente homologada pela W3C, será preciso mencionar o prefixo correspondente ao motor de renderização do navegador, neste caso o **-webkit** para que o motor WebKit interprete a propriedade em questão.

Exemplo:

**Elemento {**

```
-webkit-transform:rotate(5deg);  
-moz-transform:rotate(5deg);  
-o-transform:rotate(5deg);  
-ms-transform:rotate(5deg);  
transform:rotate(5deg);  
}
```

Segue uma tabela com os **prefixos proprietários** inerentes a cada navegador:

Prefixo proprietário	Motor de Renderização	Usado por
-webkit-	WebKit	Apple Safari & Safari for iOS Chromium / Google Chrome Navegadores baseados em Webkit
-moz-	Mozilla Foundation	Mozilla Firefox e Navegadores baseados em Gecko
-o-	Opera	Opera Desktop Browser, Opera Mini e Opera Mobile Nintendo DS & DSi Browser Nintendo Wii Internet Channel
-ms-	Trident (a.k.a. MSHTML)	Microsoft Internet Explorer

Importante mencionar que, a partir do segundo semestre de 2013, o navegador Google Chrome anunciou que passaria a utilizar o motor de renderização Blink, e neste caso não seria utilizado nenhum prefixo proprietário para testar uma nova funcionalidade neste navegador, como é usado hoje com o -webkit. No caso do Blink, será necessário habilitar o recurso que desejamos testar por meio do endereço **about:flags**, que deve ser digitado no endereço do navegador.

## 1.11. Modularização

Ao criar as folhas de estilos para o documento HTML, é necessário levar em consideração a independência do documento em relação ao estilo aplicado a este, uma vez que o mesmo documento pode ser exibido num notebook e num dispositivo móvel, como um tablet.

Para atender essa necessidade, as folhas de estilo devem ser desenvolvidas em um arquivo separado do documento.

Além de separar o estilo num arquivo externo, a criação das classes deve seguir o princípio da modularização, onde os estilos são distribuídos em várias classes, tornando a aplicação de estilos modular e independente. Vejamos o exemplo a seguir:

```
#inputNome{  
    width: 250px;  
    height: 30px;  
    border: solid 1px #eee;  
    border-radius:10px;  
    float:left;  
}  
  
#inputEmail{  
    width: 300px;  
    height: 30px;  
    border: solid 1px #eee;  
    border-radius:10px;  
    float:left;  
}
```

No exemplo, percebemos que os estilos de um campo texto são basicamente os mesmos, entretanto, estamos repetindo todas as propriedades.

Se trabalharmos com CSS modularizado, criamos apenas o estilo **input** e suas possíveis variações: **input-small**, **input-large**, e assim por diante.

```
.input{  
    border:solid 1px #eee;  
    border-radius:10px;  
    float:left;  
}  
.input-small{  
    width:100px;  
    height:30px;  
}  
.input-normal{  
    width:200px;  
    height:30px;  
}  
.input-large{  
    width:350px;  
    height:30px;  
}
```

Assim, ao aplicarmos as classes aos elementos input de formulários HTML, podemos agrupar várias classes para formatar uma entrada de dados.

```
class="input input-large"  
class="input input-small"
```

## Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- A nomenclatura CSS3 se refere ao terceiro nível da especificação da linguagem CSS, e não à versão de seus seletores;
- **Seletor:** É o elemento da linguagem HTML que queremos selecionar;
- **Propriedade:** É o item, dentro de um elemento, que desejamos formatar, como a cor da fonte de um texto;
- **Valor:** O valor representa a informação que deverá alimentar a propriedade de um elemento, onde #777 representa uma cor em formato hexadecimal, por exemplo;
- Uma forma comum de incluir a formatação de uma folha de estilo em um documento HTML, é por meio da tag `<style>`, que permite a inclusão de um bloco de folha de estilos dentro de um documento HTML;
- A forma mais comum para trabalharmos com arquivos de folhas de estilos CSS é por meio da inclusão de um arquivo externo, o que é possível através do elemento `<link>`;
- Por meio da diretiva `@import` é possível carregar um arquivo CSS dentro de outro arquivo CSS, tratando-o como um arquivo de inclusão;
- Os prefixos proprietários atuam como uma referência ao motor de renderização dos navegadores, permitindo que determinado recurso seja utilizado apenas pelo navegador para o qual este recurso foi declarado.

# Introdução ao CSS3

## Teste seus conhecimentos

1

SCANIA  
50.704.907-10007-67



**IMPACTA**  
EDITORA

**1. Para importarmos um arquivo CSS dentro de outro arquivo, que comando devemos utilizar?**

- a) <script type='text/css'></script>
- b) <link language="text/jss">
- c) @import
- d) @use
- e) @using

**2. Qual a sintaxe necessária para criar estilos dentro de um arquivo CSS?**

- a) Propriedade{Elemento:Valor}
- b) Elemento{Propriedade:Valor;}
- c) ID{Propriedade;Valor}
- d) Elemento(Propriedade:Valor)
- e) Elemento{Propriedade;Valor:}

### 3. Que atributo é utilizado para aplicar o estilo inline em uma tag <p>?

- a) <p STL="">texto</p>
- b) <p style="">texto</p>
- c) <p text-align="">texto</p>
- d) <p STYLE type ="">texto</p STYLE>
- e) <p rel="stylesheet">texto</p>

### 4. Qual prefixo proprietário devemos utilizar para aplicar um recurso ainda não homologado no navegador Firefox?

- a) -moz
- b) -ff
- c) -ms
- d) -webkit
- e) -o



# Seletores e Herança

2

- ✓ Seletores;
- ✓ Entendendo o DOM;
- ✓ Listagem de seletores da CSS3;
- ✓ Herança em CSS3;
- ✓ Especificidade e a propriedade **!important**;
- ✓ Boas práticas.



**IMPACTA**  
EDITORA

## 2.1. Seletores

Quando um documento HTML é carregado ao acessar um site, esse documento possui uma estrutura de marcação baseada em **tags**.

O conjunto de tags que formam esse documento é chamado de **Document Object Model**, ou simplesmente DOM. Para formatar o conteúdo de um documento HTML, é necessário conhecer o DOM e as ferramentas utilizadas na CSS para selecionar um ou mais itens dentro dessa estrutura.

As folhas de estilo CSS permitem selecionar os elementos do DOM utilizando os **seletores**. Estes podem selecionar elementos pelo atributo **class**, **id** ou **nome** do elemento, bem como selecionar um grupo de elementos por meio de **elementos filhos**, **elementos descendentes**, **pseudo-elementos**, e ainda agrupar vários elementos para compartilhar um mesmo estilo entre eles.

Neste capítulo veremos tudo sobre os **seletores** na CSS3, o que é **herança** e ainda como se utilizar de **boas práticas** e ter um código CSS mais limpo.

## 2.2. Entendendo o DOM

O **Document Object Model** é a estrutura de um documento HTML, que é carregada por um **User Agent**.

User Agent é o nome técnico dado aos navegadores ou programas utilizados para carregar o HTML. Isso ocorre devido a um documento HTML poder ser exibido não só por navegadores comuns, mas também por um leitor de e-mail, TV ou outro dispositivo que não utiliza necessariamente um navegador, mas precisa interpretar corretamente a linguagem HTML.

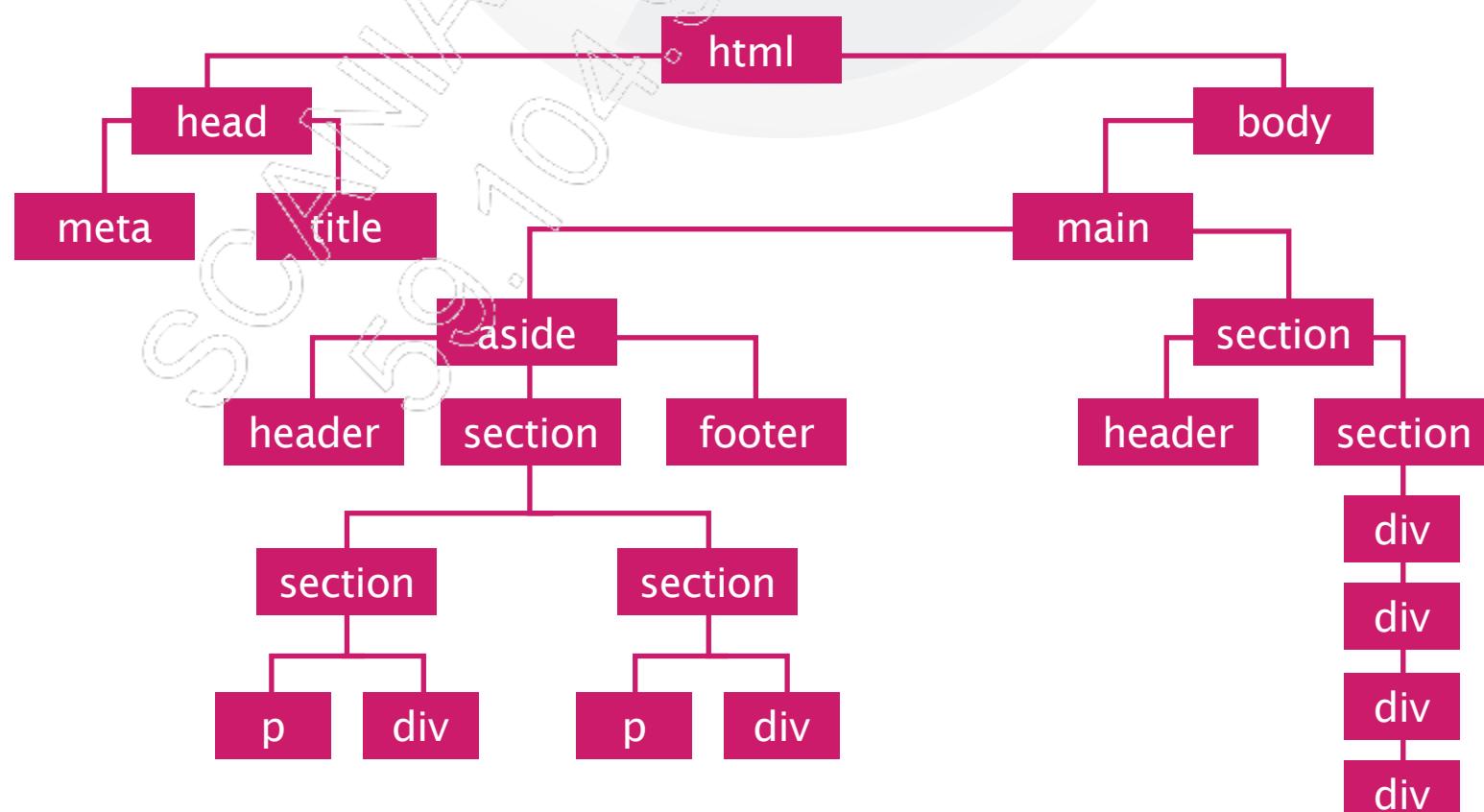
Conhecer a estrutura utilizada pelo DOM e como ele trabalha, permite que nosso código CSS e também o JavaScript sejam escritos de maneira legível pelos motores de renderização, otimizando o desempenho da aplicação e facilitando as tarefas de manutenção.

Temos a seguir um documento HTML5, e nele é possível notar como é feita a aplicação do DOM.

```

1  <!DOCTYPE html>
2  <html lang="pt-BR">
3      <head>
4          <meta charset="UTF-8">
5          <title>Entendendo o DOM</title>
6      </head>
7      <body><main role="main">
8          <aside><header>Caixas</header>
9              <section>
10                 <section class="widget-caixa">
11                     <p>Caixas de entrada</p>
12                     <div>iCloud</div>
13                 </section>
14                 <section class="widget-caixa">
15                     <p>Contas</p>
16                     <div>iCloud</div>
17                 </section>
18             </section>
19             <footer><strong>Atualizado</strong> data <strong>hora</strong></footer>
20         </aside>
21         <section>
22             <header>Arquivar | excluir | responder | novo e-mail</header>
23             <section>
24                 <div>De: Remetente@impacta.com.br</div>
25                 <div>Para: destinatario@impacta.com.br</div>
26                 <div>Assunto</div><div class="mensagem">Mensagem</div>
27             </section>
28         </section>
29     </main>
30 </body>
31 </html>
```

Para analisar melhor a forma como o DOM é interpretado, vejamos o mesmo código agora representado num formato de diagrama. Entenderemos o que são **elementos filhos, descendentes, irmãos** e como selecioná-los.



A árvore dos elementos aqui mostrada nos ajuda a entender como funciona a hierarquia de navegação dos elementos utilizados no DOM por meio das tags.

Vejamos agora os tipos de elementos encontrados na árvore do DOM:

- **Elemento:** Toda e qualquer tag que é encontrada em um documento HTML;
- **Elemento Ancestral:** O elemento HTML é ancestral a todos os outros elementos descendentes, e abaixo dele **head** e **body** também são ancestrais a todos os outros elementos;
- **Elemento Filho:** São elementos que estão imediatamente abaixo do elemento atual. Na árvore acima, **aside** é um elemento filho de **main**, assim como o elemento **footer** é um elemento filho de **aside**, porém **footer** não é um elemento filho de **main**;
- **Elemento Descendente:** É aquele que está abaixo de um elemento atual, podendo estar dentro de outros elementos filhos. Na árvore acima, **footer** é um elemento descendente de **main**, assim como os dois elementos **p** são elementos descendentes de **aside**, porém filhos de **section**.

Percebemos que o elemento **main**, que determina o conteúdo do site em sentido semântico, por exemplo, possui como elementos filhos **aside** e **section**, e estes por sua vez possuem seus elementos filhos e elementos descendentes. Antes de prosseguirmos, vamos entender melhor a definição de cada tipo de seletor disponível na CSS.

## 2.2.1. Selecionando pela classe

```
.nomeClasse{propriedade:valor}
```

Quando precisamos selecionar um elemento pelo atributo **class** deste, devemos utilizar o ponto (.) seguido do valor digitado no atributo **class**. Por exemplo, um elemento **section** que tenha um atributo **class='bloco'** será chamado de **.bloco** na folha de estilo CSS.

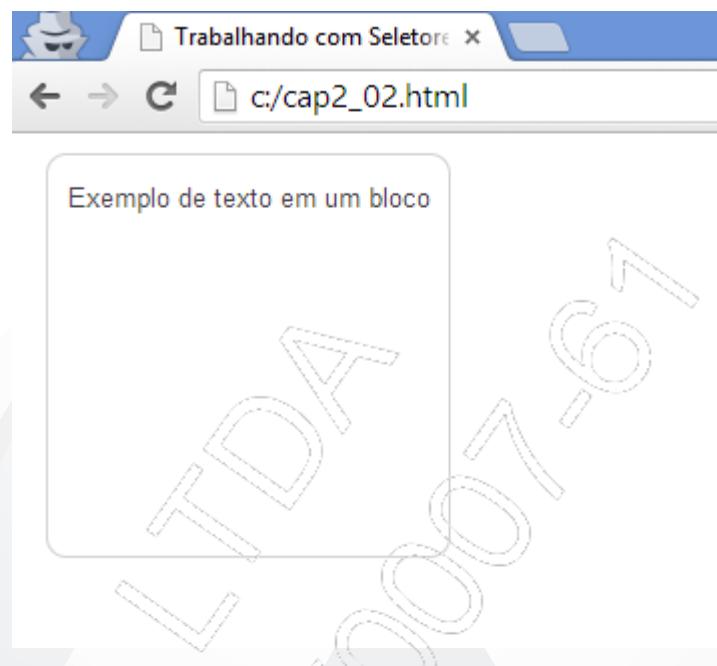
- Arquivo: **cap2\_02.html**

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Trabalhando com Seletores</title>
6     <link rel="stylesheet" type="text/css" href="css/estilo_02.css">
7 </head>
8 <body>
9 <section class='bloco'>
10    <p>Exemplo de texto em um bloco</p>
11 </section>
12 </body>
13 </html>
```

- Arquivo: **estilo\_02.css**

```
1 @charset "UTF-8";
2
3 .bloco{
4     width: 200px;
5     height: 200px;
6     margin: 10px;
7     text-align: center;
8     border-radius:10px;
9     border:solid 1px #ccc;
10    font-family: arial;
11    font-size: 13px;
12    color:#333;
13 }
```

Resultado no navegador:



Aplicar um estilo por meio do atributo class é uma boa prática de estilização, uma vez que a classe pode ser reutilizada quantas vezes forem necessárias.

## 2.2.2. Selecionando pelo ID

```
#nomeID{propriedade:valor}
```

A seleção de um elemento pelo atributo ID deste, deve ser feita utilizando o caractere (#).

O atributo ID de um elemento é utilizado para tornar o elemento único em um documento, evitando assim que se utilize o mesmo ID em outros elementos de um documento. Logo, quando utilizamos o seletor por ID, estamos aplicando um estilo muito específico e cuja formatação não será reutilizada no documento.

Um exemplo para aplicar o estilo ao elemento pelo ID é o logotipo de um site:

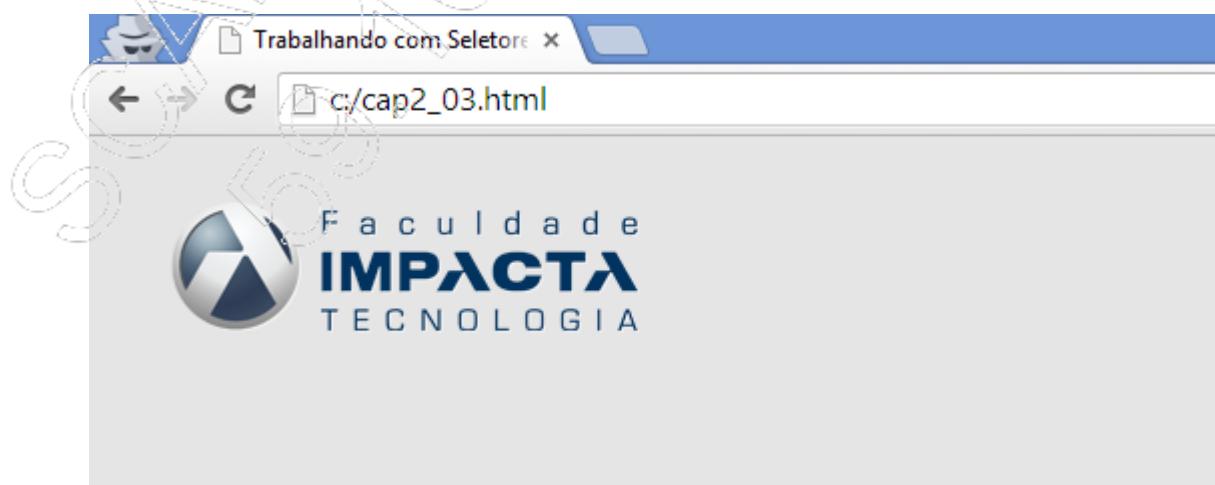
- Arquivo: **cap2\_03.html**

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Trabalhando com Seletores</title>
6     <link rel="stylesheet" type="text/css" href="css/estilo_03.css">
7 </head>
8 <body>
9 <header>
10    <div id="logo"></div>
11 </header>
12 </body>
13 </html>
```

- Arquivo: **estilo\_03.css**

```
1 @charset "UTF-8";
2
3 body{
4     background-color: #E2E2E2;
5 }
6 #logo{
7     width: 240px;
8     height: 70px;
9     margin: 33px;
10    background: url("../images/logo-impacta-header.png") no-repeat;
11 }
```

Veja o resultado no navegador:



## 2.2.3. Selecionando pelo elemento HTML

No exemplo anterior, percebemos a aplicação do estilo a um elemento HTML por meio do atributo **id**, e observamos também que foi aplicado um estilo ao fundo do documento. Para tal, o arquivo **estilo\_03.css** possui uma formatação sobre o elemento **body**.

```
1 @charset "UTF-8";
2
3 body{
4     background-color: #E2E2E2;
5 }
```

Sempre que precisarmos aplicar um estilo a um elemento e utilizamos o nome do elemento, devemos lembrar que esse estilo será aplicado a todos os elementos com este nome que são encontrados no documento, independentemente de sua localização.

## 2.3. Listagem de seletores da CSS3

Vejamos agora como selecionar os elementos dentro de um documento CSS3 e os seletores disponíveis para essa ação.

Padrão / Seletor	Nome	Descrição
*	Seletor universal	Seleciona todo e qualquer elemento.
E	Tipo de seletor	Se aplicado sem ponto ou cerquilha, se refere ao nome do elemento. Por exemplo, <b>p</b> seleciona todos os parágrafos <b>&lt;p&gt;</b> .
.nomeClasse	Seletor de Classe	Seleciona o elemento por meio de seu atributo <b>class</b> . Exemplo: o termo <b>bloco</b> seleciona qualquer elemento que tenha <b>class='bloco'</b> .
#nomeID	Seletor de id	Seleciona o elemento por meio de seu atributo <b>id</b> . Exemplo: <b>#logo</b> seleciona exatamente o elemento com <b>id='logo'</b> .

Padrão / Seletor	Nome	Descrição
<b>E &gt; F</b>	Seletor Filho	Seleciona o elemento <b>F</b> que for filho de <b>E</b> . Exemplo: <b>aside &gt; header</b> seleciona o elemento <b>header</b> que está imediatamente abaixo de <b>aside</b> .
<b>E F</b>	Seletor Descendente	Seleciona todos os elementos <b>F</b> encontrados abaixo do elemento <b>E</b> . Exemplo: <b>section p</b> seleciona todos os elementos <b>p</b> encontrados dentro de um bloco <b>section</b> .
<b>E:link</b>	Pseudo-classe para link	Seletor condicional, seleciona o elemento <b>E</b> se este for um link ainda não visitado.
<b>E:visited</b>	Pseudo-classe para link visitado	Seletor condicional, seleciona o elemento <b>E</b> se este for um link já visitado.
<b>E:active</b>	Pseudo-classe para elementos ativos	Seletor condicional, seleciona o elemento <b>E</b> no momento em que está sendo clicado.
<b>E:hover</b>	Pseudo-classe para evento Hover	Seletor condicional, seleciona o elemento quando o cursor do mouse for passado por cima deste, disparando o evento Hover da CSS.
<b>E:focus</b>	Pseudo-classe para evento Focus	Seletor condicional, seleciona o elemento quando este ganha o foco. Útil em formulários.
<b>E[atributo]</b>	Seletor de atributo	Seleciona qualquer elemento que possua o atributo em questão.
<b>E[atributo="valor"]</b>	Seletor de atributo	Seleciona o elemento que possua o atributo exatamente igual ao <b>valor</b> determinado.
<b>E[atributo^="valor"]</b>	Seletor de atributo	Seleciona o elemento que possua um atributo que inicie com a expressão determinada em <b>valor</b> .
<b>E[atributo\$="valor"]</b>	Seletor de atributo	Seleciona o elemento que possua um atributo que termine com a expressão determinada em <b>valor</b> .

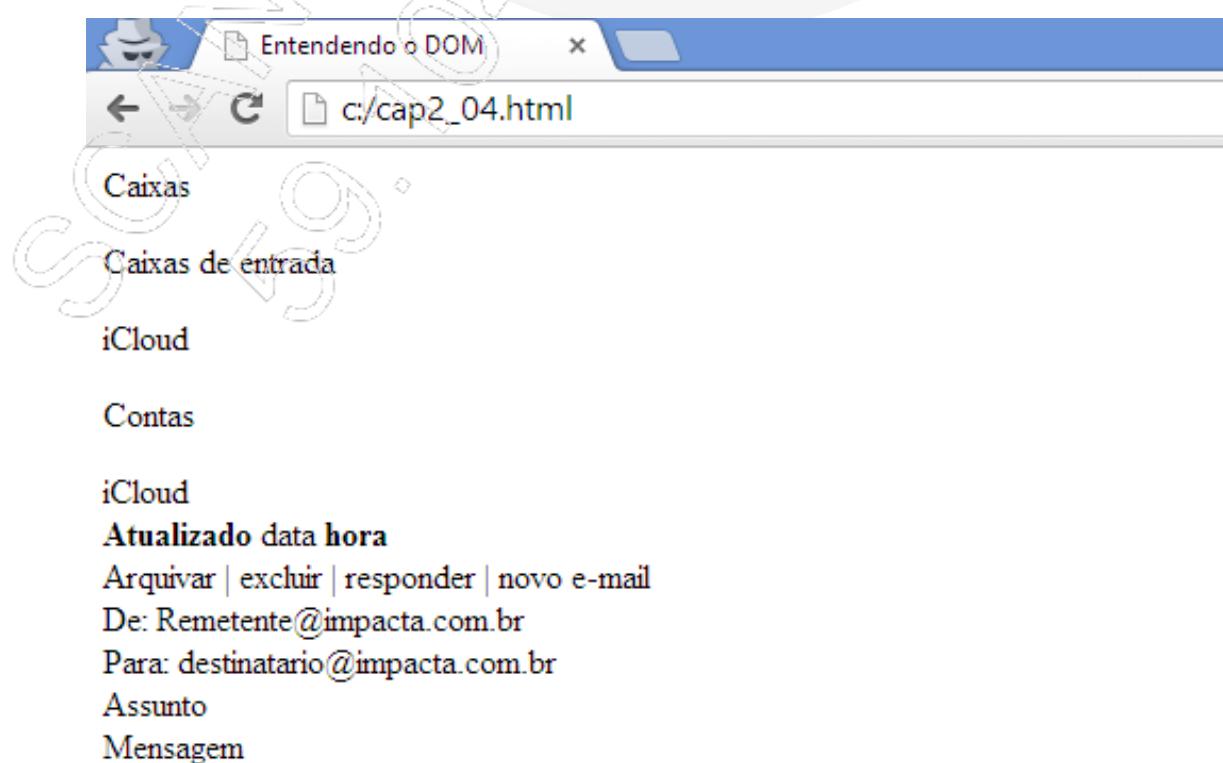
Padrão / Seletor	Nome	Descrição
<b>E:first-child</b>	Pseudo-classe para o primeiro elemento filho	Seleciona o primeiro elemento filho de um elemento. Útil em listas, tabelas ou conjuntos de tags subsequentes onde a formatação deve ser aplicada somente ao primeiro elemento da lista dentro de outro elemento.
<b>E:last-child</b>	Pseudo-classe para o último elemento filho	Semelhante à pseudo-classe <b>:first-child</b> , porém seleciona o último elemento filho de outro elemento principal.
<b>E:nth-child()</b>	Pseudo-classe para selecionar um elemento filho por posição	Permite selecionar um elemento filho dentro de uma lista pela posição deste começando em 1. Exemplos: O padrão <b>tr:nth-child(2)</b> selecionaria a segunda linha de uma tabela. Podemos utilizar os pseudo-seletores <b>:odd</b> ou <b>:even</b> para selecionar ímpar e par, respectivamente. Para colocar uma formatação zebreada em uma tabela, basta aplicar o efeito de background color: #eee dentro de um seletor <b>tr:nth-child(odd){ }</b>
<b>E + E</b>	Seletor irmão	Seleciona elementos que estão no mesmo nível. Exemplo: <code>&lt;h1&gt;&lt;/h1&gt;</code> <code>&lt;p&gt;&lt;p&gt;</code>
<b>E:checked</b>	Pseudo-classe checked	Seletor condicional para elementos que possuem o atributo <b>checked</b> .
<b>E:disabled</b>	Pseudo-classe disabled	Seletor condicional para elementos que possuem o atributo <b>disabled</b> . Útil para formatação de campos que estão desabilitados.

Vejamos um exemplo prático do uso de seletores, utilizando como base a estrutura HTML do primeiro exemplo deste capítulo.

- Arquivo: cap2\_04.html

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3   <head>
4     <meta charset="UTF-8">
5     <title>Entendendo o DOM</title>
6     <link rel="stylesheet" type="text/css" href="css/estilo.css">
7   </head>
8   <body><main role="main">
9     <aside><header>Caixas</header>
10    <section>
11      <section class="widget-caixa">
12        <p>Caixas de entrada</p>
13        <div>iCloud</div>
14      </section>
15      <section class="widget-caixa">
16        <p>Contas</p>
17        <div>iCloud</div>
18      </section>
19    </section>
20    <footer><strong>Atualizado</strong> data <strong>hora</strong></footer>
21  </aside>
22  <section>
23    <header>Arquivar | excluir | responder | novo e-mail</header>
24    <section>
25      <div>De: Remetente@impacta.com.br</div>
26      <div>Para: destinatario@impacta.com.br</div>
27      <div>Assunto</div><div class="mensagem">Mensagem</div>
28    </section>
29  </section>
30 </main></body>
31 </html>
```

Adicionamos o elemento **link** para carregar o arquivo CSS externo, chamado **estilo.css**. Sem o uso da folha de estilo, o resultado no navegador deverá ser semelhante à imagem que segue:



# CSS3

Para entendermos o uso dos seletores, vamos criar um layout para um web app leitor de e-mail similar ao do iPad.

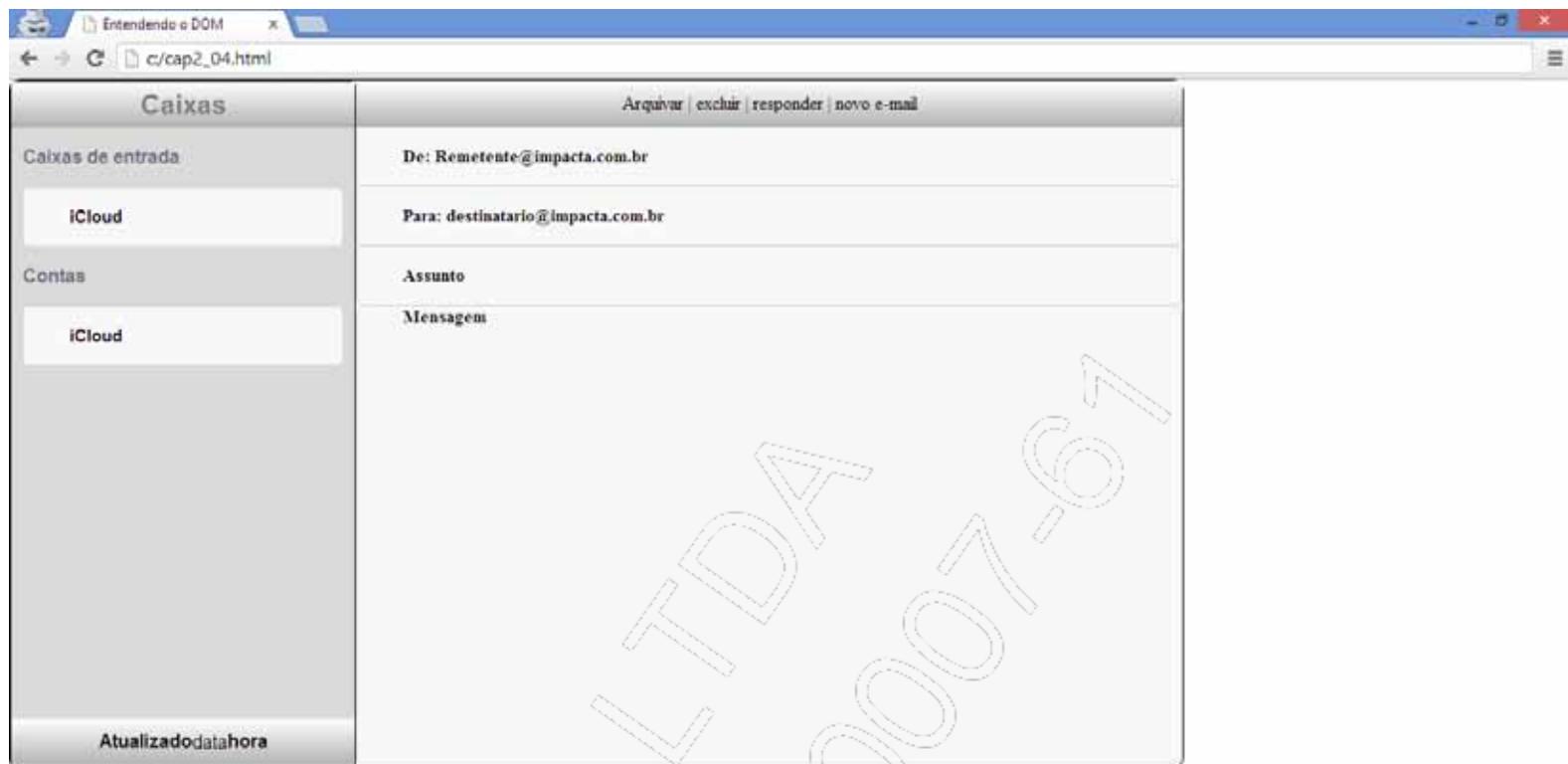
Começaremos com a formatação do menu esquerdo, com o bloco **aside**.

```
1 @charset "UTF-8";
2
3 body{
4     margin:0px 0px;
5 }
6 main{
7     width: 1024px; height:600px;
8     border-radius:10px; border:solid 1px;
9 }
10 aside{
11     width: 300px; height:100%;
12     border:solid 1px;
13     border-radius: 5px;
14     float:left;
15     background: #D6D6D6;
16     font-family: arial;
17 }
18
19 /*Seletor filho de aside*/
20 aside>header, aside>footer, section>header{
21     height: 40px;
22     display: flex;
23     align-items: center;
24     justify-content: center;
25     background-image: linear-gradient(to bottom, white, #ABABAB);
26 }
27 aside>header{color:#717276; font-size: 23px; font-weight: bold;}
28 aside>section{ height: 500px; }
```

E finalizaremos com o conteúdo do lado direito do bloco **section**.

```
29
30 /*Seletor de classe*/
31 .widget-caixa{
32     width:280px; margin: 5px 10px;
33 }
34 .widget-caixa>p{ font-size: 16px; color:#5F6674; font-weight: bold;}
35 .widget-caixa>div, main>section>section div{
36     height: 50px;
37     background-color: #F7F7F7;
38     display: flex;
39     align-items:center;
40     padding-left: 40px;
41     font-size: 15px;
42     font-weight: bold;
43     border-radius: 5px;
44     border:solid 1px #D7D8DC;
45 }
46 main>section>section div:last-child{
47     height: 400px;
48     border-radius: 10px;
49     align-items:baseline;
50 }
```

Produzindo então o resultado a seguir, no navegador **Chrome**:



As propriedades aqui utilizadas serão explicadas no decorrer dos próximos capítulos. Portanto, analisemos os seletores utilizados no exemplo em questão.

Começamos a formatação com seletores básicos, por meio do nome dos elementos **body**, **main** e **aside**.

Após a formatação dos seletores básicos, aplicamos a formatação aos seletores filhos de **aside**. Observe, na linha 20, que agrupamos três seletores desse exemplo para compartilharem a mesma formatação: **aside > header**, **aside > footer** e **section > header**. Esse recurso é útil quando um mesmo estilo deve ser aplicado a vários seletores.

Observe que os seletores filhos **aside > header**, na linha 27, possuem estilos específicos que não deveriam ser compartilhados pelos seletores agrupados da linha 20. Quando um estilo for específico para um elemento, ele deve ser declarado à parte.

# CSS3

---

Vejamos um exemplo prático do uso das pseudoclasses listadas na tabela de seletores.

- Arquivo: cap2\_05.html

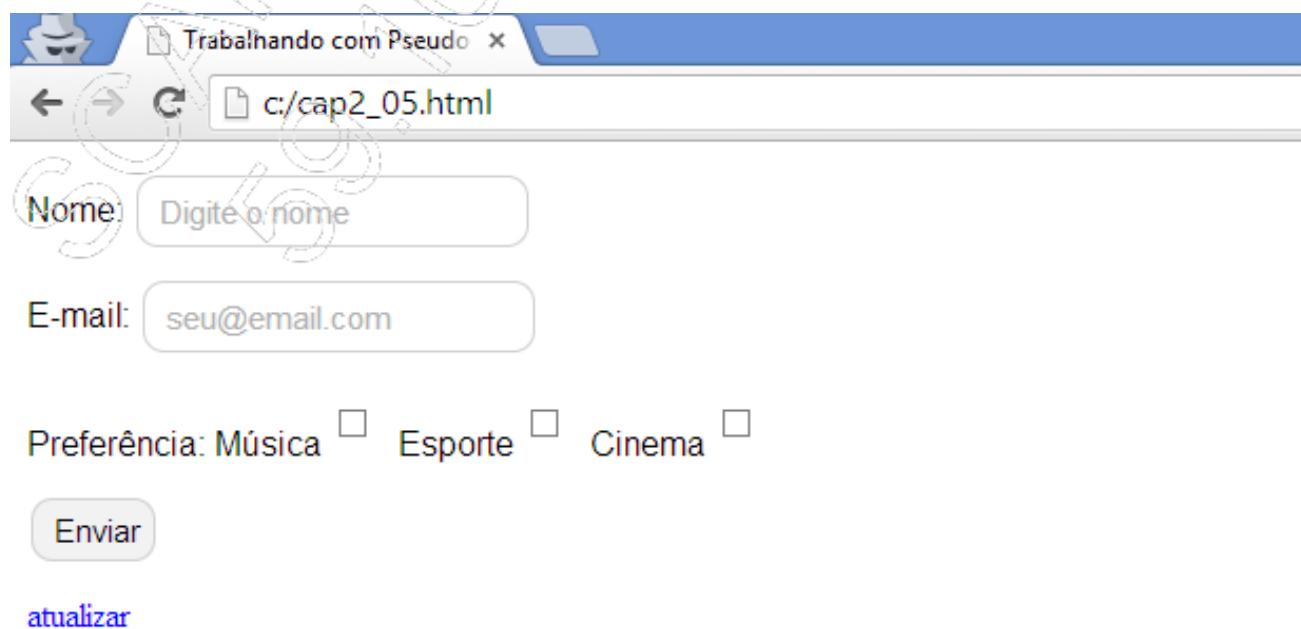
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Trabalhando com Pseudo-Classes</title>
6   <link rel="stylesheet" type="text/css" href="css/estilo_05.css">
7 </head>
8 <body>
9   <main role="main">
10    <form method="POST">
11      <p>        <label for="nome">Nome:</label>
12      <input type="text" name="nome" placeholder="Digite o nome" required>
13    </p>
14      <p>        <label for="email">E-mail:</label>
15      <input type="email" name="email" placeholder="seu@email.com" required>
16    </p>
17      <p>Preferência:
18      <label for="musica">Música</label>
19      <input type="checkbox" name="musica">&nbsp;
20      <label for="esporte">Esporte</label>
21      <input type="checkbox" name="esporte">&nbsp;
22      <label for="cinema">Cinema</label>
23      <input type="checkbox" name="cinema">
24    </p>
25    <p><input type="submit" value="Enviar"></p>
26  </form>
27  <a href="#">atualizar</a>
28 </main>
29 </body>
30 </html>
```

- Arquivo: **estilo\_05.css**

```
1 @charset "UTF-8";
2 p{
3     font-family: arial, sans-serif;
4 }
5 a{
6     text-decoration: none;
7     color: blue;
8 }
9 input{
10    border: solid 1px #ccc;
11    height: 30px;
12    padding-left: 10px;
13    font-size: 15px;
14    border-radius: 10px;
15 }
16 input:focus{
17     background-color: #F1F6FB;
18     color: #336699;
19 }
20
21 a:hover{
22     color: red;
23     text-decoration: underline;
24 }
25 a:active{
26     color: green;
27     text-decoration: underline;
28 }
```

Observamos a pseudoclasse **:focus** atuando sobre os elementos de formulário, bem como as pseudoclasses **:hover** e **:active** atuando em eventos de hyperlink.

O resultado no navegador será semelhante à imagem que segue:



## 2.4. Herança em CSS3

Como observado anteriormente, o modelo de documento do HTML e o DOM trabalham em um formato de árvore, onde o HTML é o ancestral e possui elementos filhos e descendentes. Quando aplicamos um estilo a um elemento do documento, caso este elemento possua filhos ou descendentes, estes podem herdar essa formatação, e a esse processo damos o nome de herança.

Para compreendermos melhor vejamos o exemplo a seguir.

- Arquivo: cap2\_06.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Trabalhando com Pseudo-Classes</title>
6   <link rel="stylesheet" type="text/css" href="css/estilo_06.css">
7 </head>
8 <body>
9   <main role="main">
10    <header>
11      <h1>Título da Página</h1>
12    </header>
13    <section>
14      <p>Texto dentro de uma seção. <span>formatação especial</span></p>
15    </section>
16  </main>
17 </body>
18 </html>
```

- Arquivo: estilo\_06.css

```
1 @charset "UTF-8";
2
3 p{
4   font-family: arial, sans-serif;
5   color:blue;
6   font-size: 14px;
7 }
8
```

Observe que, embora dentro do elemento `<p>` tenhamos outro elemento `<span>`, este recebe como herança a formatação do elemento `<p>`, deixando todo o parágrafo em azul.



## Título da Página

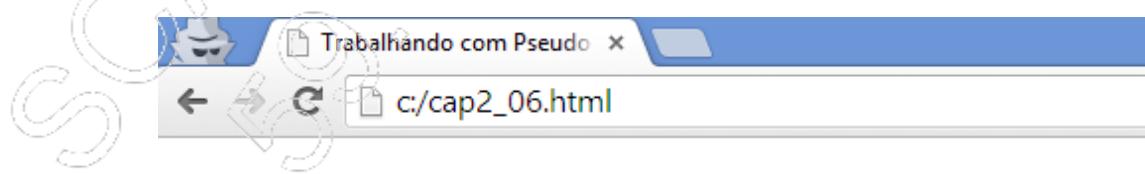
Texto dentro de uma seção. formatação especial

Agora se criarmos uma formatação especial para o elemento `<span>`, ele irá sobrepor a formatação herdada.

- Arquivo: `estilo_06.css`

```
1 @charset "UTF-8";
2
3 p{
4     font-family: arial, sans-serif;
5     color:blue;
6     font-size: 14px;
7 }
8
9 span{
10    color:green;
11    font-style: italic;
12 }
```

Veja o resultado:



## Título da Página

Texto dentro de uma seção. formatação especial

É importante lembrar que nem todas as propriedades podem ser herdadas por elementos filhos ou descendentes.

As propriedades a seguir, quando aplicadas, não permitem herança aos descendentes: **background**, **position**, **border**, **margin**, **padding**, **overflow**, **height**, **width**, **vertical-align**, **z-index**, **float**, **top**, **left** e **right**.

Como observado, algumas propriedades não são herdadas, mas é possível dizer a um elemento filho que este deve receber a mesma formatação do elemento pai, mesmo que esta não seja herdada.

Suponhamos que temos uma borda dentro de um elemento **section**, e temos um elemento **div** filho deste, e não mencionamos a borda. Como percebemos, a propriedade **border** não é passada como herança, conforme o exemplo abaixo:

- Arquivo: cap2\_07.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Trabalhando com Pseudo-Classes</title>
6   <link rel="stylesheet" type="text/css" href="css/estilo_07.css">
7 </head>
8 <body>
9   <main role="main">
10    <section class="fundo">
11      <h1>Título dentro de um section</h1>
12      <div class="fundo-div">
13          Texto dentro de um div
14      </div>
15    </section>
16  </main>
17 </body>
18 </html>
```

- Arquivo: estilo\_07.css

```
1 @charset "UTF-8";
2
3 .fundo{
4 border:solid 3px;
5 }
6
7 .fundo-div{
8
9 }
```

Veremos que a borda não é passada para o elemento filho.

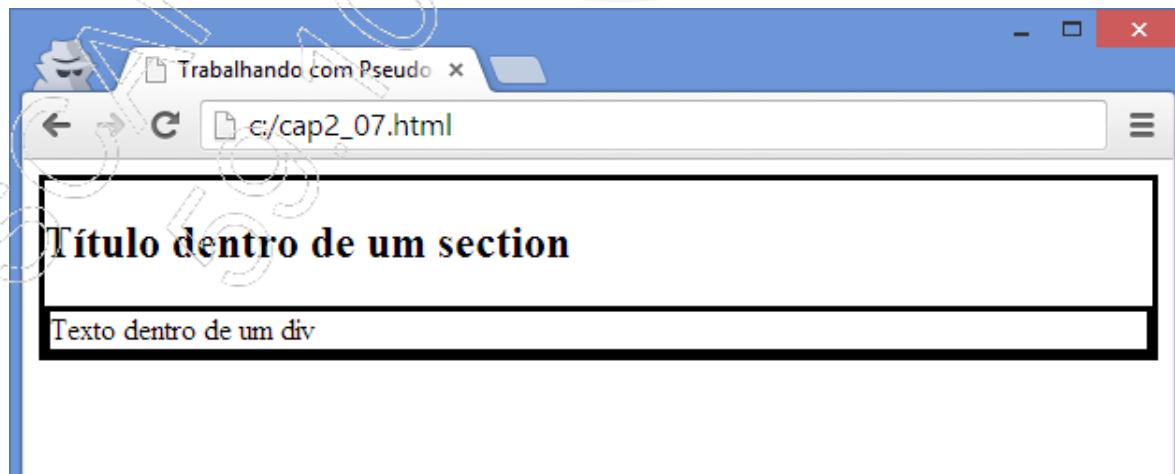


Porém, com o valor **inherit** na mesma propriedade, conseguimos forçar a herança deste. Desta forma estamos dizendo ao elemento filho que ele deve receber o mesmo valor da propriedade pai.

- Arquivo **estilo\_07.css**

```
1 @charset "UTF-8";
2
3 .fundo{
4 border:solid 3px;
5 }
6
7 .fundo-div{
8 border: inherit;
9 }
```

Veja o resultado:



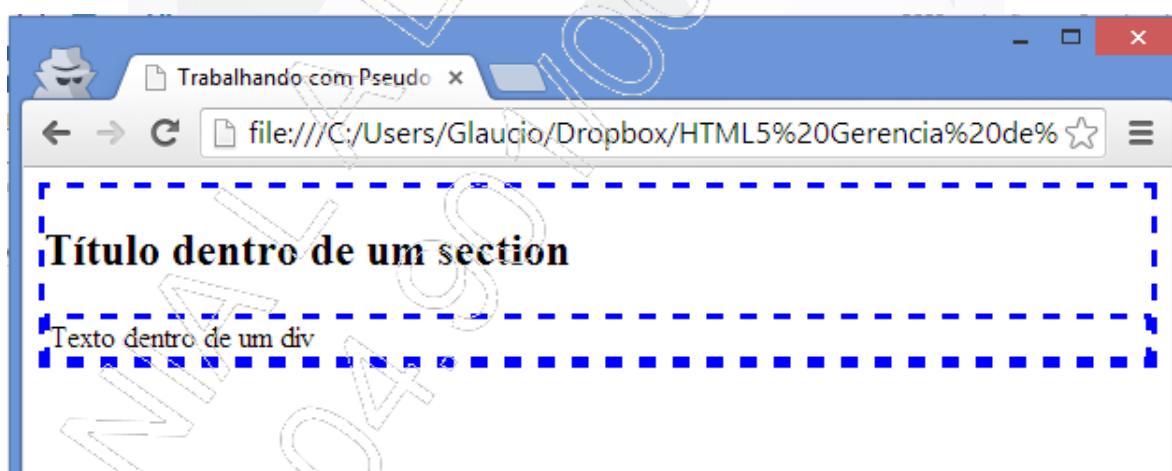
Se alterarmos o valor da propriedade pai, esta é automaticamente aplicada ao elemento filho, devido ao valor **inherit** da propriedade em questão.

CSS3

- Arquivo: `estilo_07.css`

```
1 @charset "UTF-8";
2
3 .fundo{
4 border:dashed 3px #00f;
5 }
6
7 .fundo-div{
8 border: inherit;
9 }
```

**Veja o resultado:**



Em se tratando de herança, o seletor universal (\*) deve ser utilizado com cautela, uma vez que pode influenciar no desempenho do documento. Aplicar um estilo a todos os elementos, embora a princípio pareça facilitar o trabalho do desenvolvedor, a médio e longo prazo pode tornar o código engessado.

## 2.5. Especificidade e a propriedade !important

Ao desenvolvermos a formatação dos estilos, precisamos lembrar que a ordem que criamos os estilos dentro de um arquivo CSS pode influenciar no resultado final, e caso essa ordem não esteja bem organizada, podemos ter muitas dificuldades com a manutenção do código. Não podemos esquecer que as folhas de estilo em cascata possuem essa primície de aplicar um estilo a um elemento, e em sua maioria aos elementos filhos deste.

Também é importante lembrarmos da precedência de seletores, onde **#id** é prioridade em cima da **.classe**, que é prioridade em cima do **elemento**.

Quando seletores descendentes herdam uma formatação, e se torna necessário sobrescrever esta formatação, podemos utilizar a propriedade **!important** para determinar quem deve ter prioridade na formatação do estilo.

Vejamos um exemplo:

- Arquivo: cap2\_08.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Trabalhando com Pseudo-Classes</title>
6     <link rel="stylesheet" type="text/css" href="css/estilo_08.css">
7   </head>
8   <body>
9     <main role="main">
10       <section>
11         <h1>Título dentro de um section</h1>
12         <p>Texto dentro de um section.</p>
13         <div class="fundo-div">
14           <p>Formatação específica</p>
15         </div>
16       </section>
17     </main>
18   </body>
19 </html>
```

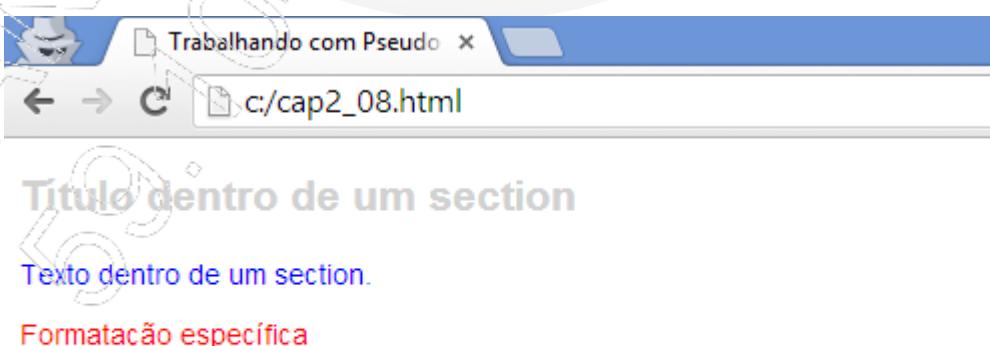
- Arquivo: **estilo\_08.css**

```
1 @charset "UTF-8";
2
3 section{
4     font-family:arial;
5     font-size:14px;
6     color: #ccc;
7 }
8
9 div p{
10    color:red;
11 }
12
13 p{
14    color:blue;
15 }
16 }
```

Observe que na estrutura atual, a ordem determinada diz que o texto dentro de um elemento **p** e dentro de um elemento **div** deve ser vermelho, e qualquer outro texto dentro do elemento **p** deve ser azul.

Mas imaginemos que desejamos quebrar essa hierarquia para **exceções** e não como regra. Neste caso, podemos utilizar a propriedade **!important** ao lado do valor que deve prevalecer.

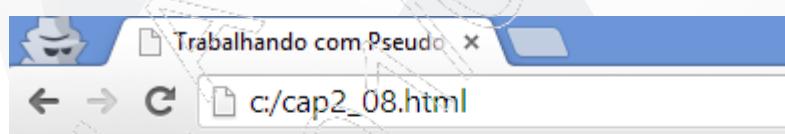
Sem a propriedade **!important**:



- Arquivo: **estilo\_08.css**

```
1 @charset "UTF-8";
2
3 section{
4     font-family:arial;
5     font-size:14px;
6     color: #ccc;
7 }
8
9 div p{
10    color:red;
11 }
12
13 p{
14    color:blue!important;
15 }
```

Agora com a propriedade **!important**:



Titulo dentro de um section

Texto dentro de um section.

Formatação específica

No entanto, quando utilizamos muitas vezes a propriedade **!important** para sobrescrever uma formatação, é possível que aconteça uma falha de projeto, ou que o mesmo possa ser revisto. É possível que a demanda de projetos tenha aumentado, ou que o conhecimento técnico do desenvolvedor tenha melhorado, então nesse caso torna-se necessário reestruturar as folhas de estilos para evitar o uso excessivo da propriedade **!important**.

## 2.6. Boas práticas

Quando trabalhamos com um código CSS de qualidade, além dela aumentar o desempenho do documento, também facilita a manutenção e organização dos estilos, possibilitando que o projeto web em questão seja escalável.

A seguir listamos algumas boas práticas de escrita CSS que, com pequenos ajustes, podem fazer uma grande diferença ao fim do projeto.

### 2.6.1. Evite qualificar demais

Evite colocar o caminho do seletor desnecessariamente, pois isso faz com que leve mais tempo a interpretação de seu CSS, além de ser uma forma suja de escrita.

- **Evite:**

```
section header #titulo{ }  
.bloco #destaque{ }
```

- **Prefira:**

```
#titulo{ }  
#destaque{ }
```

 Neste exemplo lembre-se que `#id` deve ser único no documento, por isso não há necessidade de chamá-lo com o caminho de ancestral a descendente.

## 2.6.2. Utilize declarações CSS compactas

Sempre que possível, utilize declarações de estilos compactas ao formatar espaçamento, margem e imagem de fundo, por exemplo. E essas propriedades serão melhor estudadas em capítulos posteriores.

- **Evite:**

```
.nomeClasse{  
    padding-top: 10px;  
    padding-bottom: 10px;  
    padding-left: 10px;  
    padding-right: 10px;  
    background: blue;  
    background-image: url("../image/fundo.png");  
    background-repeat: repeat-x;  
}
```

- **Prefira:**

```
.nomeClasse{  
    padding: 10px 10px 10px 10px;  
    background: blue url("../image/fundo.png") repeat-x;  
}
```

Nem todas as propriedades CSS possuem seu equivalente compacto.

## 2.6.3. Evite duplicar regras

Muitas vezes uma mesma regra pode ser utilizada em outro estilo neste caso.

- **Evite:**

```
.classeUm{  
    color: green;  
    background: gray;  
    font-size: 10px;  
}  
.classeDois{  
    color: green;  
    background: gray;  
    font-size: 10px;  
}
```

- **Prefira:**

```
.classeUm, .classeDois{  
    color: green;  
    background: gray;  
    font-size: 10px;  
}
```

 Caso uma das classes tenha declarações adicionais, é possível chamar o seletor novamente para colocar a declaração específica do mesmo, a este processo damos o nome de **condensar regras**.

## 2.6.4. Condensando regras

Quando dois estilos devem ser compartilhados pelos seletores, devemos aplicar a regra aos dois separando por vírgula, mas caso além destas regras estes possuam outras adicionais podemos fazer da seguinte maneira:

- **Evite:**

```
.classeUm{  
    color: green;  
    background: gray;  
    font-size: 10px;  
}  
  
.classeDois{  
    color: green;  
    background: gray;  
    font-size: 12px;  
}
```

- **Prefira:**

```
.classeUm, .classeDois{  
    color: green;  
    background: gray;  
}  
.classeUm{ font-size:10px; }  
.classeDois{ font-size:12px; }
```

## 2.6.5. Evite !important

Como já mencionado, o uso de **!important** é uma exceção, e indica que a ordem dos seletores não atingiu o resultado esperado. Caso seja muito utilizado em seu projeto, verifique se não é melhor reestruturar o arquivo CSS.

## Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- O **Document Object Model** é a estrutura de um documento HTML que é carregada por um **User Agent**;
- **User Agent** é o nome técnico dado aos navegadores ou programas utilizados para carregar o HTML;
- **Elemento**: Toda e qualquer tag que é encontrada em um documento HTML;
- **Elemento Ancestral**: O elemento HTML é ancestral a todos os outros elementos descendentes, e abaixo dele **head** e **body** também são ancestrais a todos os outros elementos;
- **Elemento Filho**: São elementos que estão imediatamente abaixo do elemento atual;
- **Elemento Descendente**: É aquele que está abaixo de um elemento atual, podendo estar dentro de outros elementos filhos;
- Quando precisamos selecionar um elemento pelo atributo **class** deste, devemos utilizar o ponto (.) seguido do valor digitado no atributo class;
- A seleção de um elemento pelo atributo ID do mesmo, deve ser feita utilizando o caractere (#);
- Sempre que precisamos aplicar um estilo a um elemento e utilizamos o nome deste, devemos lembrar que esse estilo será aplicado a **todos os elementos com este nome que são encontrados no documento**, independentemente de sua localização;
- Quando aplicamos um estilo a um elemento do documento, caso este elemento possua filhos ou descendentes, estes podem herdar essa formatação, e a esse processo damos o nome de **herança**;
- Quando seletores descendentes herdam uma formatação e se torna necessário sobrescrevê-la, podemos utilizar a propriedade **!important** para determinar quem deve ter prioridade na formatação do estilo.

2

# Seletores e Herança

## Teste seus conhecimentos

SCAMIA  
59.704.901  
SCAMIA  
59.704.901  
SCAMIA  
59.704.901  
SCAMIA  
59.704.901  
SCAMIA  
59.704.901



**IMPACTA**  
EDITORA

**1. Para selecionarmos o elemento input cujo name comece com txt, qual seletor devemos utilizar?**

- a) .input[name="txt"]
- b) input[name^="txt"]
- c) input(name^="txt")
- d) input[id\*="txt"]
- e) input[name\$="txt"]

**2. Qual seletor deve ser utilizado para selecionar um elemento div que seja filho de section e descendente de main, e que além disso possua o id=miniatura?**

- a) main>section div#miniatura
- b) main section div#miniatura
- c) #miniatura
- d) main section>div.minиatura
- e) #main#section>div#miniatura

**3. Como são chamados dois elementos que são filhos do mesmo pai? E qual seletor deve ser utilizado para esta situação?**

- a) elementos irmãos adjacentes, elemento1 + elemento2
- b) elementos descendentes, elemento1 < elemento2
- c) elementos irmãos adjacentes, elemento1 > elemento2
- d) elementos filhos adjacentes, elemento1 ! elemento2
- e) elementos em ID, elemento1 ~ elemento2

**4. Para determinar que uma declaração CSS é prioridade sobre qualquer outra. Qual recurso devemos utilizar?**

- a) priority!
- b) Importante
- c) !importante
- d) #priority
- e) !important

**5. Para selecionarmos o primeiro elemento li dentro de uma lista qual seletor devemos utilizar?**

- a) li:first
- b) li:first-child
- c) li#first-child
- d) ul> li.first-child
- e) ul > li:nth-child(1)

**6. Como podemos mudar a cor de fundo de um elemento input quando este ganha o foco?**

- a) input.focus{backgroundcolor:#eee;}
- b) input:focus[background-color:#eee;]
- c) input:onFocus{background-color:#eee;}
- d) input:focus{background-color:#eee;}
- e) #input.focus{background-color:#eee;}

2

# Seletores e Herança Mãos à obra!

SCAMIA 59.704.907-10007-67  
SCAMIA 59.704.907-10007-67



**IMPACTA**  
EDITORA

## Laboratório 1

### A - Trabalhando com seletores

1. Crie um arquivo chamado **cap2\_lab1.html** com a estrutura HTML a seguir:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Utilizando Seletores</title>
    <link rel="stylesheet" type="text/css" href="css/estilo_
lab2.css">
</head>
<body>
    <main role="main" class="fundo">
        <header class="topo">
            <div class="grid_12">
                <a href="#"></a>
                <nav>
                    <ul class="menu">
                        <li class="current"><a href="#">Home</a></li>
                        <li><a href="#">História</a></li>
                        <li><a href="#">Notícias</a></li>
                        <li><a href="#">Serviços</a></li>
                        <li><a href="#">Portfólio</a></li>
                        <li><a href="#">Fale-Conosco</a></li>
                    </ul>
                </nav>
            </div>
        </header>
    </main>
</body>
</html>
```

2. Crie um arquivo chamado **estilo\_lab2.css** com a estrutura CSS a seguir, aplicando os estilos aos seletores em questão.

```
@charset "UTF-8";
```

- Aplique ao elemento body:

```
body {  
    margin: 0px;  
    padding: 0px;  
    min-width: 960px;  
    background: #181514;  
}
```

- Aplique à classe fundo:

```
.fundo {  
    width: 100%;  
    min-height: 900px;  
    height: auto;  
    background: url("../images/bg-fundo.png") center 0 no-re-  
peat #110e0e;  
}
```

- Aplique ao elemento header:

```
.header {  
    height: 100px;  
    background-color: rgba(18,15, 14,0.8);  
}
```

# CSS3

---

- Aplique a todo elemento **a** que for filho de **li** que estiver dentro de **header**:

```
_____ {  
    padding: 38px 17px 38px 17px;  
    font-family: arial, sans-serif;  
    font-size: 18px;  
    text-decoration-transform: uppercase;  
    color: #fff;  
    text-decoration: none;  
}
```

- Aplique a todo **li** dentro de um **header**:

```
_____ {  
    display: inline;  
}
```

# Posicionamento e alinhamento

3

- ✓ Box Model na CSS3;
- ✓ Configurando largura e altura;
- ✓ Configurando a margem – Margin;
- ✓ Configurando espaçamento – Padding;
- ✓ Configurando posicionamento – Position;
- ✓ Elementos flutuantes para layout;
- ✓ Flexible Box Layout.

## 3.1. Introdução

Ao criarmos uma aplicação web, seja um portal de conteúdo, um site de comércio eletrônico ou mesmo recursos para redes sociais, é necessário projetar um layout bem estruturado, organizado e modular, de forma que permita ao desenvolvedor criar o site com seus elementos da forma como foi desenhado.

Para esta tarefa, um dos assuntos mais importantes que é preciso dominar é a forma como a CSS trabalha com o posicionamento e o alinhamento dos elementos em um site. Neste capítulo veremos o que é Box Model, como trabalhar com margem, espaçamento e posicionamento dos elementos, além de conhecer o novo formato de alinhamento de conteúdo da CSS3, chamado Flexible Box Layout ou simplesmente Flex Box.

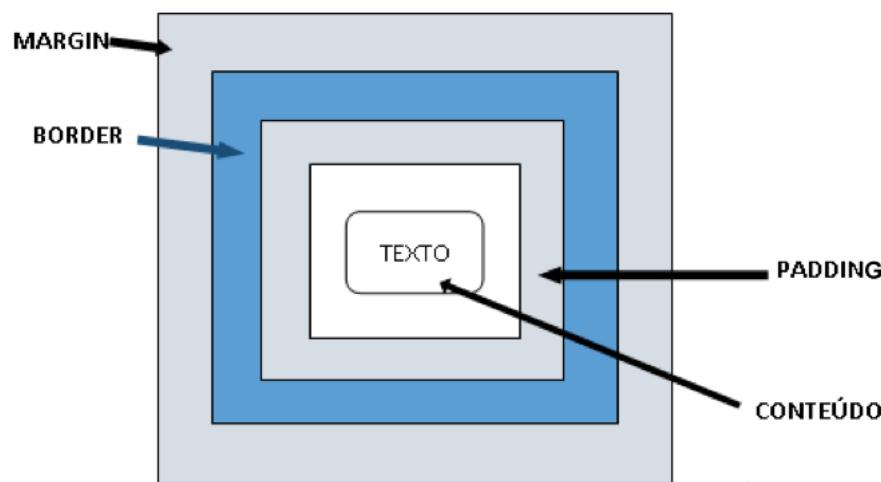
### 3.1.1. Box Model na CSS3

Um dos princípios básicos da formatação de layout em CSS é um conceito chamado CSS Box Model, onde cada elemento do HTML é encarado como um box, uma caixa. Tendo cada elemento o conceito de uma caixa, o trabalho de posicionamento, alinhamento, definição de largura, altura e cores é realizado inteiramente pela CSS.

O CSS Box Model pode ser dividido em dois tipos:

- Box Nível de Bloco – Block-level, onde uma `div`, um cabeçalho ou um parágrafo são exemplos de Boxes Nível de Bloco;
- Box Nível de Linha – Inline box, onde um elemento `<span>` ou uma imagem são exemplos de Boxes Nível de Linha.

A seguir, temos o diagrama da CSS Box Model juntamente com a descrição dos itens que a compõem, e veremos como utilizar seus recursos.



- **MARGIN:** Trata-se do espaço que separa o Box de outros elementos que compõem a página. Vale destacar que a margem (margin) não está dentro dos limites de um elemento. As margens são utilizadas com a finalidade de mover a caixa do elemento;
- **CONTEÚDO:** Trata-se dos itens contidos em um elemento, os quais permanecem dentro do container;
- **BORDER:** Trata-se dos limites de um elemento, isto é, das linhas de contorno do Box;
- **PADDING:** Trata-se dos espaços em branco existentes entre o conteúdo e os limites de um elemento.

A largura do box no qual está contido o elemento é determinada pela largura do conteúdo, somada às larguras referentes ao border e ao padding. Já a largura do elemento em si é determinada apenas pela largura ocupada por seu conteúdo, assim como sua altura.

Em uma página, todos os elementos visíveis são de **bloco** ou **inline**. Os elementos de bloco têm sua largura determinada pela largura do bloco em que se encontram. Eles se iniciam sempre em uma linha nova e, depois de finalizados, há uma nova mudança de linha. Já os elementos **inline** têm sua largura determinada apenas por seu conteúdo e, ao contrário dos elementos de bloco, não se iniciam sempre em uma nova linha. Dessa forma, temos que os elementos inline comportam-se da mesma maneira que um texto simples.

Como exemplo de elemento de bloco, podemos mencionar o **<section>**. Já como exemplo de um elemento inline, podemos mencionar o **<span>**.

## 3.1.2. Configurando largura e altura

Para definirmos o formato de um elemento, começamos com as dimensões horizontais e verticais. Para isso, utilizamos as propriedades **width** e **height**, bem como suas propriedades complementares: **min-width** para largura mínima e **max-width** para largura máxima, além de **min-height** e **max-height** para alturas mínima e máxima, respectivamente. Essas propriedades possuem o valor padrão **auto**.

Vejamos na tabela a seguir os valores das propriedades para largura e altura:

Propriedade	Valor inicial	Descrição
<b>width</b>	auto	Determina a largura do elemento, e pode ser aplicado a qualquer elemento, com exceção de textos ou linhas de tabelas. Caso o valor inicial seja mantido, a largura aumenta de acordo com o conteúdo do elemento;
<b>min-width</b>	0	Determina a largura mínima do elemento;
<b>max-width</b>	none	Determina a largura máxima do elemento;
<b>height</b>	auto	Determina a altura do elemento, e pode ser aplicado a qualquer elemento, com exceção de textos e altura de colunas de uma tabela. Caso o valor inicial “auto” seja mantido, ele poderá ser alterado pelo seu conteúdo.
<b>min-height</b>	0	Determina a altura mínima do elemento.
<b>max-height</b>	none	Determina a altura máxima do elemento.

Podemos determinar os valores em pixels ou ems, e ainda em porcentagem. Neste caso, é importante lembrar que esta porcentagem será calculada com referência à largura do elemento pai do elemento que estamos formatando.

Por exemplo: se um elemento section possui largura de 300px, e dentro dele colocamos um elemento div com 10% de largura, este será de 30px, ou seja, 10% da largura do elemento pai.

Vejamos um exemplo do uso das propriedades **width** e **height**:

- Arquivo: **cap3\_01.html**

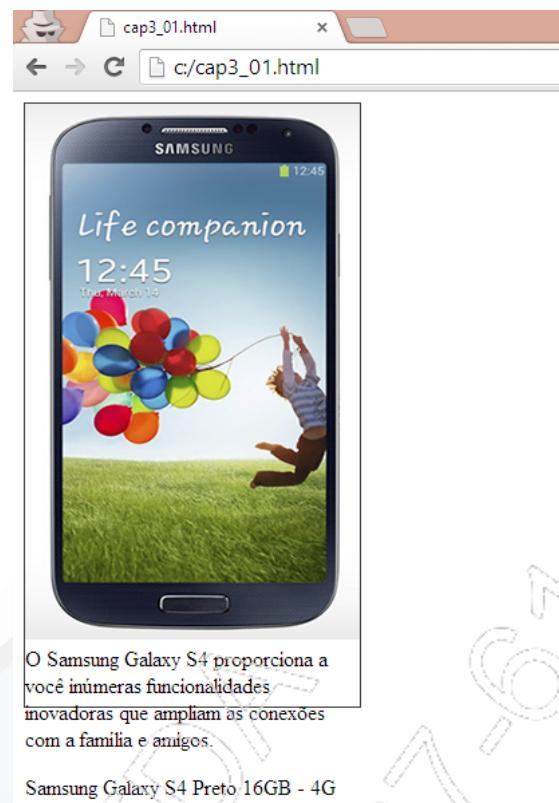
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_01.css">
7 </head>
8 <body>
9     <section class="bloco">
10        
11        <div class="descricao">
12            O Samsung Galaxy S4 proporciona a você inúmeras
13            funcionalidades inovadoras que ampliam as conexões
14            com a família e amigos.
15        </div>
16        <footer>
17            <p>Samsung Galaxy S4 Preto 16GB - 4G </p>
18        </footer>
19    </section>
20 </body>
21 </html>
```

- Arquivo: **estilo3\_01.css**

```
1 @charset "UTF-8";
2
3 .bloco{
4     width:250px;
5     height: 450px;
6     border:solid 1px #333;
7 }
```

# CSS3

Veja o resultado:



Observe que, após a definição da altura o texto que ultrapassa o limite fica exposto, o que seria facilmente contornado com a propriedade **overflow:hidden**.

```
1 @charset "UTF-8";
2
3 .bloco{
4     width:250px;
5     height: 450px;
6     border:solid 1px #333;
7     overflow: hidden;
8 }
```

Veja o resultado:



### 3.1.3. Configurando margem - Margin

Quando formatamos um elemento HTML, podemos determinar a quantidade de espaçamentos que este deverá possuir nas margens em volta dele. Para tal, utilizamos a propriedade **margin** com suas variações:

Propriedade	Valor inicial	Descrição
<b>margin-top</b>	0	Determina o espaçamento a partir da margem superior de um elemento.
<b>margin-left</b>	0	Determina o espaçamento a partir da margem esquerda de um elemento.
<b>margin-right</b>	0	Determina o espaçamento a partir da margem direita de um elemento.
<b>margin-bottom</b>	0	Determina o espaçamento a partir da margem inferior de um elemento.
<b>margin</b>	0	Forma abreviada de escrever todas as propriedades acima, que aceita de um a quatro valores, separados por espaços.

Para analisarmos o uso da propriedade **margin**, utilizaremos novamente o arquivo do exemplo anterior, porém renomeando-o para **cap3\_02.html**. Vamos adicionar um elemento **<main>** e uma **class container** a este elemento.

- Arquivo: **cap3\_02.html**

```

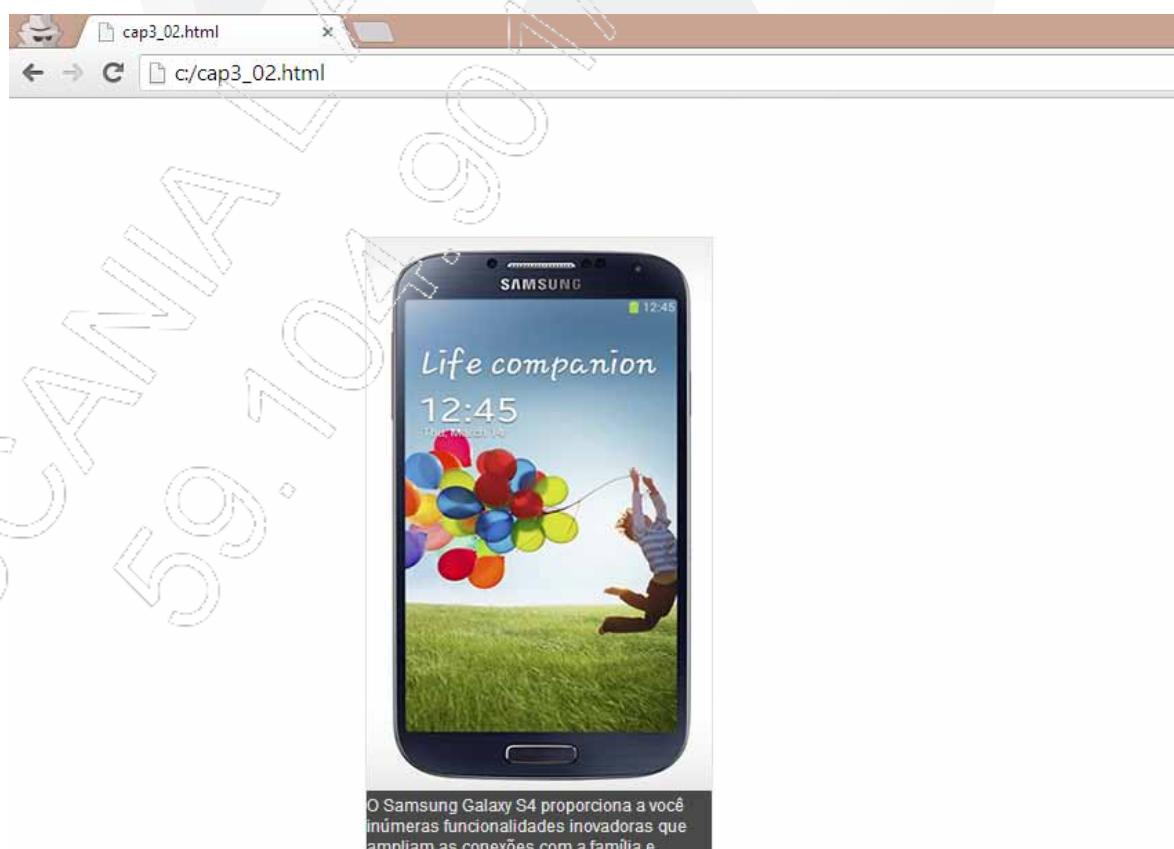
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title></title>
6      <link rel="stylesheet" type="text/css" href="css/estilo3_02.css">
7  </head>
8  <body>
9      <main role="main" class="container">
10         <section class="bloco">
11             
12             <div class="descricao">
13                 O Samsung Galaxy S4 proporciona a você inúmeras
14                 funcionalidades inovadoras que ampliam as conexões
15                 com a família e amigos.
16             </div>
17             <footer>
18                 <p>Samsung Galaxy S4 Preto 16GB - 4G </p>
19             </footer>
20         </section>
21     </main>
22 </body>
23 </html>

```

- Arquivo **estilo3\_02.css**:

```
1 @charset "UTF-8";
2
3 .container{
4     margin-left: 200px;
5     margin-top: 0px;
6 }
7
8 .bloco{
9     width:250px;
10    height: 450px;
11    border:solid 1px #e1e1e1;
12    overflow: hidden;
13    margin-top: 100px;
14    margin-left: 50px;
15    font-family:Arial;
16    font-size: 12px;
17    background-color: rgba(16,16,16,0.8);
18    color:#e1e1e1;
19 }
```

Após utilizar a propriedade **margin** para as classes **container** e **bloco**, veja o resultado:



### 3.1.3.1. Removendo a margem padrão das páginas

Quando desenvolvemos sites e web apps para diversos tipos de navegadores, é comum que todos eles tenham uma margem padrão no documento HTML, que pode diferir de um navegador para outro. Essa margem padrão é inserida para que o conteúdo do documento tenha um mínimo de legibilidade, mesmo que não seja carregada nenhuma folha de estilo. Para remover esta margem, que pode interferir na formatação dos elementos, basta criar um seletor para o elemento **body**, eliminando margens e preenchimentos:

```

1 @charset "UTF-8";
2
3 body{
4     margin: 0;
5     padding: 0;
6     color: #333;
7 }
```

### 3.1.4. Configurando espaçamento - Padding

Quando formatamos um elemento HTML, podemos determinar a quantidade de espaçamentos internos que este deverá possuir em relação ao seu conteúdo. Para isso, utilizamos a propriedade **padding** com suas variações:

Propriedade	Valor inicial	Descrição
<b>padding-top</b>	0	Determina o espaçamento interno do elemento a partir da margem superior deste.
<b>padding-left</b>	0	Determina o espaçamento interno do elemento a partir da margem esquerda deste.
<b>padding-right</b>	0	Determina o espaçamento interno do elemento a partir da margem direita deste.
<b>padding-bottom</b>	0	Determina o espaçamento interno do elemento a partir da margem inferior deste.
<b>padding</b>	0	Forma abreviada de escrever todas as propriedades acima, que aceita de um a quatro valores, separados por espaços.

Vejamos o uso da propriedade **padding** para adicionar espaçamento nas margens internas do elemento.

# CSS3

---

No exemplo anterior do arquivo **cap3\_02.html**, observe como o texto ficou sem a propriedade **padding**:



Agora vamos adicionar a propriedade **padding**, e também veremos mais um exemplo de uso da propriedade **margin-top**.

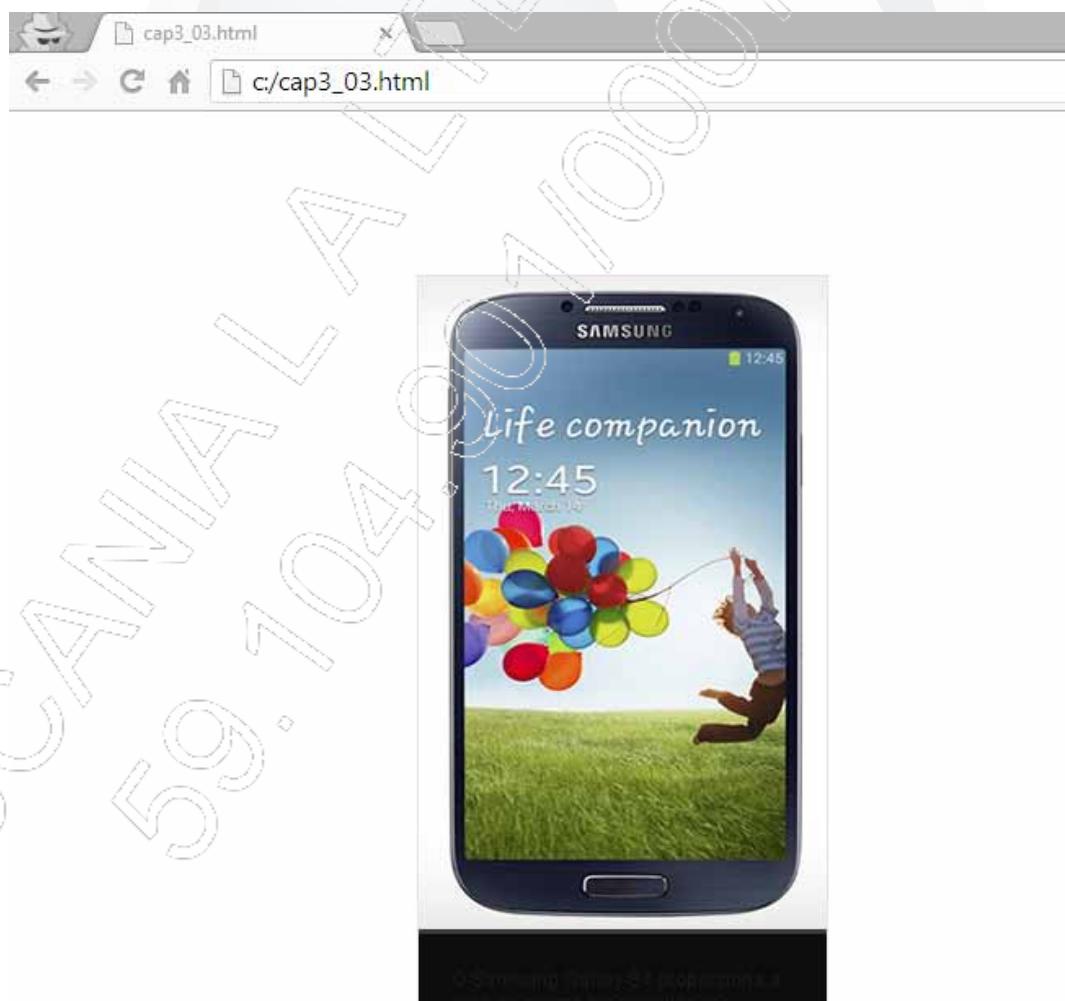
```
13 .bloco{  
14     width: 250px;  
15     height: 450px;  
16     border: solid 1px #e1e1e1;  
17     overflow: hidden;  
18     margin-top: 100px;  
19     margin-left: 50px;  
20     font-family: Arial;  
21     font-size: 12px;  
22     background-color: rgba(0,0,0,0.8);  
23     color: #e1e1e1;  
24 }  
25  
26 .descricao{  
27     padding:20px;  
28     position:relative;  
29     height: 200px;  
30     background-color: rgba(0,0,0,0.8);  
31     color:#161616;  
32     transition-duration:1s;  
33 }  
34 .descricao:hover{  
35     margin-top:-100px;  
36     color: #fff;  
37     background-color: rgba(0,0,0,0.7);  
38     cursor:pointer;  
39     transition-duration:1s;  
40 };
```

Observe que na classe descrição adicionamos uma propriedade **padding** com 20px, e quando o usuário passar o mouse o elemento sobe 100px para cima. Isso ocorre devido à inserção da propriedade **margin-top:-100px**, que significa 100px a menos da posição atual. As outras propriedades serão explicadas a seguir e em capítulos posteriores.

Veja o resultado:

- Arquivo: [cap3\\_03.html](#)

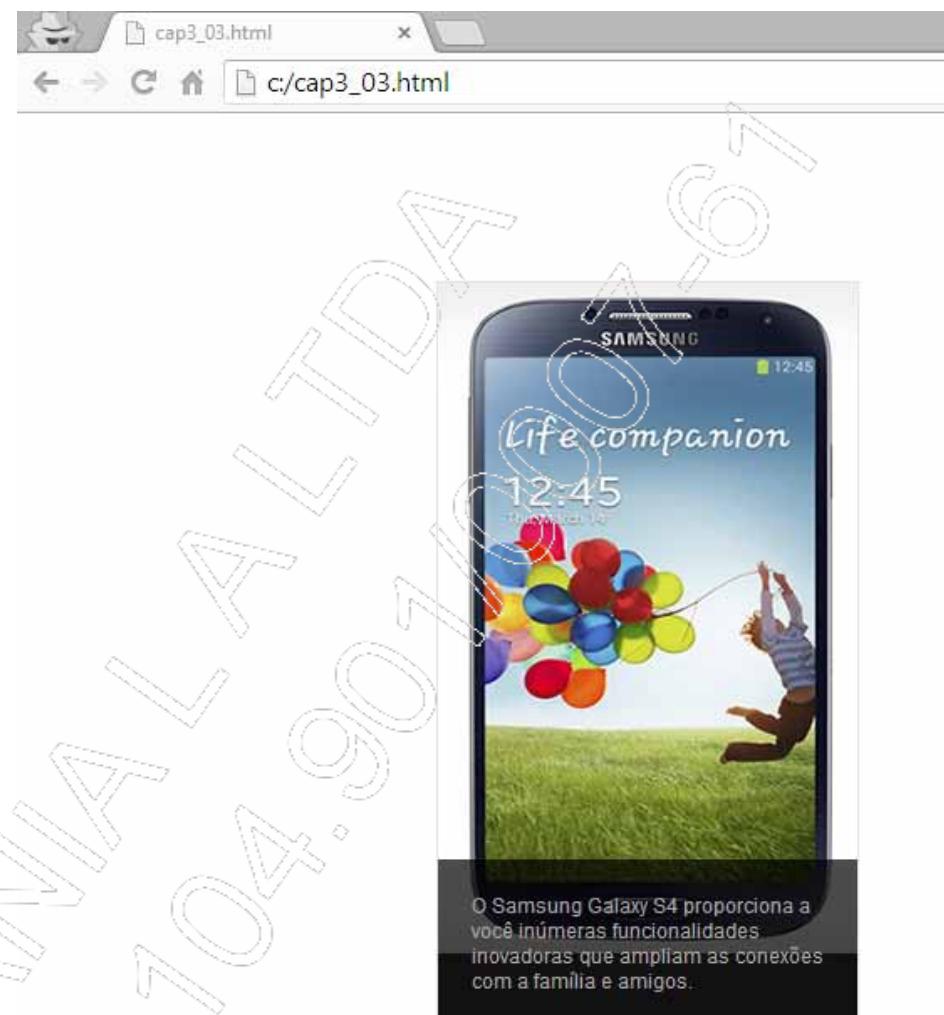
Estado inicial:



# CSS3

---

Agora o mesmo arquivo ao passar o mouse:



A propriedade **transition-duration:1s** permite criar o efeito deslizante, sobre o qual aprenderemos mais no capítulo sobre transições em CSS3.

## 3.2. Configurando posicionamento - Position

Uma das formas que podemos utilizar para posicionar elementos corretamente com CSS, é a propriedade **position**, que possui os seguintes valores: **inherit**, **static**, **fixed**, **relative**, **absolute**. Este é um recurso útil quando precisamos determinar a posição de elementos específicos de um site, mas para a criação do layout é recomendável o uso da propriedade **float**, que veremos adiante.

Além da propriedade position, podemos utilizar as propriedades **top**, **left**, **right** e **bottom**, além de **z-index** para posicionar os elementos de acordo com a necessidade do site.

### 3.2.1. Posicionamento inherit

Quando o elemento position possui o valor **inherit**, significa que este está herdando o mesmo valor de position determinado pelo seu elemento pai. Logo, quando mudamos o position de um elemento, caso elementos dentro deste possuam o valor **inherit**, serão afetados pela mesma formatação.

### 3.2.2. Posicionamento estático

Quando determinamos um valor **static** para uma propriedade position de um elemento, o mesmo irá seguir a posição padrão do elemento, conforme sua criação no documento. Vale lembrar que neste caso o elemento não aceitará a formatação de **top**, **left**, **right**, **bottom**.

# CSS3

---

Vejamos um exemplo:

- Arquivo: **estilo3\_04.css**

```
1 @charset "UTF-8";
2 body{
3     margin: 0;
4     padding: 0;
5     color: #333;
6 }
7 header{
8     position:static;
9     font-family: arial;
10    height: 60px;
11    width: 100%;
12    background-color: #333;
13    color:#fff;float:left;
14 }
15 div{float:left; font-size: 18px;
16     font-style: italic;
17     padding:10px;
18 }
19 .busca{
20     position: static;
21     width:600px;
22     height: 20px;float: left;
23     top:10px;border: none;
24     border-radius: 10px;
25     background-color: #888;
26 }
27 .container{
28     width: 200px;height: 600px;
29 }
```

Observe na linha 20 do elemento que na classe busca o **position** foi determinado com o valor **static**, o que inutiliza a propriedade **top:10px**. Isso ocorre devido ao fato do valor **static** determinar a ordem padrão do elemento na marcação HTML.

- Arquivo: cap3\_04.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_04.css">
7 </head>
8 <body>
9     <header>
10        <div id="logotipo">CSS3</div>
11        <div class="busca">
12            <input type="search" class="busca">
13        </div>
14    </header>
15    <main role="main" class="container">
16
17    </main>
18 </body>
19 </html>
```

Veja o resultado:



## 3.2.3. Posicionamento fixo

Quando determinamos o **position** com o valor **fixed**, o elemento permanece fixo na tela e não altera a sua posição quando a barra de rolagem é movimentada. Este é um recurso útil para menus fixos no topo ou rodapé fixo, bem como para conteúdo lateral de um site.

Vejamos um exemplo da aplicação do valor **fixed**:

- Arquivo: cap3\_05.css

```
1 @charset "UTF-8";
2 body{
3     margin: 0;
4     padding: 0;
5     color: #333;
6 }
7 header{
8     position:fixed;
9     font-family: arial;
10    top:0;
11    height: 60px;
12    width: 100%;
13    background-color: #333;
14    color:#fff;float:left;
15 }
16 div{float:left; font-size: 18px;
17     font-style: italic;
18     padding:10px;
19 }
20 .busca{
21     position: static;
22     width:600px;
23     height: 20px;float: left;
24     top:10px;border: none;
25     border-radius: 10px;
26     background-color: #888;
27 }
28 .container{
29     width: 700px;height: 600px;
30 }
```

Observe na linha 8 do exemplo anterior, quando o elemento **header** recebe **fixed** como valor da propriedade **position**, fazendo com que o cabeçalho fique fixo na página.

- Arquivo: cap3\_05.html

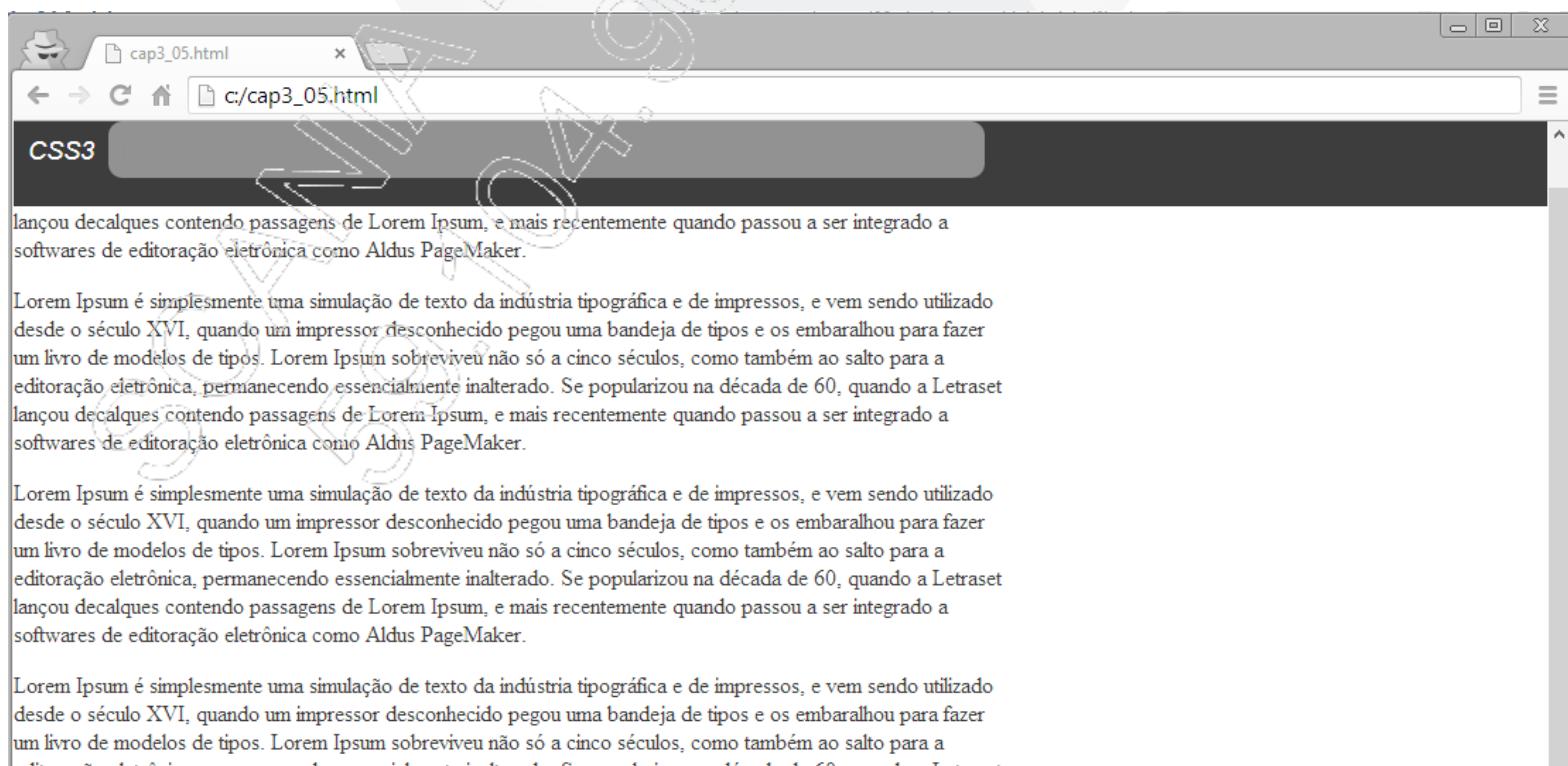
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_05.css">
7 </head>
8 <body>
9     <header>
10        <div id="logotipo">CSS3</div>
11        <div class="busca">
12            <input type="search" class="busca">
13        </div>
14    </header>
15    <main role="main" class="container">
16        <p>Lorem Ipsum é simplesmente uma simulação de texto da indústria
17 tipográfica e de impressos, e vem sendo utilizado desde o século
18 XVI, quando um impressor desconhecido pegou uma bandeja de tipos
19 e os embaralhou para fazer um livro de modelos de tipos. Lorem
20 Ipsum sobreviveu não só a cinco séculos, como também ao salto para
21 a editoração eletrônica, permanecendo essencialmente inalterado.
22 Se popularizou na década de 60, quando a Letraset lançou decalques
23 contendo passagens de Lorem Ipsum, e mais recentemente quando passou
24 a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.
25 </p>
```

```
26        <p>Lorem Ipsum é simplesmente uma simulação de texto da indústria
27 tipográfica e de impressos, e vem sendo utilizado desde o século
28 XVI, quando um impressor desconhecido pegou uma bandeja de tipos
29 e os embaralhou para fazer um livro de modelos de tipos. Lorem
30 Ipsum sobreviveu não só a cinco séculos, como também ao salto para
31 a editoração eletrônica, permanecendo essencialmente inalterado.
32 Se popularizou na década de 60, quando a Letraset lançou decalques
33 contendo passagens de Lorem Ipsum, e mais recentemente quando passou
34 a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.
35        <p>Lorem Ipsum é simplesmente uma simulação de texto da indústria
36 tipográfica e de impressos, e vem sendo utilizado desde o século
37 XVI, quando um impressor desconhecido pegou uma bandeja de tipos
38 e os embaralhou para fazer um livro de modelos de tipos. Lorem
39 Ipsum sobreviveu não só a cinco séculos, como também ao salto para
40 a editoração eletrônica, permanecendo essencialmente inalterado.
41 Se popularizou na década de 60, quando a Letraset lançou decalques
42 contendo passagens de Lorem Ipsum, e mais recentemente quando passou
43 a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.
44 </p>
```

```
45 |     |     |<p>Lorem Ipsum é simplesmente uma simulação de texto da indústria  
46 |     |     |tipográfica e de impressos, e vem sendo utilizado desde o século  
47 |     |     |XVI, quando um impressor desconhecido pegou uma bandeja de tipos  
48 |     |     |e os embaralhou para fazer um livro de modelos de tipos. Lorem  
49 |     |     |Ipsum sobreviveu não só a cinco séculos, como também ao salto para  
50 |     |     |a editoração eletrônica, permanecendo essencialmente inalterado.  
51 |     |     |Se popularizou na década de 60, quando a Letraset lançou decalques  
52 |     |     |contendo passagens de Lorem Ipsum, e mais recentemente quando passou  
53 |     |     |a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.  
54 |     |<p>Lorem Ipsum é simplesmente uma simulação de texto da indústria  
55 |     |tipográfica e de impressos, e vem sendo utilizado desde o século  
56 |     |XVI, quando um impressor desconhecido pegou uma bandeja de tipos  
57 |     |e os embaralhou para fazer um livro de modelos de tipos. Lorem  
58 |     |Ipsum sobreviveu não só a cinco séculos, como também ao salto para  
59 |     |a editoração eletrônica, permanecendo essencialmente inalterado.  
60 |     |Se popularizou na década de 60, quando a Letraset lançou decalques  
61 |     |contendo passagens de Lorem Ipsum, e mais recentemente quando passou  
62 |     |a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.  
63 |</p>  
64 |</main>  
65 |</body>  
66 |</html>
```

Observe que duplicamos vários parágrafos para gerar uma barra de rolagem com conteúdo em tela.

Vejamos o resultado mesmo descendo a barra de rolagem:



### 3.2.4. Posicionamento relativo

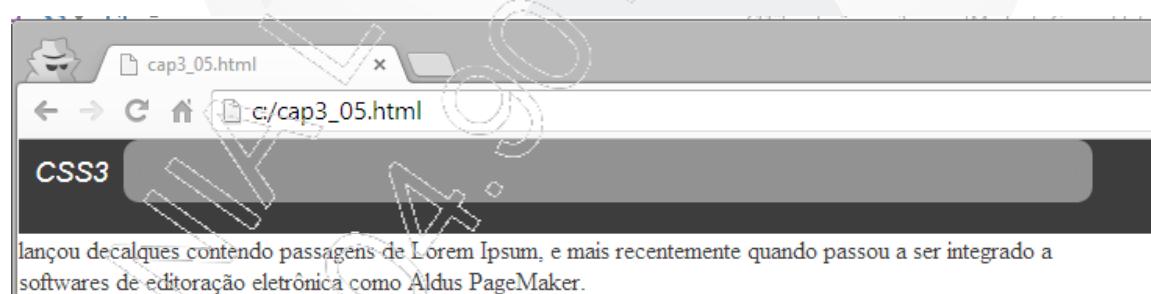
Quando determinamos o posicionamento relativo ao elemento, ele começará a ser contado a partir da posição atual do elemento no documento. No entanto, este poderá ser afetado pelas propriedades **top**, **left**, **right**, **bottom**.

Vejamos como exemplo a classe **busca** do exemplo anterior: a propriedade **position** estava com o valor **static**, o que anulava a propriedade **top:10px**. Entretanto, quando mudamos o valor da propriedade **position** para **relative**, este passa a aceitar a formatação **top**.

- Arquivo: cap3\_05.css

```
20 .busca{  
21     position: static;  
22     width:600px;  
23     height: 20px;float: left;  
24     top:10px;border: none;  
25     border-radius: 10px;  
26     background-color: #888;  
27 }
```

Resultado:

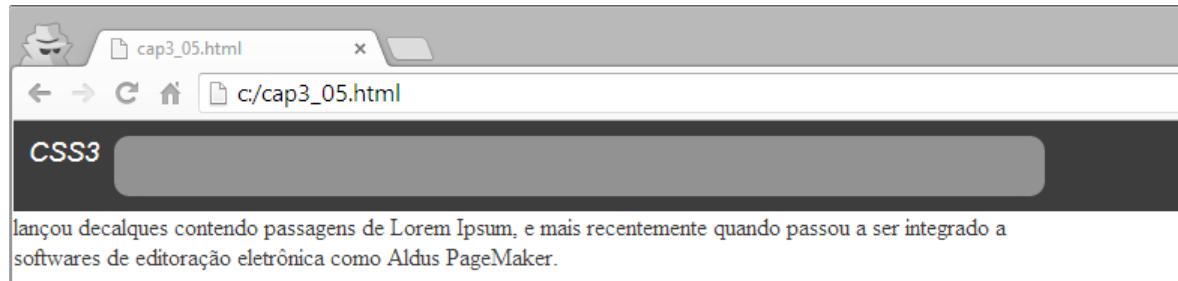


Mudando a propriedade para **relative**:

- Arquivo: cap3\_05.css

```
20 .busca{  
21     position: relative;  
22     width:600px;  
23     height: 20px;float: left;  
24     top:10px;border: none;  
25     border-radius: 10px;  
26     background-color: #888;  
27 }
```

Veja a mudança no campo com a classe busca:



## 3.2.5. Posicionamento absoluto

Neste tipo de posicionamento é possível determinar a posição absoluta do elemento no documento HTML. Podemos utilizar, por exemplo, tal elemento em uma posição específica em relação as margens do documento, fazendo com que assim este possua um posicionamento diferente da ordem em que foi escrito em sua marcação HTML.

- Arquivo: cap3\_06.css

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_06.css">
7   </head>
8   <body>
9     <header>
10       <div id="logotipo">CSS3</div>
11       <div class="busca">
12         <input type="search" class="busca">
13       </div>
14     </header>
15     <main role="main" class="container">
16       <div class="banner">
17         <h1>iPhone 5S</h1>
18         <p class="descricao">
19           <br/>
20           Rede 4G, mídias sociais de uma
21           forma como você nunca viu.
22         </p>
23         <div class="iphone"></div>
24       </div>
25     </main>
26   </body>
27 </html>
```

- Arquivo: cap3\_06.css

```
1 @charset "UTF-8";
2 body{
3     margin: 0; padding: 0; color: #333;
4 }
5 header{
6     position:fixed; font-family: arial;
7     top:0; height: 60px; width: 100%;
8     background-color: #333;
9     color:#fff; float:left;
10 }
11 div{float:left; font-size: 18px;
12     font-style: italic; padding:10px;
13 }
14 .busca{
15     position: relative;
16     width:600px; height: 20px; float: left;
17     top:10px; border: none;
18     border-radius: 10px;
19     background-color: #888;
20 }

21 .banner{
22     width:850px; height: 320px;
23     position: relative; left:100px; top:100px;
24     border:solid 1px #eee;
25     border-radius: 10px;
26 }
27 .iphone{
28     width: 390px; height:200px;
29     background: url('../images/iphone.png') no-repeat;
30     position: absolute; right:0; bottom:0;
31 }
32 h1{
33     font-family: arial; font-style: normal;
34     position: relative; left:80px; top:50px;
35 }
36 .descricao{
37     width:200px; position: relative;
38     left:90px; top:30px; font-family: arial;
39     font-size: 20px;
40 }
```

## CSS3

---

Observe na linha 30 do exemplo anterior que, com a propriedade **position:absolute**, colocamos a propriedade **right**, determinando para que o elemento se alinhasse à direita, e fixo na base por meio da propriedade **bottom**.

Veja o resultado:



### 3.3. Elementos flutuantes para layout

Quando cada elemento do documento HTML é tratado como uma caixa ou um box, uma das principais formas de posicionar este elemento no documento é por meio da propriedade **float**, que possui três valores:

Valor	Descrição
<b>left</b>	Posiciona o elemento à esquerda enquanto existir espaço, do contrário jogará o próximo elemento para baixo.
<b>right</b>	Posiciona o elemento à direita enquanto existir espaço, do contrário jogará o próximo elemento para baixo.
<b>none</b>	O elemento não flutua, sendo este o valor padrão.

Vejamos um exemplo da propriedade **float**.

- Arquivo: cap3\_07.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_07.css">
7   </head>
8   <body>
9     <header>
10       <div id="logotipo">CSS3</div>
11       <div id="busca-topo">
12         <input type="search" class="busca">
13       </div>
14       <nav>
15         Produtos | Artigos | Promoções | Sobre Nós
16       </nav>
17     </header>
18   </body>
19 </html>
```

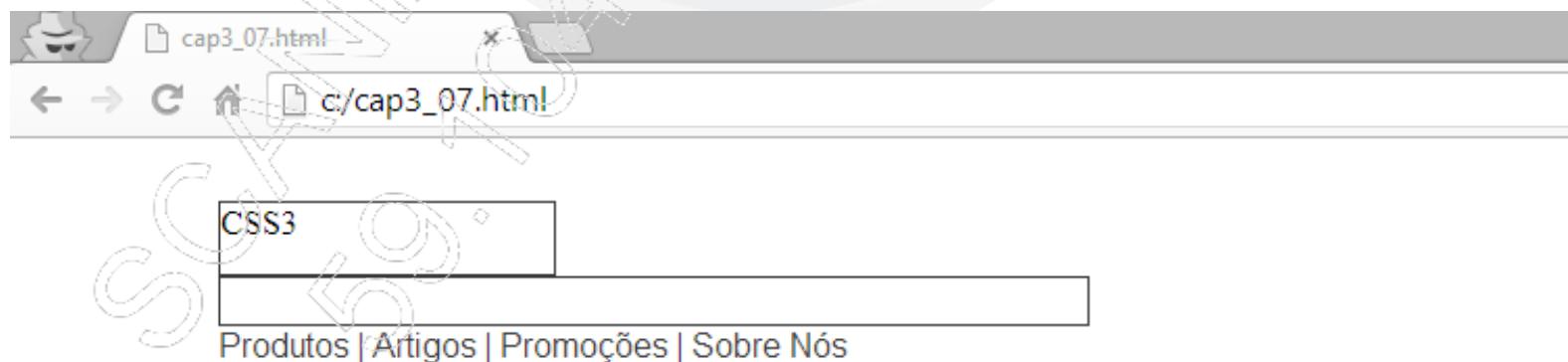
# CSS3

---

- Arquivo: **estilo3\_07.css**

```
1 @charset "UTF-8";
2
3 body{
4     margin: 0px; padding: 0px;
5 }
6 header{
7     margin-left: 100px; margin-top: 30px;
8 }
9 #logotipo{
10    width:160px; height: 34px;
11    border: solid 1px #333;
12 }
13 #busca-topo{
14    width:500px; height: 24px;
15 }
16 .busca{
17    width:418px; height: 24px;
18    border: solid 1px #333;
19 }
20 nav{
21     font-family: arial, sans-serif;
22     color:#333;
23 }
```

Observe no exemplo anterior que não estamos utilizando a propriedade **float**. Veja o resultado:



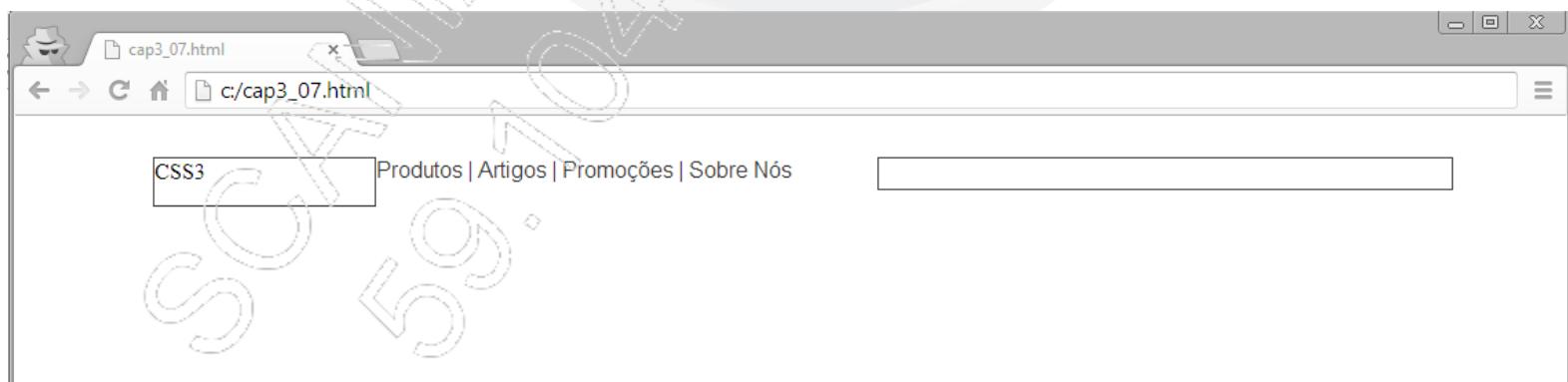
Agora vamos alterar o arquivo **estilo3\_07.css** adicionando a propriedade **float** em algumas declarações.

- Arquivo: **estilo3\_07.css**

```
1 @charset "UTF-8";
2 body{
3     margin: 0px; padding: 0px;
4 }
5 header{
6     margin-left: 100px; margin-top: 30px;
7 }
8 #logotipo{
9     width:160px; height: 34px;
10    position: relative;
11    float: left;
12    border: solid 1px #333;
13 }
14 #busca-topo{
15    width:500px; height: 24px;
16    position: relative;
17    float: right;
18 }
19 .busca{
20    width:418px; height: 24px;
21    border: solid 1px #333;
22 }
23 nav{
24    font-family: arial, san-serif;
25    color:#333;
26 }
```

Adicionamos a declaração **float:left** para o id **#logotipo** e a declaração **float:right** para o id **#busca-topo**.

E temos o seguinte resultado:



Porém, temos um problema: observe que embora o menu **nav** tenha sido declarado após os itens **logotipo** e **busca**, o mesmo apareceu posicionado entre estes elementos. Isso ocorre por que há um espaço em que este elemento pode se encaixar e o mesmo assim o faz. Supondo que este não é o comportamento desejado, como fazemos para o menu ficar abaixo dos elementos de logotipo e busca? É necessário colocar a declaração **clear:both** no menu de navegação.

## 3.3.1. Clear

A propriedade **clear** associada ao valor **both** permite que o elemento que a recebe seja forçado a ir para uma nova linha abaixo de um elemento flutuante. Com isso, resolvemos o problema do exemplo anterior.

```
23 nav{  
24     position: relative;  
25     top: 20px;  
26     font-family: arial, sans-serif;  
27     color: #333;  
28     clear: both;  
29 }
```

Agora vejamos o resultado corretamente:



## 3.3.2. Trabalhando camadas com z-index

Em um documento HTML, a ordem em que os elementos são escritos podem interferir diretamente no resultado final do layout renderizado na tela do usuário. Assim, às vezes dois elementos que ocupam áreas próximas ou sobreescritas podem não ter o resultado desejado devido à ordem em que foram escritos.

Podemos então determinar a ordem de exibição dos elementos no eixo de profundidade. Isso é feito pelo **z-index**, que é o índice do vértice z que identifica profundidade, como o nome sugere. Logo, um elemento que deve aparecer ao fundo dos outros elementos, deverá ter o **z-index zero**, e o elemento que deverá aparecer à sua frente, **z-index:1**, e assim por diante.

Vejamos um exemplo:

- Arquivo: cap3\_08.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_08.css">
7   </head>
8   <body>
9     <header>
10       <div id="logotipo">CSS3</div>
11       <div id="busca-topo">
12         <input type="search" class="busca">
13       </div>
14       <nav>
15         | Produtos | Artigos | Promoções | Sobre Nós
16       </nav>
17     </header>
18     <div id="banner-full">
19       <div class="descricao">
20         <p>iPhone 4S</p>
21         <p>Novas experiências</p>
22         <p>Conheça o modelo ideal para você</p>
23       </div>
24       <div class="foto-produto">
25         
26         
27       </div>
28     </div>
29   </body>
30 </html>
```

# CSS3

---

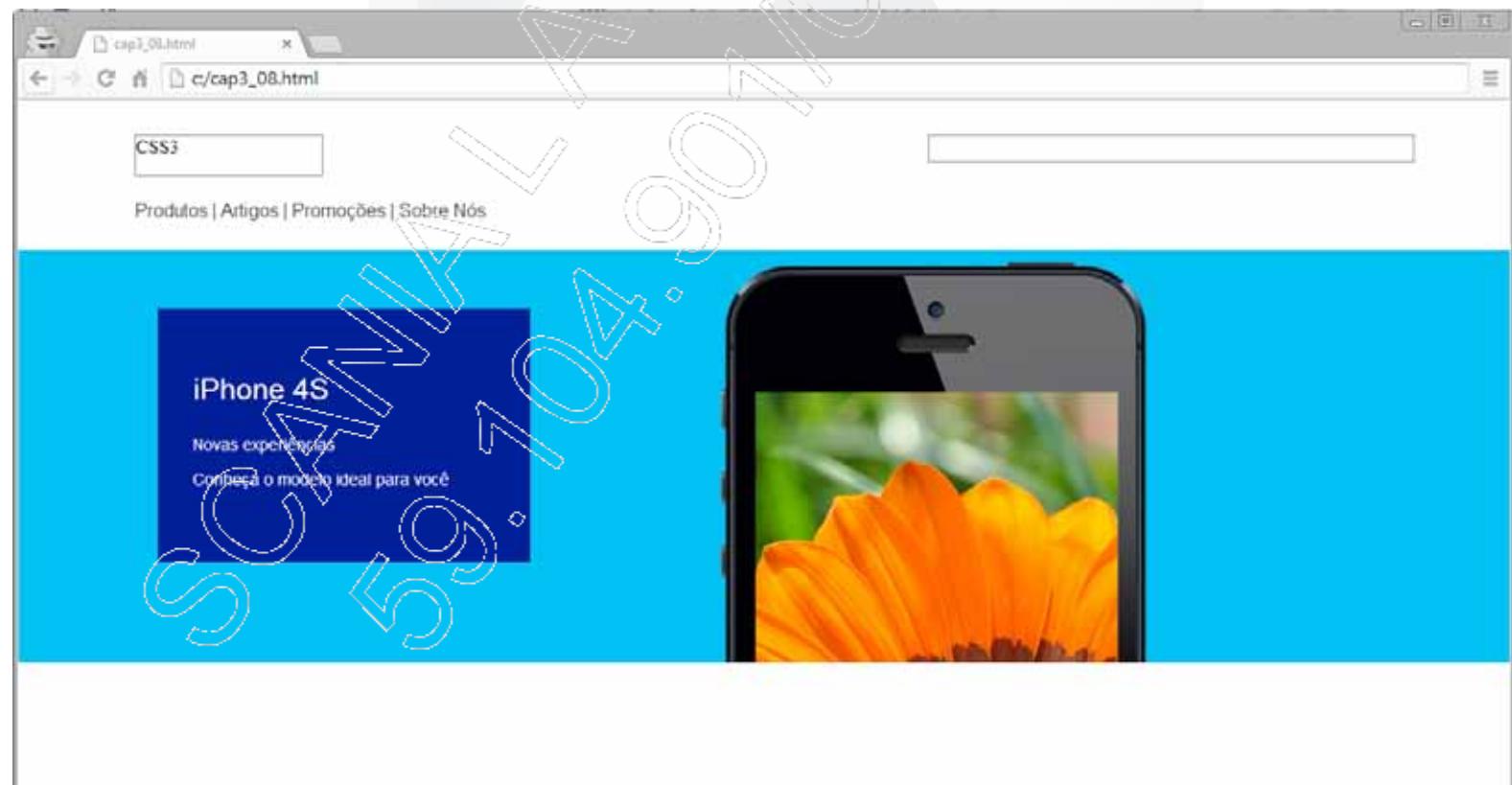
- Arquivo: **estilo3\_08.css**

```
1 @charset "UTF-8";
2 body{
3     margin: 0px; padding: 0px;
4 }
5 header{
6     margin-left: 100px; margin-top: 30px;
7     height:100px;
8 }
9 #logotipo{
10    width:160px;height: 34px;
11    position:relative;
12    float:left;
13    border: solid 1px #888;
14 }
15 #busca-topo{
16    width:500px;height: 24px;
17    position:relative;
18    float:right;
19 }
20 .busca{
21    width:418px;height: 24px;
22    border: solid 1px #888;
23 }
```

```
24 nav{
25     position:relative;
26     top: 20px;
27     font-family: arial, sans-serif;
28     color:#333;
29     clear: both;
30 }
31
32 #banner-full{
33     width: 100%;
34     height: 355px;
35     position: relative;
36     background-color: #01BCF3;
37     overflow: hidden;
38 }
```

```
39 .descricao{  
40     width: 260px;  
41     height: 158px;  
42     background-color: #00188F;  
43     position: absolute;  
44     left: 120px; top: 50px;  
45     font-family: arial;  
46     color:#fff;  
47     font-size: 14px;  
48     padding: 30px;  
49 }  
50 .descricao p:first-child{  
51     font-size: 25px;  
52 }  
53 #iphone{  
54     width:377px;  
55     position: absolute;  
56     left:600px; top: 10px;  
57 }  
58 #impacta-site{  
59     position: absolute;  
60     left: 634px;  
61     top: 120px;  
62 }
```

Vejamos o resultado:



Se alterarmos a declaração do seletor **#impacta-site** definindo um **z-index** como 1 e deixando o **z-index** do seletor **#iphone** como 0, temos o seguinte resultado:

# CSS3

---

- Arquivo: **estilo3\_08.css**

```
53 #iphone{  
54     width:377px;  
55     position: absolute;  
56     left:600px; top: 10px;  
57     z-index: 0;  
58 }  
59 #impacta-site{  
60     position: absolute;  
61     left: 634px;  
62     top: 120px;  
63     z-index: 1;  
64 }
```

Resultado:



## 3.4. Flexible Box Layout

Um dos novos recursos da CSS3 mais empolgantes é chamado de Módulo de Layout Flexível, ou simplesmente Flex Layout.

Em 2013 este módulo ainda estava no nível de candidato a recomendação da W3C, e é um item muito importante pois facilita imensamente a criação e posicionamento de um layout, permitindo a criação de colunas mais facilmente, com igualdade de dimensão. O Flexible Box Layout, ou simplesmente FlexBox Layout, não utiliza mais as contas que temos que fazer ao utilizar o box model.

Para trabalhar com este recurso, é necessário criar uma área onde este módulo irá atuar. Pode ser uma **div**, uma **section** ou qualquer elemento que receberá itens filhos. Neste contexto, utilizamos a propriedade **display**, com o valor **flex** ou **inline-flex** para determinar que esta área é um Flex Container.

Fazemos isso com a seguinte declaração:

```
elemento{  
  display:flex;  
}
```

Ou se for um elemento de linha como um **span**, por exemplo:

```
elemento{  
  display: inline-flex;  
}
```

Valor	Descrição
<b>flex</b>	O elemento se torna um flex container em nível de bloco.
<b>inline-flex</b>	O elemento se torna um flex container em nível de linha.

Vejamos o que é possível fazer com o novo módulo de layout flexível:

- Criar o Flex Container;
- Determinar a direção de linhas ou colunas dentro de um Flex Container;
- Determinar a ordem de exibição dos elementos;
- Ajustar o espaço em volta dos elementos.

Como esse módulo não consta como nativo em alguns navegadores, será necessário utilizar prefixos proprietários, como no IE10. Neste caso, em vez de colocar `display:flex`, utilize também `display:-ms-flex`.

## 3.4.1. Controlando o fluxo dentro de um container

Para determinarmos a ordem do fluxo de um container, temos algumas propriedades do módulo flexível que são úteis para essa tarefa.

Valor	Valor inicial	Descrição
<b>flex-direction</b>	row	Determina a direção do fluxo, que pode ser linha, coluna ou ainda ser em sentido reverso.
<b>flex-wrap</b>	nowrap	Pode controlar se o elemento utilizará múltiplas linhas ou múltiplas colunas.
<b>flex-flow</b>	row nowrap	Um atalho para a propriedade flex-direction e flex-wrap.

A propriedade **flex-direction** pode receber os valores a seguir:

Valor	Descrição
<b>row</b>	Determina que o container será formado por linhas.
<b>column</b>	Determina que o container será formado por colunas.
<b>row-reverse</b>	Determina que o container será formado por linhas na ordem reversa.
<b>column-reverse</b>	Determina que o container será formado por colunas na ordem reversa.

A propriedade **flex-wrap** pode receber os valores a seguir:

Valor	Descrição
<b>nowrap</b>	Determina que o container terá apenas uma linha ou uma coluna.
<b>wrap</b>	Determina que, quando a altura (no caso de coluna) ou a largura (no caso de linha) do container for atingida, será gerada automaticamente uma nova coluna ou linha, dependendo do fluxo determinado em flex-direction.
<b>wrap-reverse</b>	O container se torna multilinha, entretanto a ordem das colunas ou linhas geradas será o oposto do eixo transversal.

Vejamos alguns exemplos:

- Arquivo: **cap3\_09.html**

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="css/estilo3_09.css">
7   </head>
8   <body>
9     <section id="container">
10       <p> Item 1 </p>
11       <p> Item 2 </p>
12       <p> Item 3 </p>
13       <p> Item 4 </p>
14     </section>
15   </body>
16 </html>
```

# CSS3

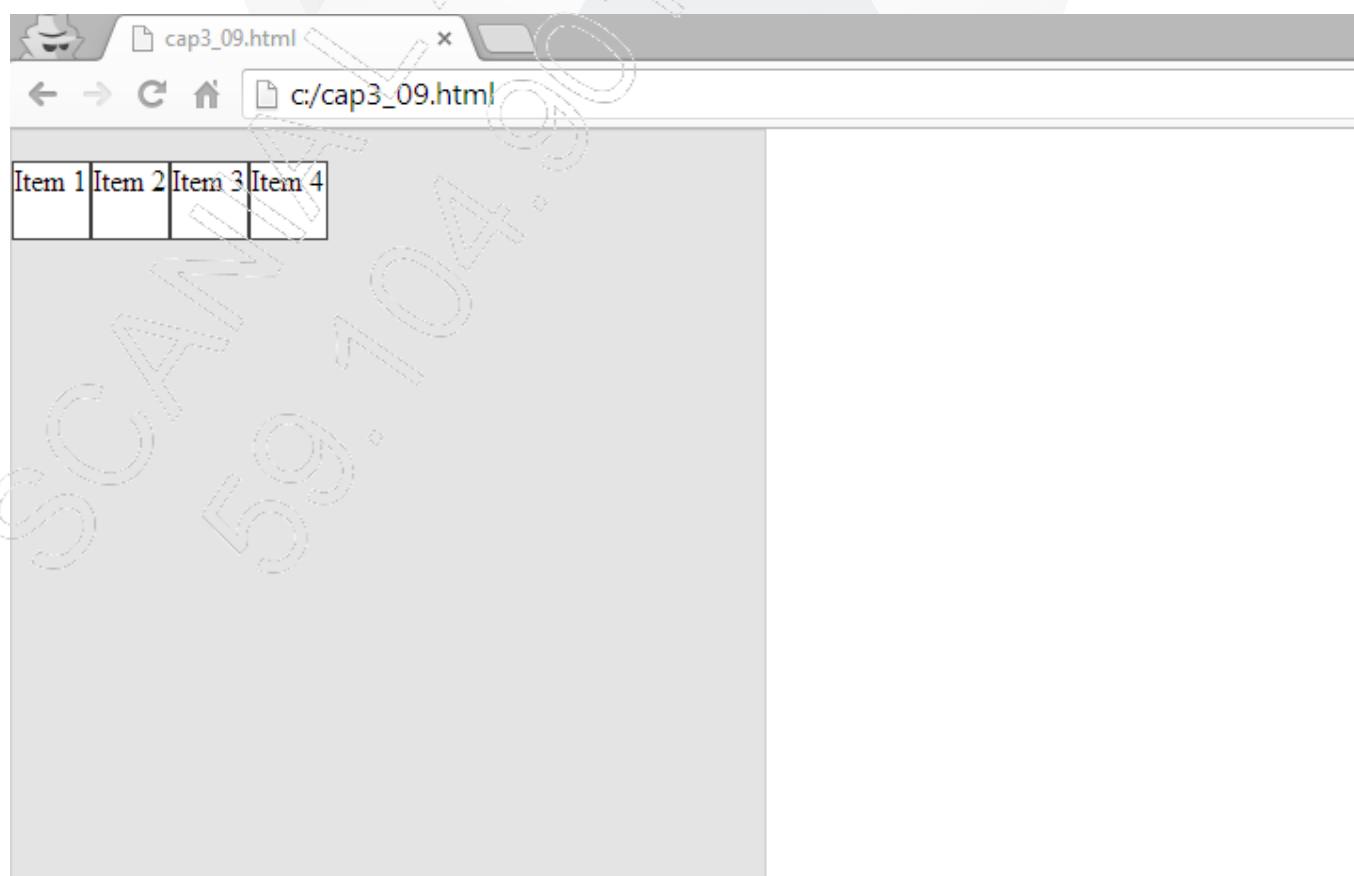
---

Vejamos a aplicação da propriedade **flex-direction:row**.

- Arquivo: **estilo3\_09.css**

```
1 @charset "UTF-8";
2 body{
3     margin: 0px; padding: 0px;
4 }
5
6 #container{
7     width: 400px;
8     height: 400px;
9     background-color: #e1e1e1;
10    border: solid 1px #ccc;
11    display: flex;
12    flex-direction:row;
13 }
14
15 p{
16     width:40px;
17     height: 40px;
18     border: solid 1px #333;
19     background-color: #fff;
20 }
```

O resultado obtido mostrará os elementos de parágrafo internos em linha.

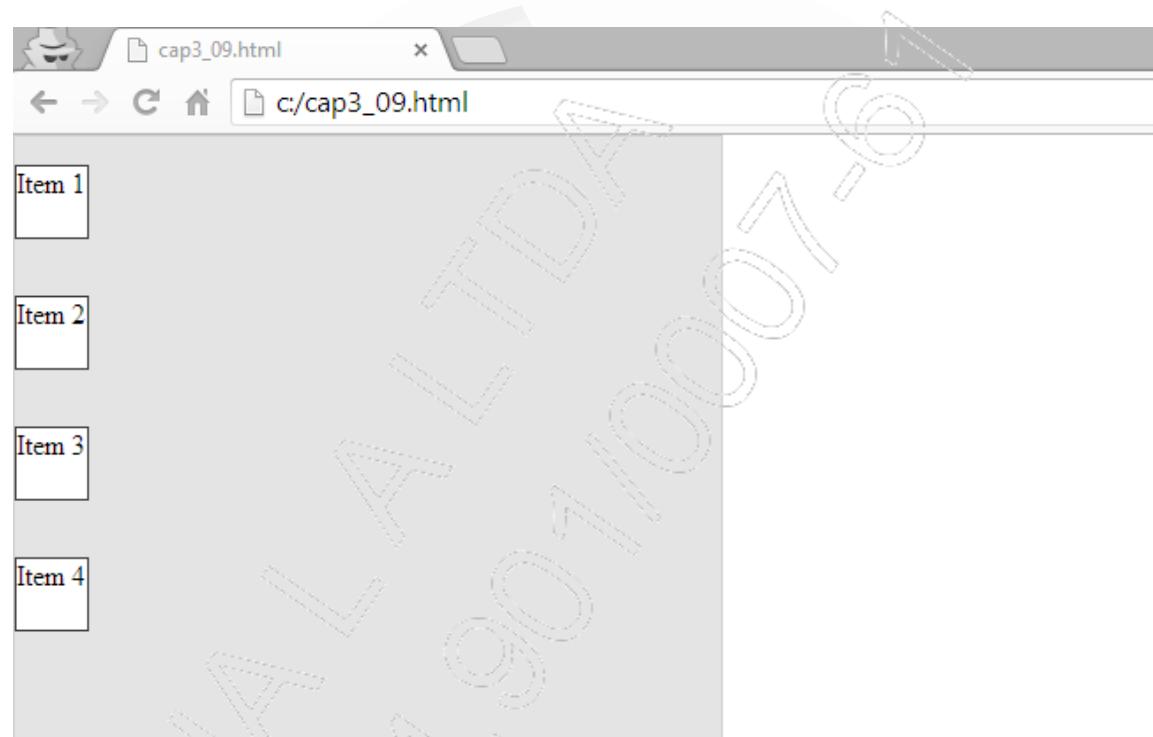


Vejamos a aplicação da propriedade **flex-direction:column**.

- Arquivo: **estilo3\_09.css**

```
6 #container{  
7     width: 400px;  
8     height: 400px;  
9     background-color: #e1e1e1;  
10    border: solid 1px #ccc;  
11    display: flex;  
12    flex-direction:column;  
13 }
```

Observe que não alteramos o HTML ou qualquer outro cálculo.



Observe na imagem a seguir o que acontece ao utilizarmos a propriedade **row-reverse**: a ordem é revertida, começando com o item 4 e terminando no item 1. Perceba também que os elementos são alinhados ao final do eixo principal.

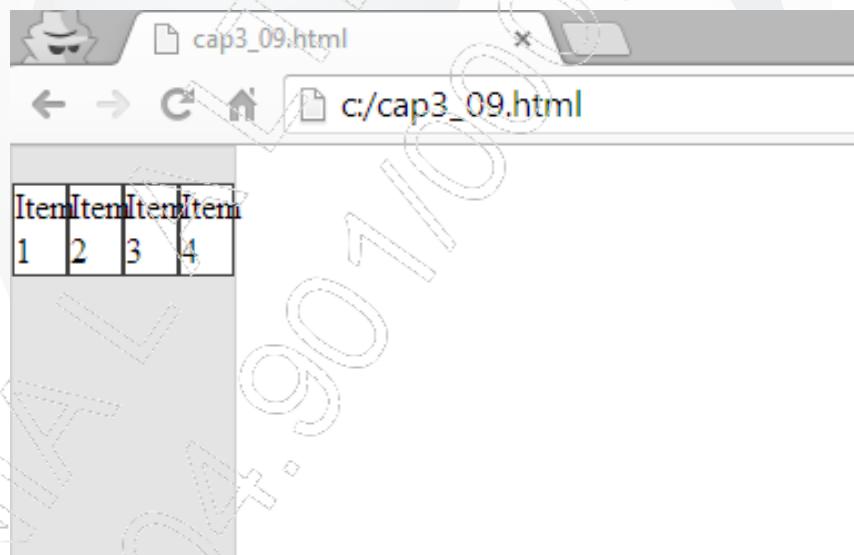


# CSS3

Agora vamos alterar a largura do container, de modo que os itens ficarão compactados.

```
6 #container{  
7     width: 100px;  
8     height: 400px;  
9     background-color: #e1e1e1;  
10    border: solid 1px #ccc;  
11    display: flex;  
12    flex-direction:row;  
13}  
14  
15 p{  
16     width:40px;  
17     height: 40px;  
18     border: solid 1px #333;  
19     background-color: #fff;  
20}
```

Observe que a largura do container ficou em 100px, o que representa um valor insuficiente para abrigar cada parágrafo de 40px, num total de 4 parágrafos. O seguinte resultado é exibido:



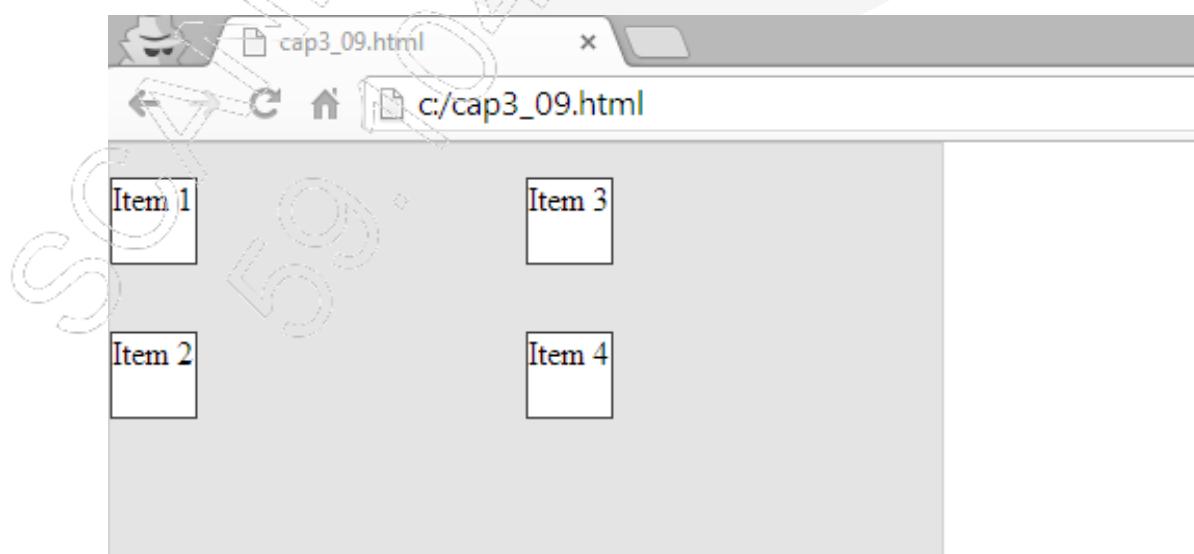
Entretanto, se adicionarmos a declaração **flex-wrap:wrap**, perceba que ao notar não ser possível posicionar o elemento na mesma linha, o próprio módulo flexível gera uma nova linha.

```
5  
6 #container{  
7     width: 100px;  
8     height: 400px;  
9     background-color: #e1e1e1;  
10    border: solid 1px #ccc;  
11    display: flex;  
12    flex-direction:row;  
13    flex-wrap:wrap;  
14}
```

Um resultado mais interessante então é exibido:



Do mesmo modo, se mudarmos a disposição para colunas e reduzirmos a altura, teremos a facilidade da geração de novas colunas:



Para alinharmos os itens podemos utilizar dentro do container as propriedades **align-items**, **justify-content**, **align-content**. Vejamos a diferença entre elas:

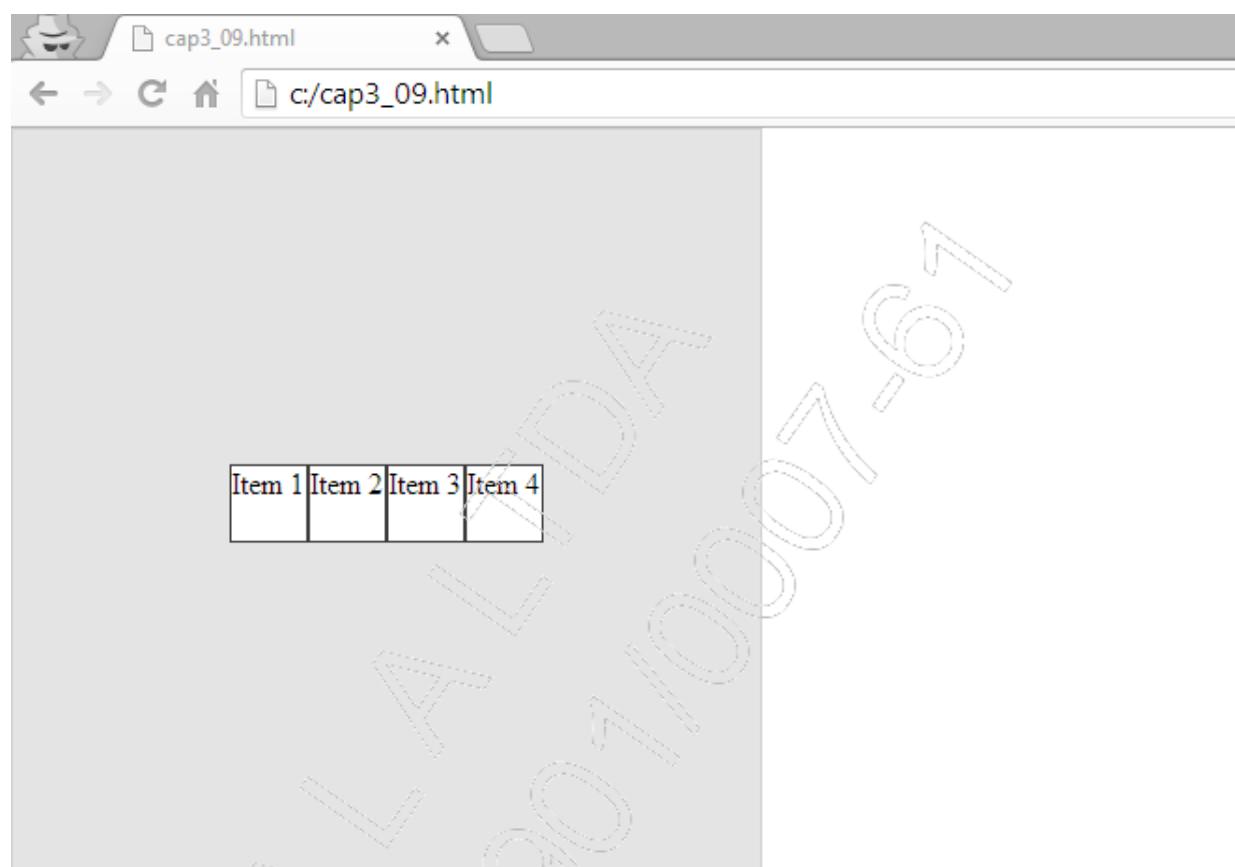
Propriedade	Descrição
<b>align-items</b>	Determina o alinhamento padrão dos flex-items no eixo transversal.
<b>justify-content</b>	Determina o alinhamento dos flex-items ao longo do eixo principal do flex container.
<b>align-content</b>	Determina o alinhamento dos flex-items no eixo transversal, no entanto deve ser utilizado para alinhamento em mais de uma linha ou mais de uma coluna, e a propriedade flex-wrap deve estar habilitada para wrap ou wrap-reverse.
<b>align-self</b>	Sobrescreve o alinhamento do flex-item ignorando o alinhamento determinado no flex container.

Em versões anteriores da CSS, era necessário fazer alguns cálculos para centralizar os itens na tela, mas agora com as propriedades **align-items** e **justify-content**, podemos centralizar facilmente os elementos na tela:

- Arquivo: **estilo3\_09.css**

```
6 #container{  
7     width: 400px;  
8     height: 400px;  
9     background-color: #e1e1e1;  
10    border: solid 1px #ccc;  
11    display: flex;  
12    flex-direction: row;  
13    flex-wrap: nowrap;  
14    align-items: center;  
15    justify-content:center;  
16 }
```

Vejamos o resultado:

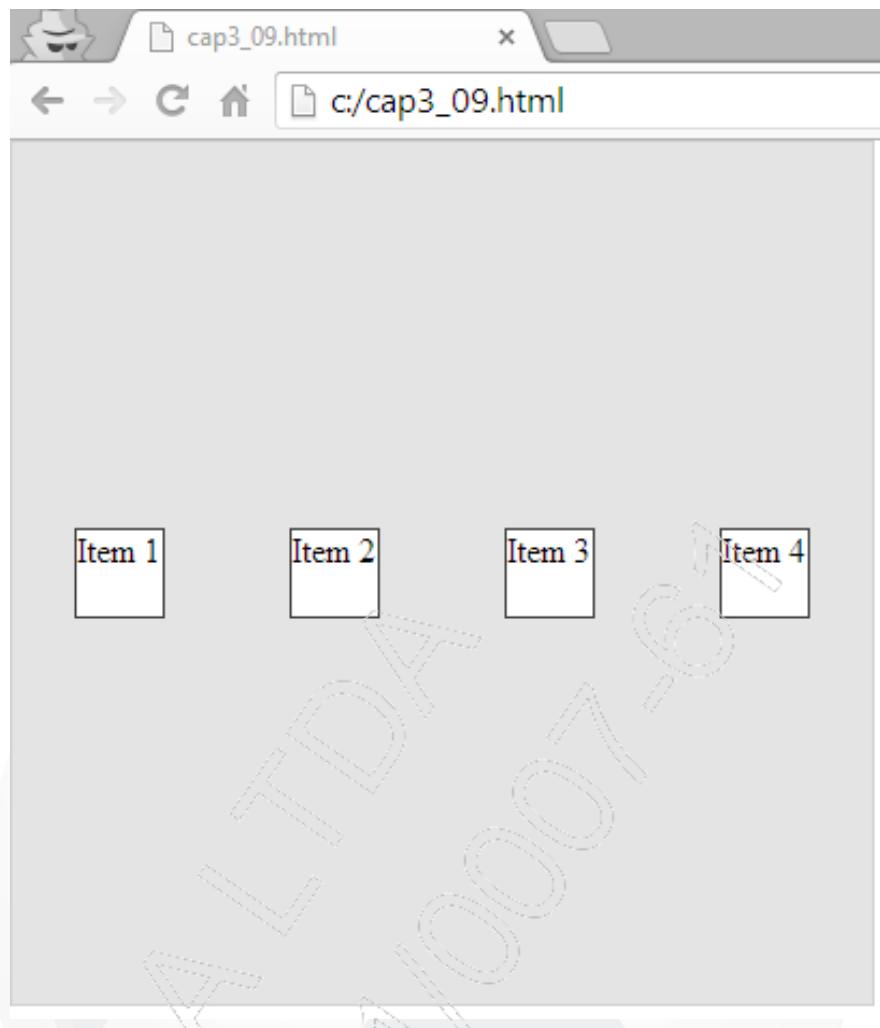


Caso seja necessário adicionar espaços em volta dos elementos, podemos utilizar o valor **space-around** ao invés de utilizarmos center:

```
6 #container{  
7   width: 400px;  
8   height: 400px;  
9   background-color: #e1e1e1;  
10  border: solid 1px #ccc;  
11  display: flex;  
12  flex-direction: row;  
13  flex-wrap: nowrap;  
14  align-items: center;  
15  justify-content:space-around;  
16 }
```

# CSS3

Veja o resultado:



Na tabela a seguir temos os valores possíveis para as propriedades de alinhamento do módulo flexível para layout.

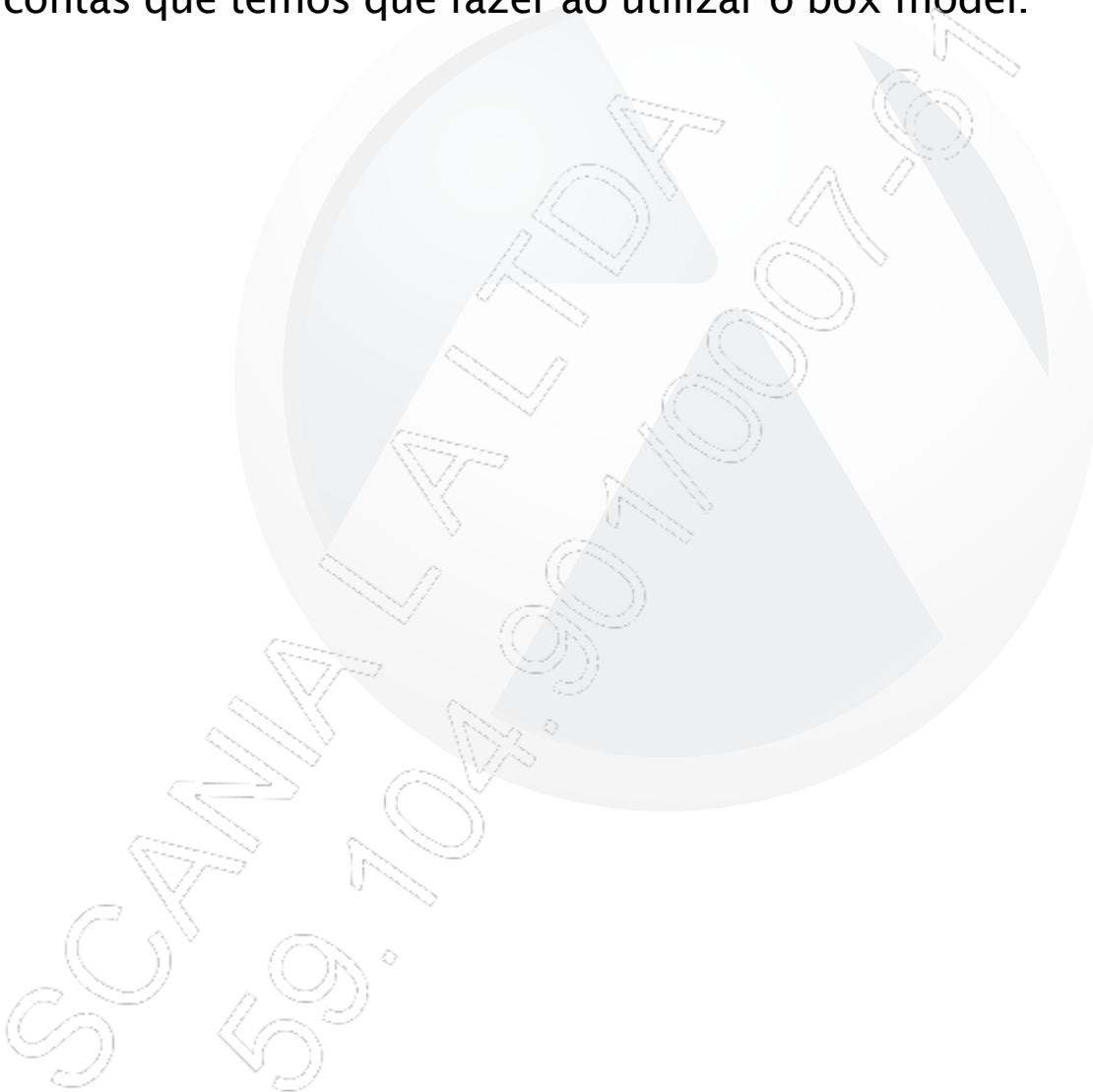
valor	Descrição
<b>flex-start</b>	O alinhamento dos flex-items se dará a partir do início do eixo principal, e os próximos elementos serão alinhados na sequência, sendo este o valor padrão de alinhamento.
<b>flex-end</b>	O alinhamento dos flex-items se dará a partir do fim do eixo principal, e os próximos elementos serão compactados e alinhados na mesma sequência.
<b>center</b>	O alinhamento dos flex-items se dará ao centro do eixo em questão, se for a propriedade align-items será em cima do eixo principal, e se for sobre a propriedade justify-content será sobre o eixo transversal.
<b>space-between</b>	Determina espaços entre os flex-items, utilizando todo o espaço do eixo em questão.
<b>space-around</b>	Determina espaços em volta dos flex-items, dividindo o espaço de maneira uniforme.

## Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- O módulo box model da CSS pode ser de dois tipos: box nível de bloco (block-level) ou box nível de linha (inline);
- **Margin** é o espaço que separa o box de outros elementos que compõem a página, porém não está dentro dos limites de um elemento, sendo utilizada com a finalidade de mover a caixa do elemento; **Conteúdo** são os itens contidos em um elemento, os quais permanecem dentro do container; **Border** é o limite de um elemento, isto é, são as linhas de contorno do box; **Padding** é o espaço em branco existente entre o conteúdo e os limites de um elemento;
- Para determinar a largura e altura mínima de um elemento, utilizamos as propriedades **min-width** e **min-height**, respectivamente;
- Quando o conteúdo de um elemento filho estoura o limite determinado pelo elemento superior, podemos ocultá-lo por meio da declaração **overflow:hidden**;
- Para posicionar um elemento em relação à margem esquerda do navegador ou de um elemento ancestral, utilizamos a propriedade **margin-left**;
- Para adicionarmos espaçamento interno em um elemento utilizamos a propriedade **padding**, e caso esse espaçamento seja em relação ao topo utilizamos **padding-top**;
- Para posicionar corretamente um elemento utilizamos a propriedade **position**;
- Quando o elemento position possui o valor **inherit**, significa que este está herdando o mesmo valor position determinado pelo seu elemento pai;
- Quando o elemento position possui o valor **absolute**, significa que este irá ser posicionado em relação ao eixo do navegador, não importando a ordem dos elementos no documento;

- Quando determinamos o **position** com o valor **fixed**, o mesmo fica fixo na tela e não altera a sua posição quando a barra de rolagem é movimentada;
- Quando determinamos o posicionamento relativo ao elemento, este começará a ser contado a partir da posição atual do elemento no documento;
- Quando cada elemento do documento HTML é tratado como uma caixa ou um box, uma das principais formas de posicionar este elemento no documento é por meio da propriedade **float**;
- O Flexible Box Layout ou simplesmente FlexBox Layout não utiliza mais as contas que temos que fazer ao utilizar o box model.



**3**

# **Posicionamento e alinhamento**

## **Teste seus conhecimentos**

SCAM 50°  
50.70A  
007-67  
TDA



**IMPACTA**  
EDITORA

**1. Para posicionar um elemento de maneira que este não se mova junto com a barra de rolagem, qual declaração devemos utilizar?**

- a) position:fixed;
- b) float:fixed
- c) position absolute;
- d) position:relative;
- e) margin-top:fixed;

**2. Qual o valor padrão da propriedade position?**

- a) fixed
- b) relative
- c) absolute
- d) static
- e) none

**3. Para determinarmos que um elemento deva ser alinhado à esquerda no módulo box model, qual declaração devemos utilizar?**

- a) margin-left:float;
- b) float:relative
- c) position:float;
- d) float:left;
- e) float:min-left;

**4. Para evitar que o conteúdo que excedeu o limite de largura e altura de um elemento seja exibido, causando assim uma quebra de layout, qual declaração devemos utilizar?**

- a) clear:both;
- b) overflow:hidden;
- c) max-width e max\_height.
- d) limit
- e) reflow

**5. Para utilizar o módulo Flex Box Layout sem prefixos proprietários (-webkit, -ms...), qual declaração devemos utilizar?**

- a) display:flexbox;
- b) flex-display;
- c) display:inline-box.
- d) display:block;
- e) display:flex;

**6. Para alinhar os elementos dentro de um flex container, de forma centralizada, qual declaração devemos utilizar?**

- a) align:center;
- b) align:items:center;
- c) align\_content:center.
- d) align-items:center;
- e) align-itens:center;

3

# Posicionamento e alinhamento

## Mãos à obra!

SCAMIA 1000-67  
59.704.901  
SCAMIA 1000-67



**IMPACTA**  
EDITORA

No exercício a seguir iremos criar um projeto de layout para smartphone, tablet e desktop. Esta atividade será utilizada no laboratório dos capítulos a seguir.

## Laboratório 1

### A - Criando arquivos de base

1. Crie uma pasta no seu diretório atual de trabalho chamada **projeto**;
2. Dentro da pasta **projeto**, crie as pastas **css**, **images** e **js**;
3. Dentro da raiz da pasta **projeto** coloque os três arquivos HTML (**smartphone.html**, **tablet.html**, **desktop.html**) fornecidos pelo instrutor, que servirão de base para a aplicação CSS;
4. Crie três arquivos na pasta **css** (**smartphone.css**, **tablet.css**, **desktop.css**).

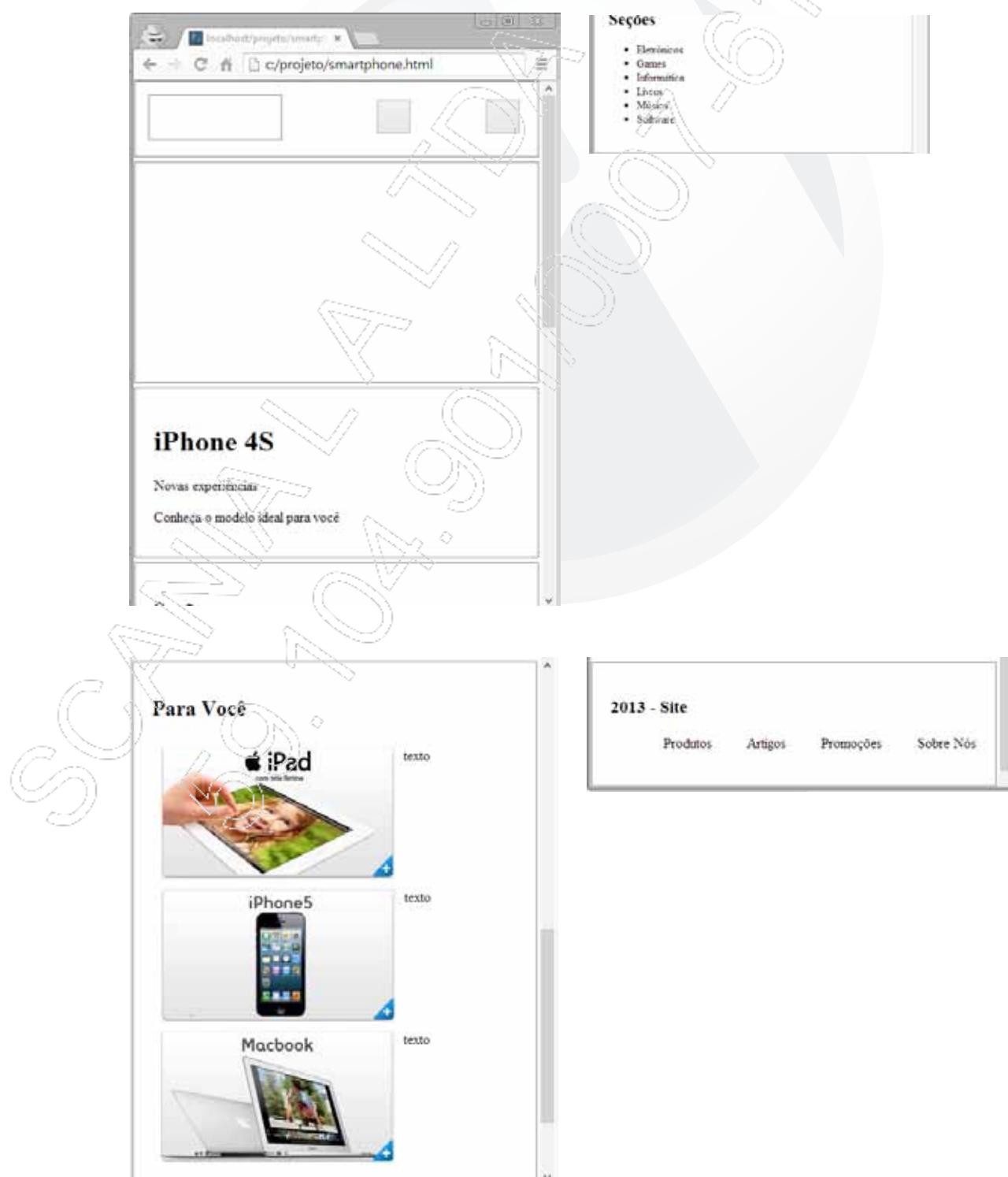
## Laboratório 2

### A – Criando a base de um layout para smartphone

1. Abra o arquivo **smartphone.css** e coloque a estrutura padrão a seguir:

```
@charset "UTF-8";  
body{  
    margin: 0px; padding: 0px;  
}
```

O objetivo é criar a estrutura de layout do site utilizando as propriedades de posicionamento e alinhamento, conforme a imagem a seguir:



# CSS3

Para tanto, colocar as propriedades e valores da tabela a seguir no arquivo **smartphone.css**:

Seletores	Propriedade	Valor
header, #logo-css3, #banner-full, .descricao, #secoes, #para-voce, footer	border	solid 1px #777
	position	relative
	float	left
header, #banner-full	width	99%
header	margin	0
	height	85px
#logo-css3	width	150px
	height	50px
	margin	15px
#button-menu-nav	width	40px
	height	40px
	margin-top	20px
	margin-left	20%
#menu-nav ul	display	None
	width	40px
	height	40px
#button-lupa	top	20px
	right	20px
	position	relative
	float	Right
#banner-full	margin-top	5px
	height	250px
.descricao	height	150px
	width	90%
.descricao, #secoes, #para-voce, footer	margin-top	5px
	padding	4.5%
#secoes, #para-voce, footer	min-height	200px
#para-voce img, #para-voce p	margin	10px 10px
footer	min-height	50px
footer li	padding	20px
	display	inline

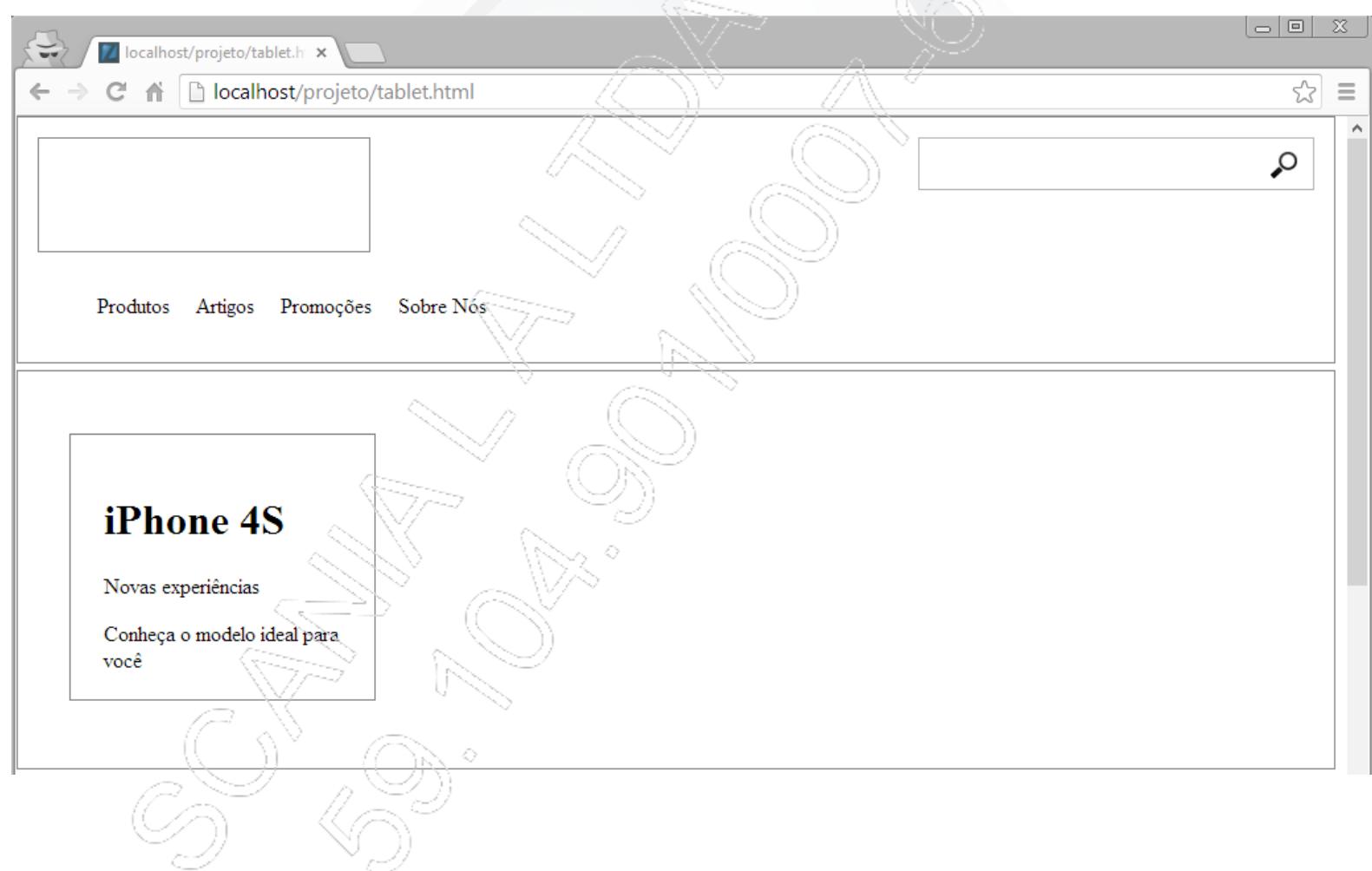
# Laboratório 3

## A - Criando a base de um layout para tablet

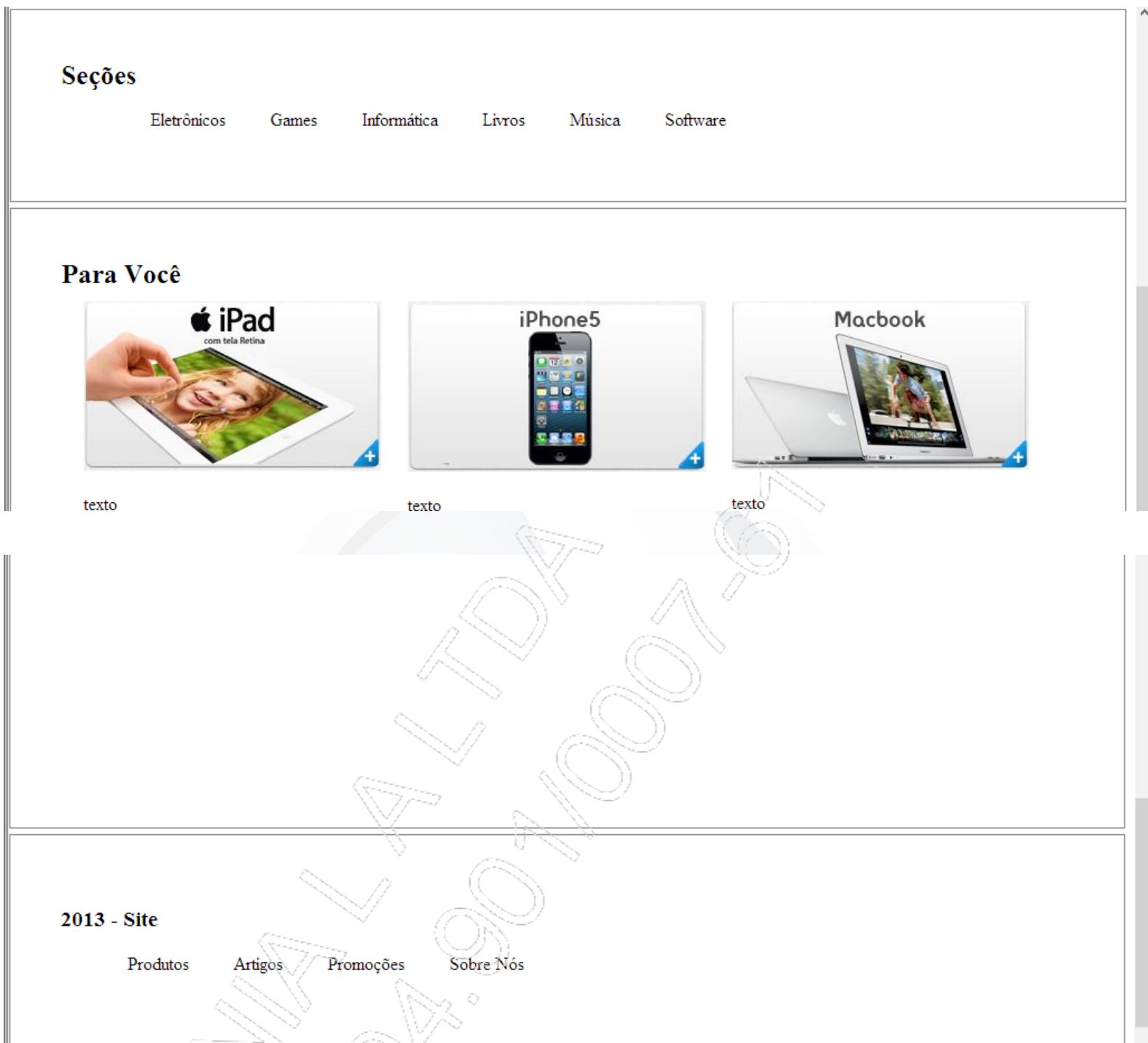
1. Abra o arquivo **tablet.css** e coloque a estrutura padrão que segue:

```
@charset "UTF-8";  
body{  
    margin: 0px; padding: 0px;  
}
```

O objetivo é criar a estrutura de layout do site utilizando as propriedades de posicionamento e alinhamento, conforme a imagem a seguir:



# CSS3



Para tanto, colocar as propriedades e valores da tabela a seguir no arquivo tablet.css:

Seletores	Propriedade	Valor
header, #logo-css3, #banner-full, .descricao, #secoes, #para-voce, footer	border	solid 1px #777
	position	relative
	float	left
header, #banner-full	width	99%
header	margin	0
	height	185px

Seletores	Propriedade	Valor
#logo-css3	width	250px
	height	85px
	margin	15px
#button-menu-nav	width	40px
	height	40px
	margin-top	20px
	margin-left	20%
#menu-nav	float	left
	width	90%
	clear	both
#menu-nav li	padding-left	20px
	display	inline
	width	300px
input[type='search']	height	40px
	margin	15px
	float	right
#button-lupa	width	35px
	height	35px
	top	20px
	right	20px
#button-lupa img	position	absolute
	float	Right
	background	none
	z-index	1
#banner-full	width	20px
	height	20px
#banner-full	margin-top	5px
	height	300px

# CSS3

Seletores	Propriedade	Valor
.descricao	width	180px
	height	150px
	padding	25px
	position	absolute
	float	left
	top	240px
	left	40px
	z-index	1
#secoes, #para-voce, footer	width	90%
#secoes	margin-top	5px
#secoes li	padding	4.5%
	min-height	50px
	display	inline
	padding-left	40px
h2	margin	0
	padding	0
#para-voce, footer	min-height	200px
#para-voce img, #para-voce p	margin	10px 10px
footer	float	left
footer li	min-height	50px
	padding	20px
	display	inline
.para-voce-item	width	280px
	height	400px
	float	left
	margin-left	10px
	position	relative

# Trabalhando com cores, backgrounds e gradientes

4

- ✓ Trabalhando com cores;
- ✓ Trabalhando com backgrounds;
- ✓ Trabalhando com gradientes.

## 4.1. Trabalhando com cores

Um dos princípios da formatação nas folhas de estilo CSS é aquele relacionado às cores, sendo ela utilizada para textos, backgrounds, bordas e outras partes dos elementos em um documento. Neste capítulo veremos como trabalhar com as cores e também sua opacidade, além do background e do gradiente na CSS3.

As cores em CSS podem ser declaradas pelos seguintes métodos:

- Declaração Hexadecimal;
- Declaração RGB (red, green, blue);
- Declaração RGBA (red, green, blue, alpha-opacity);
- Declaração HSL (hue, saturation, lightness);
- Declaração HSLA (hue, saturation, lightness, alpha-opacity);
- Declaração pré-definida.

Na maioria dos documentos HTML, são utilizadas as declarações Hexadecimal, RGB e HSL. No entanto, houve uma evolução no módulo CSS3, abrindo o leque para novos formatos: são o RGBA e o HSLA, que permitem definir a opacidade ou a transparência em uma escala de 0.0 (100% transparente) a 1.0 (0% transparente).

### 4.1.1. Declaração Hexadecimal

A declaração Hexadecimal é suportada na maioria dos navegadores disponíveis no mercado. A declaração é feita com #RRGGBB, onde o RR representa a cor vermelha, GG a cor verde e BB a cor azul. Os valores são representados por números hexadecimais e devem ser entre 0 e FF.

```
1  p
2  {
3      background-color: #FF00CC;
4 }
```

## 4.1.2. Declaração RGB

A declaração RGB é suportada na maioria dos navegadores disponíveis no mercado. A declaração pode ser feita em porcentagens entre 0% a 100%, ou pode ser designada por um número inteiro entre 0 e 255.

```
1 p
2 {
3     background-color: rgb(0,10,255);
4 }
```

ou

```
1 p
2 {
3     background-color: rgb(10%,20%,30%);
4 }
```

 Não é permitido declarar itens usando número e porcentagem.

## 4.1.3. Declaração RGBA

Esta é uma declaração disponível na versão CSS3 e os navegadores que a suportam são:

Navegador	Versão
Internet Explorer	9 ou superior
Mozilla Firefox	3 ou superior
Google Chrome	Todas as versões
Safari	Todas as versões
Opera	10 ou superior

Esta declaração é uma extensão da declaração RGB, com a possibilidade adicional da opacidade para o elemento gráfico. O parâmetro de opacidade deve ser um número entre 0.0 (transparente) e 1.0 (opaco).

```
1  div
2  {
3      background-color: rgba(255,0,10,0.5);
4 }
```

## 4.1.4. Declaração HSL

Esta é uma declaração disponível na versão CSS3 e os navegadores que a suportam são:

Navegador	Versão
Internet Explorer	9 ou superior
Mozilla Firefox	3 ou superior
Google Chrome	Todas as versões
Safari	Todas as versões
Opera	10 ou superior

A declaração HSL utiliza três parâmetros: **Hue** (tom), **Saturation** (saturação) e **Lightness** (luminosidade). O primeiro valor é o tom da cor e a medida de um ângulo (um valor em torno de 0 a 360 graus), e os dois outros parâmetros são expressos em porcentagens entre 0% e 100%.

```
1  div
2  {
3      background-color: hsl(0,100%,50%);
4 }
```

## 4.1.5. Declaração HSLA

Esta é uma declaração disponível na versão CSS3 e os navegadores que a suportam são:

Navegador	Versão
Internet Explorer	9 ou superior
Mozilla Firefox	4 ou superior
Google Chrome	Todas as versões
Safari	Todas as versões
Opera	10 ou superior

Esta declaração é uma extensão da declaração HSLA, com a possibilidade de ser adicionada a opacidade para o elemento gráfico. O parâmetro de opacidade deve ser um número entre 0.0 (transparente) e 1.0 (opaco).

```
1 div
2 {
3     background-color: hsl(0,100%,50%,0.5);
4 }
```

## 4.1.6. Declaração predefinida

Existem 147 nomes de cores preefinidas no CSS, tais como red, black, white, etc. Estas declarações predefinidas são suportadas em todos os navegadores.

## 4.2. Trabalhando com backgrounds

A propriedade background define cor ou imagem de um elemento, e na versão CSS2.1 existiam as seguintes propriedades:

- background;
- background-image;
- background-repeat;
- background-attachment;
- background-position.

No CSS3 foram criadas as propriedades a seguir:

- background-clip;
- background-origin;
- background-size;
- backgrounds múltiplos.

Veja os navegadores que atendem estas novas propriedades:

Navegador	Versão
Internet Explorer	9 ou superior
Mozilla Firefox	4 ou superior
Google Chrome	4 ou superior
Safari	5 ou superior
Opera	10.5 ou superior

## 4.2.1. Propriedade background-clip

Esta propriedade é usada para determinar a posição em que o background começará a ser desenhado, sendo ele uma imagem ou uma cor. Existem três valores que podem ser utilizados: **border-box**, **content-box** e **padding-box**.

### 4.2.1.1. border-box

Este valor preenche todo o elemento, inclusive a borda.

```
1 <style type="text/css">
2   div{
3     border: 10px dashed black;
4     padding: 20px;
5     font-size: 20px;
6     text-align: center;
7     background-color: #CCC;
8   }
9   .fundo{
10    background-clip: border-box;
11  }
12 </style>
13 <div class="fundo">
14   Treinamento de CSS3 - Impacta Tecnologia
15 </div>
```

Resultado:

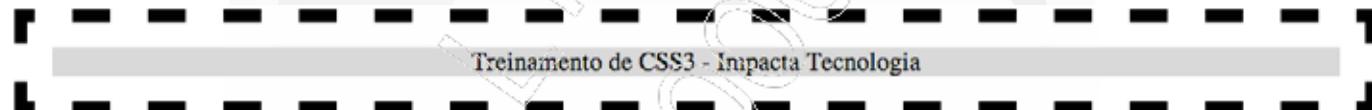


## 4.2.1.2.content-box

Este valor preenche apenas o conteúdo dentro do elemento.

```
1 <style type="text/css">
2   div{
3     border: 10px dashed black;
4     padding: 20px;
5     font-size: 20px;
6     text-align: center;
7     background-color: #CCC;
8   }
9   .fundo{
10    background-clip: content-box;
11  }
12 </style>
13 <div class="fundo">
14   Treinamento de CSS3 - Impacta Tecnologia
15 </div>
```

Resultado:

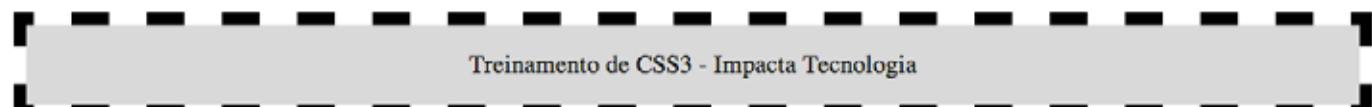


## 4.2.1.3.padding-box

Este valor preenche todo o elemento, com exceção da borda.

```
1 <style type="text/css">
2   div{
3     border: 10px dashed black;
4     padding: 20px;
5     font-size: 20px;
6     text-align: center;
7     background-color: #CCC;
8   }
9   .fundo{
10    background-clip: padding-box;
11  }
12 </style>
13 <div class="fundo">
14   Treinamento de CSS3 - Impacta Tecnologia
15 </div>
```

Resultado:



## 4.2.2. Propriedade background-origin

Esta propriedade é utilizada para definir a posição relativa da imagem com relação ao elemento. Existem três valores que podem ser utilizados: **border-box**, **content-box** e **padding-box**.

### 4.2.2.1. border-box

Este valor é utilizado para definir a posição da imagem com base na borda do elemento.

```
1 <style type="text/css">
2   div
3   {
4     border:5px dashed black;
5     padding: 50px;
6     background-image: url(css3logo.jpg);
7     background-size: 50px 50px;
8     background-repeat: no-repeat;
9     background-color: #CCC;
10    }
11   .fundo{
12     background-origin: border-box;
13   }
14 </style>
15 <div class="fundo">
16   Treinamento de CSS3 - Impacta Tecnologia
17 </div>
```

Resultado:



## 4.2.2.2.content-box

Este valor é utilizado para definir a posição da imagem com base no conteúdo do elemento.

```
1 <style type="text/css">
2   div
3   {
4     border:5px dashed black;
5     padding: 50px;
6     background-image: url(images/css3logo.jpg);
7     background-size: 50px 50px;
8     background-repeat: no-repeat;
9     background-color: #CCC;
10    }
11    .fundo{
12      background-origin: content-box;
13    }
14  </style>
15  <div class="fundo">
16    Treinamento de CSS3 - Impacta Tecnologia
17  </div>
```

Resultado:



## 4.2.2.3.padding-box

Este valor é utilizado para definir a posição da imagem com base no espaçamento do elemento.

```
1 <style type="text/css">
2   div
3   {
4     border:5px dashed black;
5     padding: 50px;
6     background-image: url(images/css3logo.jpg);
7     background-size: 50px 50px;
8     background-repeat: no-repeat;
9     background-color: #CCC;
10    }
11    .fundo{
12      background-origin: padding-box;
13    }
14  </style>
15  <div class="fundo">
16    Treinamento de CSS3 - Impacta Tecnologia
17  </div>
```

Resultado:



## 4.2.3. Propriedade background-size

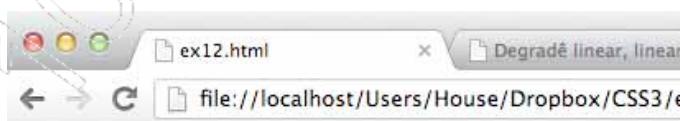
Esta propriedade é utilizada para definir o tamanho da imagem utilizada. O valor recebido pode ser um tamanho fixo em pixel, porcentagem ou, ainda, utilizados os valores definidos como **cover** ou **contain**.

### 4.2.3.1. Definindo tamanho com valor fixo

Utilizando este valor, o primeiro item a ser definido é a largura da imagem (width) e o segundo a altura (height), ambos fixados em pixel. Caso somente um valor seja atribuído, por padrão o segundo pode ser definido como ‘auto’, ou seja, o valor é atribuído automaticamente.

```
1 <style type="text/css">
2   div{
3     width: 500px;
4     height: 500px;
5   }
6   #fundo
7   {
8     background-image: url(images/css3logo.jpg);
9     background-repeat: no-repeat;
10    background-size: 200px 200px;
11  }
12 </style>
13 <div id="fundo"></div>
```

Resultado:



Temos ainda a opção utilizando o valor ‘auto’:

```
1 <style type="text/css">
2   div{
3     width: 500px;
4     height: 500px;
5   }
6   #fundo
7   {
8     background-image: url(images/css3logo.jpg);
9     background-repeat: no-repeat;
10    background-size: auto;
11  }
12 </style>
13 <div id="fundo"></div>
```

Resultado:

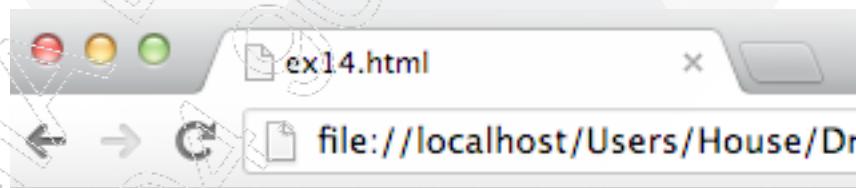


## 4.2.3.2. Definindo tamanho com porcentagem

Utilizando este valor, o primeiro item a definir é a largura da imagem (width) e o segundo a altura (height), ambos fixados em porcentagem. Caso somente um valor seja atribuído, por padrão o segundo pode ser definido como ‘auto’, ou seja, o valor é atribuído automaticamente.

```
1 <style type="text/css">
2   div{
3     width: 500px;
4     height: 500px;
5   }
6   #fundo
7   {
8     background-image: url(images/css3logo.jpg);
9     background-repeat: no-repeat;
10    background-size: 5% 10%;
11  }
12 </style>
13 <div id="fundo"></div>
```

Resultado:

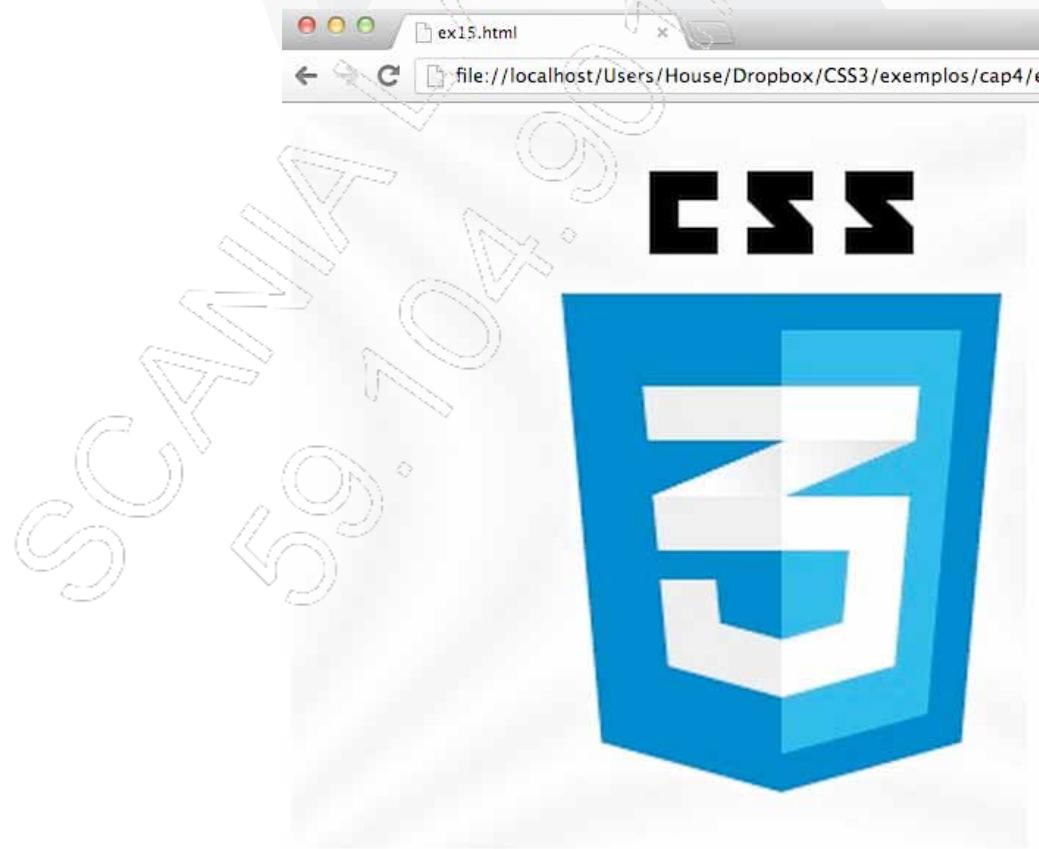


## 4.2.3.3. Definindo tamanho com o valor cover

Utilizando este valor, o background definido é escalável, utilizando por completo a área do elemento definido na tela.

```
1 <style type="text/css">
2   div{
3     width: 500px;
4     height: 500px;
5   }
6   #fundo
7   {
8     background-image: url(images/css3logo.jpg);
9     background-repeat: no-repeat;
10    background-size: cover;
11  }
12 </style>
13 <div id="fundo"></div>
```

Resultado:



## 4.2.3.4. Definindo tamanho com o valor contain

Utilizando este valor, a imagem preenche com base limite no tamanho da largura (width) e da altura (height) o elemento definido na tela.

```
1 <style type="text/css">
2   div{
3     width: 500px;
4     height: 500px;
5   }
6   #fundo
7   {
8     background-image: url(images/css3logo.jpg);
9     background-repeat: no-repeat;
10    background-size: contain;
11  }
12 </style>
13 <div id="fundo"></div>
```

Resultado:

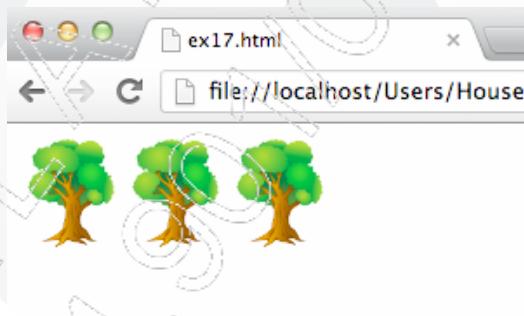


## 4.2.3.5. Múltiplos backgrounds

Um dos recursos mais incríveis do CSS3 é o de permitir a inserção de mais de um background em um elemento.

```
1 <style type="text/css">
2   div{
3     width: 100%;
4     height: 100%;
5   }
6   .fundo{
7     background-image: url(images/arvore.png), url(images/arvore.png);
8     background-repeat: no-repeat;
9     background-size: auto;
10    background-position: left top, 60 0, 120 0;
11  }
12 </style>
13 <div class="fundo"></div>
```

Resultado:



Segue uma lista dos navegadores nos quais este recurso é compatível:

Navegador	Versão
Internet Explorer	9 ou superior
Mozilla Firefox	3.6 ou superior
Google Chrome	11 ou superior
Safari	1.3 ou superior
Opera	10 ou superior

## 4.3. Trabalhando com gradientes

Com a propriedade **gradient** é possível inserir efeitos em degradê (gradiente) sem a utilização de imagens, utilizando puro código CSS3. Como ilustram as tabelas a seguir, na declaração dessas funções só é alterado o início da sintaxe, pois o início representa justamente qual motor de navegador definido que irá funcionar.

Existem dois tipos de gradientes:

- **Linear:** Função linear-gradient
- **Radial:** Função radial-gradient

Para o uso desta função é necessário utilizar dois parâmetros:

- **Origem e/ou ângulo do degradê:** O primeiro parâmetro seria a origem, onde começará o degradê e/ou o ângulo de disposição do gradiente de cor. Pode ser definido que o efeito comece de cima, de baixo ou a partir de qualquer canto. Por padrão, os degradês serão distribuídos em um gradiente em linha reta, mas também é possível indicar um ângulo diferente com o qual se irá produzir o gradiente de cor.
- **Lista de cores e, opcionalmente, o lugar até onde se deve mostrar cada uma:** Em seguida, é preciso colocar as cores, todas as necessárias, que devem ser utilizadas no degradê, separadas por vírgulas. Além disso, se preferir, existe a possibilidade de definir as paradas de cor “color stops”, que consiste em declarar o lugar onde deve começar o gradiente da cor.

### 4.3.1. Gradiente Linear

Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja a seguir como esta propriedade irá funcionar:

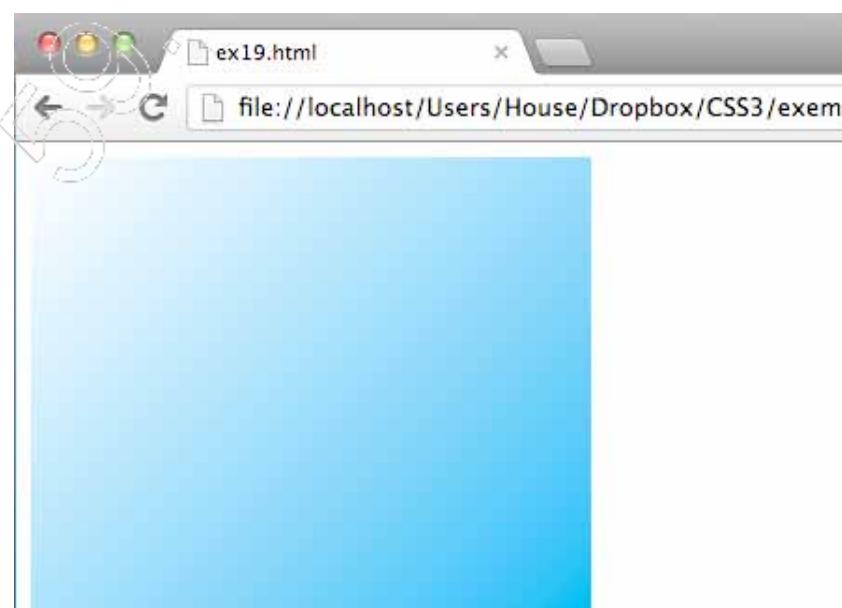
Navegador	Sintaxe
Internet Explorer 10	-ms-linear-gradient
Mozilla Firefox 3.6+	-moz-linear-gradient
Opera 11.10+	-o-linear-gradient
WebKit (Chrome 11+ / Safari 5.1)	-webkit-linear-gradient
W3C Markup, IE 10 Release Preview	linear-gradient

Exemplo:

```

1 <style type="text/css">
2   div{
3     width: 300px;
4     height: 300px;
5   }
6   .fundo{
7     background-image: -ms-linear-gradient(top left, #FFFFFF 0%, #00A3EF 100%);
8     background-image: -moz-linear-gradient(top left, #FFFFFF 0%, #00A3EF 100%);
9     background-image: -o-linear-gradient(top left, #FFFFFF 0%, #00A3EF 100%);
10    background-image: -webkit-linear-gradient(top left, #FFFFFF 0%, #00A3EF 100%);
11    background-image: linear-gradient(to bottom right, #FFFFFF 0%, #00A3EF 100%);
12  }
13 </style>
14 <div class="fundo"></div>
```

Resultado:



## 4.3.2. Gradiente Radial

Nesta opção existe a possibilidade de se definir um gradiente **Circular** ou **Elíptico**.

### 4.3.2.1. Gradiente Radial Circular

Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja a seguir como esta propriedade irá funcionar:

Navegador	Sintaxe
Internet Explorer 10	-ms-radial-gradient
Mozilla Firefox 3.6+	-moz-radial-gradient
Opera 11.10+	-o-radial-gradient
WebKit (Chrome 11+ / Safari 5.1)	-webkit-radial-gradient
W3C Markup, IE 10 Release Preview	radial-gradient

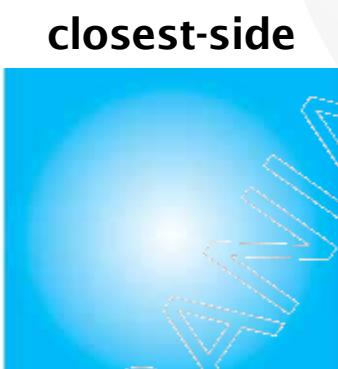
Exemplo:

```
1 <style type="text/css">
2   div{
3     width: 300px;
4     height: 300px;
5   }
6   .fundo{
7     background-image: -ms-radial-gradient(center, circle farthest-corner, #FFFFFF 0%, #00A3EF 100%);
8     background-image: -moz-radial-gradient(center, circle farthest-corner, #FFFFFF 0%, #00A3EF 100%);
9     background-image: -o-radial-gradient(center, circle farthest-corner, #FFFFFF 0%, #00A3EF 100%);
10    background-image: -webkit-radial-gradient(center, circle farthest-corner, #FFFFFF 0%, #00A3EF 100%);
11    background-image: radial-gradient(circle farthest-corner at center, #FFFFFF 0%, #00A3EF 100%);
12  }
13 </style>
14 <div class="fundo"></div>
```

Resultado:



Veja que neste exemplo foi utilizado o parâmetro **circle**, que possibilita um gradiente **circular**, além do parâmetro **farthest-corner**, que define o tamanho do gradiente. É possível utilizar também **closest-side**, **closest-corner** e **farthest-side**.



**closest-corner**



**farthest-side**

## 4.3.2.2.Gradiente Radial Elíptico

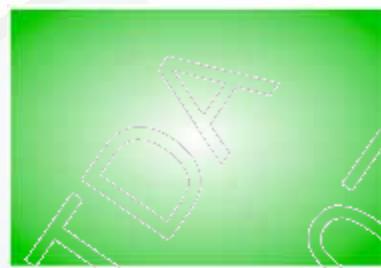
Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja a seguir como esta propriedade irá funcionar:

Navegador	Sintaxe
Internet Explorer 10	-ms-radial-gradient
Mozilla Firefox 3.6+	-moz-radial-gradient
Opera 11.10+	-o-radial-gradient
WebKit (Chrome 11+ / Safari 5.1)	-webkit-radial-gradient
W3C Markup, IE 10 Release Preview	radial-gradient

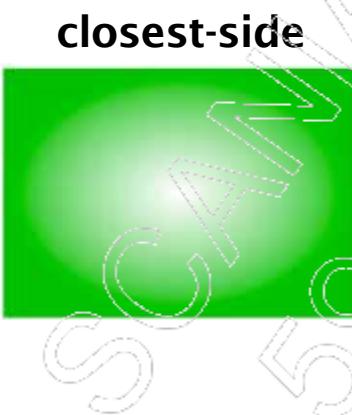
Exemplo:

```
1 <style type="text/css">
2   div{
3     width: 300px;
4     height: 300px;
5   }
6   .fundo{
7     background-image: -ms-radial-gradient(center, ellipse farthest-corner, #FFFFFF 0%, #00AA00 100%);
8     background-image: -moz-radial-gradient(center, ellipse farthest-corner, #FFFFFF 0%, #00AA00 100%);
9     background-image: -o-radial-gradient(center, ellipse farthest-corner, #FFFFFF 0%, #00AA00 100%);
10    background-image: -webkit-radial-gradient(center, ellipse farthest-corner, #FFFFFF 0%, #00AA00 100%);
11    background-image: radial-gradient(ellipse farthest-corner at center, #FFFFFF 0%, #00AA00 100%);
12  }
13 </style>
14 <div class="fundo"></div>
```

Resultado:



Veja que neste exemplo foi utilizado o parâmetro **ellipse**, que possibilita um gradiente elíptico, além do parâmetro **farthest-corner**, que define o tamanho do gradiente. É possível utilizar também **closest-side**, **closest-corner** e **farthest-side**.



# Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- A formatação de **cores** nas folhas de estilo CSS é um de seus principais recursos, sendo utilizada para textos, backgrounds, bordas e outras partes dos elementos em um documento;
- A propriedade **background** define cor ou imagem de um elemento, e na CSS3 foram criadas as seguintes: **background-clip**, **background-origin**, e **background-size**, além dos **backgrounds múltiplos**;
- A propriedade **gradient** permite inserir efeitos em degradê (gradiente) sem a utilização de imagens, utilizando puro código CSS3, sendo eles o gradiente linear e o radial (circular ou elíptico).

4

# Trabalhando com cores, backgrounds e gradientes

Teste seus conhecimentos



**IMPACTA**  
EDITORA

## 1. Qual das alternativas abaixo é incorreta?

- a) background-color: rgb(0, 10, 255)
- b) background-color: rgba(0, 10, 255, 1.0)
- c) background-color: rgba(0, 10, 255, 0.1)
- d) background-color: rgb(0, 10%, 255)
- e) background-color: hsla(0%, 10%, 99%, 0.1)

## 2. Qual a função da propriedade background-origin?

- a) Definir a posição relativa da imagem com relação ao elemento.
- b) Definir a porcentagem da imagem no tamanho original do elemento.
- c) Preencher todo o elemento.
- d) Definir a absoluta relativa da imagem com relação ao elemento.
- e) Nenhuma das alternativas anteriores está correta.

## 3. Quantos backgrounds são possíveis de inserir em um único elemento?

- a) 1
- b) 2
- c) 3
- d) Não tem limite.
- e) Nenhuma das alternativas anteriores está correta.

## 4. Quais os tipos de gradientes disponíveis?

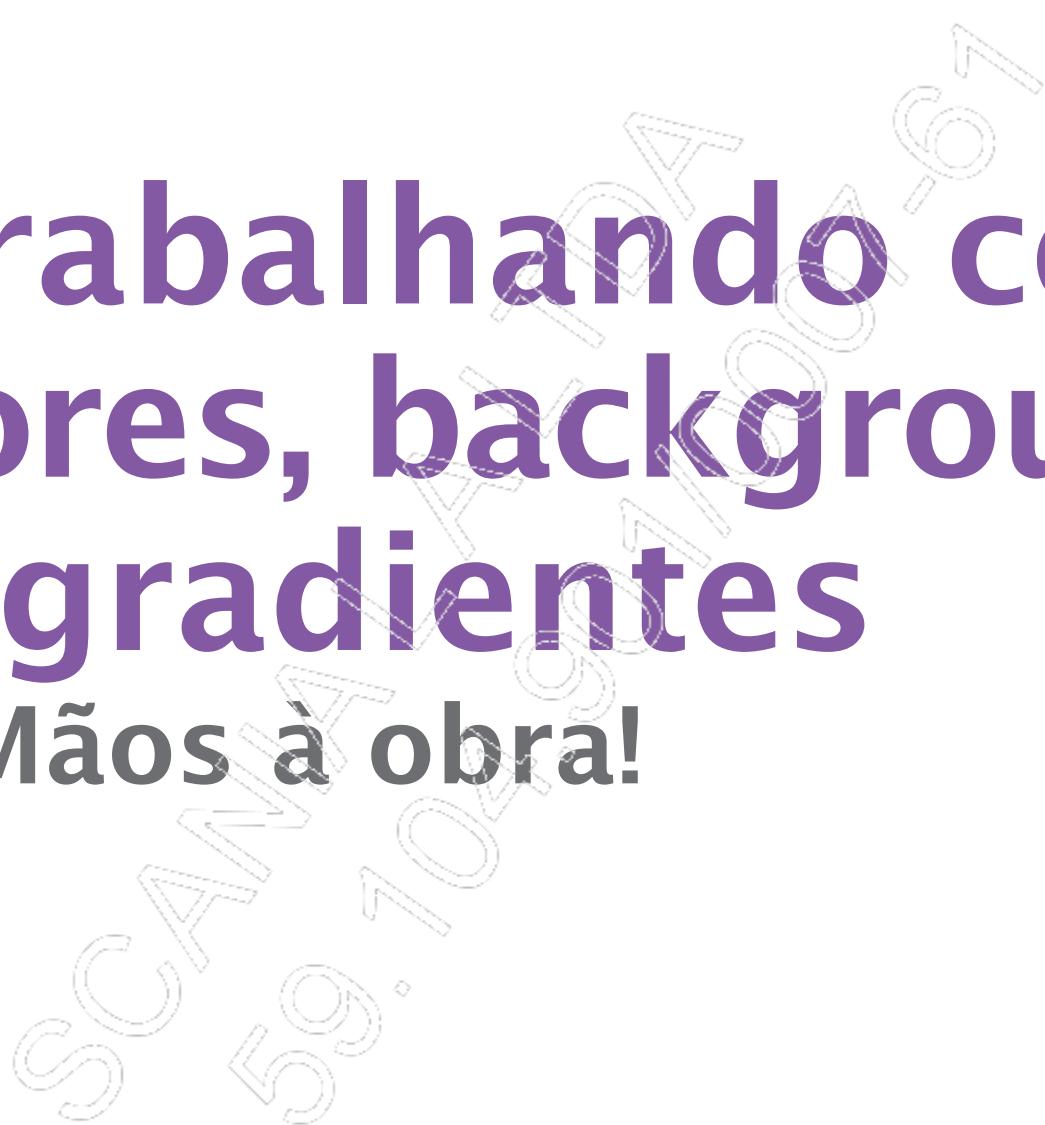
- a) linear-gradient e radial-gradient
- b) circle-gradient e ellipse-gradient
- c) linear-gradient e circle-gradient
- d) radial-gradient e circle-gradient
- e) Nenhuma das alternativas anteriores está correta.



# 4

# Trabalhando com cores, backgrounds e gradientes

Mãos à obra!



**IMPACTA**  
EDITORA

## Laboratório 1

### A - Inserindo um background

Abra, na pasta projeto criada no laboratório anterior, os arquivos **smartphone.css**, **tablet.css**, **desktop.css**. Adicione a propriedade **background-image** ao seletor de id **#logo-css3**, carregando a imagem **css3-logo.png**, sem repetição.



# Bordas e sombras

5

- ✓ Bordas;
- ✓ Box shadow;
- ✓ Text shadow.

SCANNY ALTA  
59.704.007-67



**IMPACTA**  
EDITORA

## 5.1. Bordas

Podemos definir estilos de molduras, mais conhecidas como **bordas**, para um elemento, onde suas quatro faces podem ter ou um estilo uniforme ou um estilo aplicado a cada uma delas individualmente.

O uso dessa propriedade no CSS3 torna possível definir tamanho, estilo, cor, arredondamento ou mesmo ter uma imagem adicionada como borda. As propriedades disponíveis são:

- border-width;
- border-style;
- border-color;
- border-radius (propriedade disponível apenas no CSS3);
- border-image (propriedade disponível apenas no CSS3).

### 5.1.1. border-width

Essa propriedade é utilizada para definir a espessura de uma borda. É possível definir a espessura individualmente, uma para cada lado do elemento.

```
1 div{  
2     border-width-top: 10px; /* Propriedade que define o tamanho da borda superior */  
3     border-width-right: 8px; /* Propriedade que define o tamanho da borda do lado direito */  
4     border-width-bottom: 6px; /* Propriedade que define o tamanho da borda inferior */  
5     border-width-left: 4px; /* Propriedade que define o tamanho da borda do lado esquerdo */  
6 }
```

É possível também usar uma sintaxe abreviada que pode ser escrita de quatro maneiras:

```
1 div{  
2     border-width: 10px 10px 10px 10px; /* Top, right, bottom, left */  
3     border-width: 10px 5px 10px; /* top (right / left) bottom */  
4     border-width: 10px 10px; /* (top / bottom) (right / left) */  
5     border-width: 10px; /* (top / right / bottom / left) */  
6 }
```

## 5.1.2. border-style

Essa propriedade é utilizada para definir o estilo de uma borda. A seguir, os valores disponíveis que podem ser utilizados:

Estilo	Descrição
dotted	Define uma borda pontilhada.
dashed	Define uma borda tracejada.
solid	Define uma borda contínua.
double	Define uma borda com duas linhas contínuas.
groove	Define uma borda com aparência entalhada, sendo que este valor depende da atribuição da propriedade border-color.
ridge	Define uma borda com aparência ressaltada, sendo que este valor depende da atribuição da propriedade border-color.
inset	Define uma borda em baixo-relevo, sendo que este valor depende da atribuição da propriedade border-color.
outset	Define uma borda em alto-relevo, sendo que este valor depende da atribuição da propriedade border-color.
none	Define que o elemento não utilizará uma borda.
hidden	Define que o elemento não exibirá uma borda.

Assim como o tamanho das bordas, é possível também inserir individualmente um estilo para cada uma das extremidades do elemento:

```

1 div{
2   border-style-top: solid; /* Propriedade que define o estilo da borda superior */
3   border-style-right: solid; /* Propriedade que define o estilo da borda do lado direito */
4   border-style-bottom: solid; /* Propriedade que define o estilo da borda inferior */
5   border-style-left: solid; /* Propriedade que define o estilo da borda do lado esquerdo */
6 }
```

É possível também usar uma sintaxe abreviada que pode ser escrita de quatro maneiras:

```

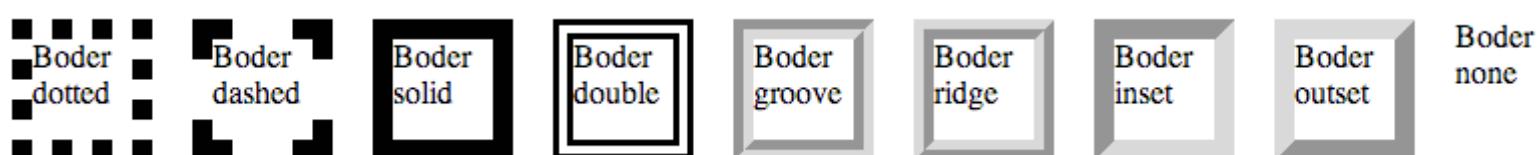
1 div{
2   border-style: solid none dashed solid; /* Top, right, bottom, left */
3   border-style: dashed solid dashed; /* top (right / left) bottom */
4   border-style: solid double; /* (top / bottom) (right / left) */
5   border-style: solid; /* (top / right / bottom / left) */
6 }
```

# CSS3

A seguir um exemplo com a utilização de todas as variações da propriedade **border-style**, com seus valores definidos:

```
1 <style type="text/css">
2   div{
3     border-width: 10px;
4     width: 50px;
5     height: 50px;
6     margin: 10px;
7     float: left;
8   }
9
10 .border_dotted{
11   border-style: dotted;
12 }
13 .border_dashed{
14   border-style: dashed;
15 }
16 .border_solid{
17   border-style: solid;
18 }
19 .border_double{
20   border-style: double;
21 }
22
23 /* Para o estilo groove ter efeito é necessária ter a propriedade border-color */
24 .border_groove{
25   border-style: groove;
26   border-color: #CCC;
27 }
28
29 /* Para o estilo ridge ter efeito é necessária ter a propriedade border-color */
30 .border_ridge{
31   border-style: ridge;
32   border-color: #CCC;
33 }
34
35 /* Para o estilo inset ter efeito é necessária ter a propriedade border-color */
36 .border_inset{
37   border-style: inset;
38   border-color: #CCC;
39 }
40
41 /* Para o estilo outset ter efeito é necessária ter a propriedade border-color */
42 .border_outset{
43   border-style: outset;
44   border-color: #CCC;
45 }
46 .border_none{
47   border-style: none;
48 }
49 </style>
50 <div class="border_dotted">Boder dotted</div>
51 <div class="border_dashed">Boder dashed</div>
52 <div class="border_solid">Boder solid</div>
53 <div class="border_double">Boder double</div>
54 <div class="border_groove">Boder groove</div>
55 <div class="border_ridge">Boder ridge</div>
56 <div class="border_inset">Boder inset</div>
57 <div class="border_outset">Boder outset</div>
58 <div class="border_none">Boder none</div>
```

O resultado será:



## 5.1.3. border-color

Essa propriedade é utilizada para definir a cor de uma borda. Essas cores podem ser definidas utilizando um código hexadecimal, RGB, RGBA, HSL, HSLA ou inscritas de modo declarado.

É possível também definir a espessura individualmente, uma para cada lado do elemento.

```
1 div{
2   border-color-top: #CCC; /* Propriedade que define a cor da borda superior */
3   border-color-right: red; /* Propriedade que define a cor da borda do lado direito */
4   border-color-bottom: hsla(100,100,75%, 0.8); /* Propriedade que define a cor da borda inferior */
5   border-color-left: rgba(100,255,100,0.8); /* Propriedade que define a cor da borda do lado esquerdo */
6 }
```

Assim como o tamanho das bordas e o estilo, é possível ainda inserir individualmente uma cor para cada uma das extremidades:

```
1 div{
2   border-color: red #CCC #CCC black; /* Top, right, bottom, left */
3   border-color: rgb(10,255,10) hsla(10,100%,75%,0.5) black; /* top (right / left) bottom */
4   border-color: black rgb(10,255,10); /* (top / bottom) (right / left) */
5   border-color: #CDC; /* (top / right / bottom / left) */
6 }
```

## 5.1.4. border-radius

Anteriormente, no CSS2, era mais complicado adicionar bordas arredondadas, pois havia a necessidade de usar diferentes imagens para cada extremidade. Já no CSS3, criar bordas arredondadas é simples e fácil: existe uma propriedade específica na qual é possível inserir o tamanho do arredondamento em pixels.

Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja como esta propriedade irá funcionar:

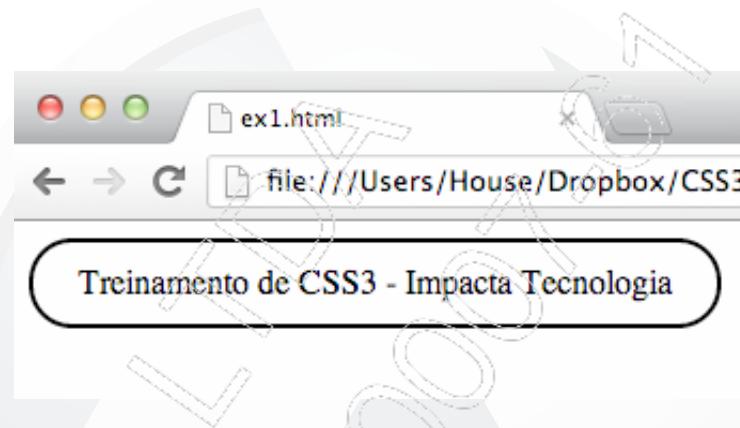
Navegador	Sintaxe
Internet Explorer 9 ou superior	-ms-border-radius
Mozilla Firefox 3.6+	-moz-border-radius
Opera 11.10+	-o-border-radius
WebKit (Chrome 11+ / Safari 5.1)	-webkit-border-radius
W3C Markup, IE 10 Release Preview	border-radius

# CSS3

Exemplo:

```
1 <style type="text/css">
2   div{
3     width: 300px;
4     text-align: center;
5     padding: 10px;
6     border: 2px solid;
7     border-radius: 20px;
8     -ms-border-radius: 20px;
9     -moz-border-radius: 20px;
10    -o-border-radius: 20px;
11    -webkit-border-radius: 20px;
12  }
13 </style>
14 <div>Treinamento de CSS3 – Impacta Tecnologia</div>
```

Resultado:



É possível também inserir um valor separadamente para cada extremidade. A declaração é feita da seguinte forma: **border-[top/bottom]-[left/right]-radius**

Exemplo:

```
1 <style type="text/css">
2   div{
3     width: 300px;
4     text-align: center;
5     padding: 10px;
6     border: 2px solid;
7     border-top-left-radius: 15px 20px;
8     -moz-border-radius-topleft: 15px 20px;
9     -webkit-border-top-left-radius: 15px 20px;
10    -o-border-top-left-radius: 15px 20px;
11    -ms-border-top-left-radius: 15px 20px;
12  }
13 </style>
14 <div>Treinamento de CSS3 – Impacta Tecnologia</div>
```

Resultado:



## 5.1.5. border-image

É possível também utilizar uma imagem como borda de um elemento da tela. O uso desta propriedade é definido da seguinte forma: **border-image:url(caminho da imagem)**

De acordo com as especificações da W3C, é possível criar uma imagem para ser usada como um estilo de borda. Nesse caso, a borda será construída com porções extraídas dos cantos e das laterais da imagem.

Existe ainda a possibilidade de serem utilizadas as seguintes propriedades:

- border-image-source;
- border-image-slice;
- border-image-width;
- border-image-outset;
- border-image-repeat (valores disponíveis: stretch, repeat, round e space);
- border-image (propriedade abreviada).

Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja como esta propriedade irá funcionar:

Navegador	Sintaxe
Internet Explorer 11+	-ms-border-image
Mozilla Firefox 3.5+	-moz-border-image
Opera 11.00+	-o-border-image
WebKit (Chrome 4+ / Safari 3.1)	-webkit-border-image
W3C Markup	border-image

Da mesma forma que é possível inserir múltiplos backgrounds, é possível também inserir múltiplas imagens em uma propriedade **border-image**.

## 5.1.5.1.border-image-source

Esta propriedade é utilizada para definir o caminho físico no qual a imagem está localizada. Esta propriedade é utilizada da seguinte forma:

```
1 div{  
2     border-image-source: url(imagem.png);  
3     -ms-border-image-source: url(imagem.png);  
4     -moz-border-image-source: url(imagem.png);  
5     -o-border-image-source: url(imagem.png);  
6     -webkit-border-image-source: url(imagem.png);  
7 }
```

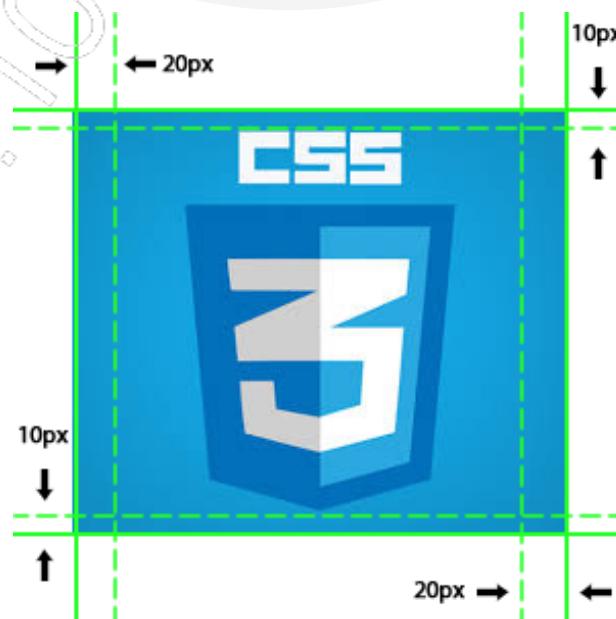
## 5.1.5.2.border-image-slice

Esta propriedade é utilizada para definir um fatiamento na imagem. Esta propriedade é utilizada da seguinte forma:

```
1 div{  
2     border-image-slice: 10 30 10 30;  
3     -ms-border-slice: 10 17 fill; /* A palavra-chave fill é utilizada para inserir na parte central o conteúdo da imagem ao qual se aplicou a borda */  
4     -moz-border-slice: 10% 10% 10% 10%;  
5     -o-border-slice: 10% 20% 20% 10%;  
6     -webkit-border-slice: 1 2 3 4;  
7 }
```

A palavra chave **fill** é utilizada para inserir na parte central o conteúdo da imagem sobre a qual foi aplicada a borda.

Exemplo:



Código:

```
1 <style type="text/css">
2   div{
3     width: 300px;
4     height: 300px;
5     border-image-source: url("img/logo.jpg");
6     -moz-border-image-source: url("img/logo.jpg");
7     -webkit-border-image-source: url("img/logo.jpg");
8     -o-border-image-source: url("img/logo.jpg");
9     -ms-border-image-source: url("img/logo.jpg");
10    border-image-slice: 20 10 fill;
11    -moz-border-image-slice: 20 10 fill;
12    -webkit-border-image-slice: 20 10 fill;
13    -o-border-image-slice: 20 10 fill;
14    -ms-border-image-slice: 20 10 fill;
15  }
16 </style>
17 <div></div>
```

Resultado:



### 5.1.5.3.border-image-width

Esta propriedade é utilizada para definir a espessura da borda que receberá a imagem. A declaração desta propriedade pode ser feita da seguinte forma:

```
1 div{
2   border-image-width: 10px 20px 30px 40px; /* Os valores podem ser em pixel */
3   -ms-border-image-width: 3em 4em 5em; /* Os valores podem ser em medida CSS de comprimento */
4   -moz-border-image-width: 10 20; /* Os valores podem ser em numeros */
5   -o-border-width: 10px 10 10px 10; /* Os valores podem ser em pixel e numeros */
6   -webkit-border-image-width: 10 20;
7 }
```

## 5.1.5.4.border-image-outset

Esta propriedade é utilizada para definir uma espessura fora da área normal de borda sobre a qual se estenderá a imagem:

```
1 div{  
2     border-image-outset: 10px 20px 30px 40px; /* Os valores podem ser em pixel */  
3     -ms-border-image-outset: 3em 4em 5em; /* Os valores podem ser em medida CSS de comprimento */  
4     -moz-border-image-outset: 10 20; /* Os valores podem ser em numeros */  
5     -o-border-image-outset: 10px 10 10px 10; /* Os valores podem ser em pixel e numeros */  
6     -webkit-border-image-outset: 10 20;  
7 }
```

## 5.1.5.5.border-image-repeat

Essa propriedade é utilizada para definir como será dimensionada e distribuída a imagem pelas bordas e pela área de conteúdo. Os valores definidos para esta propriedade são: **stretch**, **repeat**, **round** e **space**.

A declaração desta propriedade é feita da seguinte forma:

```
1 div{  
2     border-image-repeat: repeat; /* A imagem se repete até preencher toda extensão da borda */  
3     -ms-border-image-repeat: stretch; /* Faz com que o imagem seja alongado para preencher toda extensão da borda */  
4     -moz-border-image-repeat: round; /* Tem o mesmo efeito do repeat, com a diferença de preencher o conteúdo  
5                                     em caso de a repetição não resultar um numero inteiro */  
6     -o-border-image-repeat: space; /* Tem o mesmo efeito do repeat, com a diferença de faz uma redistribuição do conteúdo em  
7                                     caso de a repetição não resultar um numero inteiro */  
8     -webkit-border-image-repeat: repeat space; /* É possível nesta propriedade atribuir até dois valores */  
9 }
```

## 5.2. Box shadow

No CSS3, outro recurso muito interessante é o uso de **sombra**s para os elementos gráficos. Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja como esta propriedade irá funcionar:

Navegador	Sintaxe
<b>Internet Explorer 9 ou superior</b>	-ms-box-shadow
<b>Mozilla Firefox 3.6+</b>	-moz-box-shadow
<b>Opera 11.10+</b>	-o-box-shadow
<b>WebKit (Chrome 11+ / Safari 5.1)</b>	-webkit-box-shadow
<b>W3C Markup, IE 10 Release Preview</b>	box-shadow

O uso desta propriedade deve ser:

```

1 div{
2   box-shadow: 10px 20px -15px #CCC;
3   -ms-box-shadow: 10px 20px 15px #CCC;
4   -moz-box-shadow: 10px 20px 15px #CCC;
5   -o-box-shadow: 10px 20px 15px #CCC;
6   -webkit-box-shadow: 10px 20px -15px #CCC;
7 }
```

Onde:

- o primeiro valor define o deslocamento horizontal da sombra, podendo ser um valor positivo ou negativo;
- o segundo valor define o deslocamento vertical da sombra, podendo ser um valor positivo ou negativo;
- o terceiro valor define a espessura do esfumaçamento;
- o quarto valor define a cor do esfumaçamento.

# CSS3

A seguir, uma aplicação prática da utilização do **box shadow**, que neste exemplo terá declarado apenas **box-shadow**:

```
1 <style type="text/css">
2   div{
3     width: 100px;
4     height: 100px;
5     margin: 20px;
6     float: left;
7   }
8   .box_shadow_1{
9     box-shadow: 10px 10px 2px #CCC;
10  }
11  .box_shadow_2{
12    box-shadow: 10px -10px 20px red;
13  }
14  .box_shadow_3{
15    box-shadow: -2px -2px 12px 2px yellow;
16  }
17  .box_shadow_4{
18    box-shadow: 10px 10px 50px 20px green inset; /* A palavra chave inset define uma sombra interna ao elemento */
19  }
20 </style>
21 <div class="box_shadow_1">Sombra 1</div>
22 <div class="box_shadow_2">Sombra 2</div>
23 <div class="box_shadow_3">Sombra 3</div>
24 <div class="box_shadow_4">Sombra 4</div>
```

Resultado:



## 5.2.1. Text shadow

Esta propriedade é utilizada para definir uma sombra em textos. O uso é semelhante ao do **box shadow**, com exceção do uso da palavra-chave **inset**, pois nessa propriedade não existe essa definição.

Para que o navegador reconheça este efeito, será necessária a utilização de filtros e declarações diferentes. Veja como esta propriedade irá funcionar:

Navegador	Sintaxe
Internet Explorer 10 ou superior	-ms-text-shadow
Mozilla Firefox 21 ou superior	-moz-text-shadow
Opera 16 ou superior	-o-text-shadow
WebKit (Chrome 27+ / Safari 5.1)	-webkit-text-shadow
W3C Markup, IE 10 Release Preview	text-shadow

O uso desta propriedade deve ser feito como segue:

```

1 div{
2   text-shadow: 10px 20px -15px #CCC;
3   -ms-text-shadow: 10px 20px 15px #CCC;
4   -moz-text-shadow: 10px 20px 15px #CCC;
5   -o-text-shadow: 10px 20px 15px #CCC;
6   -webkit-text-shadow: 10px 20px -15px #CCC;
7 }
```

Onde:

- o primeiro valor define o deslocamento horizontal da sombra, podendo ser um valor positivo ou negativo;
- o segundo valor define o deslocamento vertical da sombra, podendo ser um valor positivo ou negativo;
- o terceiro valor define a espessura do esfumaçamento;
- o quarto valor define a cor do esfumaçamento.

A seguir, uma aplicação prática da utilização do **text shadow**, que neste exemplo terá declarado apenas **text-shadow:valores**

```
1 <style type="text/css">
2   div{
3     width: 200px;
4     height: 200px;
5     margin: 20px;
6     font-size: 40px;
7     float: left;
8   }
9   .text_shadow_1{
10     text-shadow: 2px -2px #CCC, 4px -4px #eee, 6px -6px #000;
11   }
12  .text_shadow_2{
13    text-shadow: 0 0 2px red;
14    color:white;
15  }
16  .text_shadow_3{
17    text-shadow: 10px 10px #CCC;
18  }
19  .text_shadow_4{
20    text-shadow:-10px -10px green; /* A palavra chave inset define uma sombra interna ao elemento */
21  }
22</style>
23<div class="text_shadow_1">IMPACTA</div>
24<div class="text_shadow_2">IMPACTA</div>
25<div class="text_shadow_3">IMPACTA</div>
26<div class="text_shadow_4">IMPACTA</div>
```

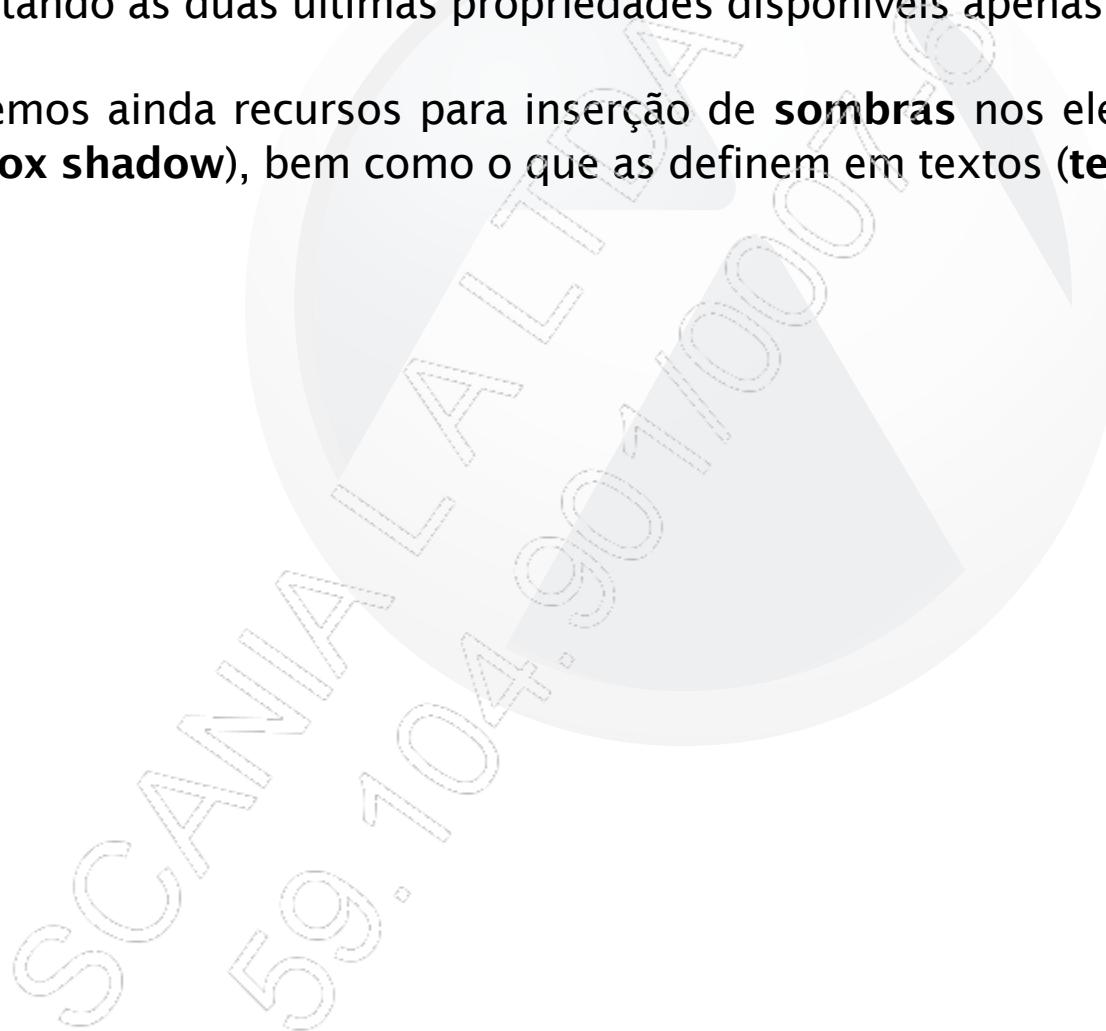
Resultado:



## Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- Podemos definir estilos de molduras, mais conhecidas como **bordas**, para um elemento, onde suas quatro faces podem ter ou um estilo uniforme ou um estilo aplicado a cada uma delas individualmente;
- As propriedades dessa função permitem definir espessura (**border-width**), estilo (**border-style**), cor (**border-color**), arredondamento (**border-radius**) ou mesmo ter uma imagem adicionada como borda, (**border-image**), estando as duas últimas propriedades disponíveis apenas no CSS3;
- Temos ainda recursos para inserção de **sombras** nos elementos gráficos (**box shadow**), bem como o que as definem em textos (**text shadow**).





**5**

# **Bordas e sombras**

## **Teste seus conhecimentos**

SCANIA TDA  
59.704.907-67



**IMPACTA**  
EDITORA

**1. Qual dos valores a seguir não é possível de ser inserido como estilo em uma propriedade border-style?**

- a) solid
- b) double
- c) dotted
- d) shadow
- e) inset

**2. Qual a função da propriedade border-image-slice?**

- a) Definir um fatiamento na imagem.
- b) Definir um espaçamento na imagem.
- c) Preencher todo o elemento e deixar as bordas fixas.
- d) Definir a absoluta relativa da imagem com relação ao elemento.
- e) Nenhuma das alternativas anteriores está correta.

3. Qual a palavra-chave que pode ser utilizada como valor na propriedade box-shadow?

- a) outset
- b) inset
- c) fill
- d) x-repeat
- e) Nenhuma das alternativas anteriores está correta.

4. Quais valores estão definidos na propriedade border-image-repeat?

- a) stretch, repeat, round e space
- b) stretch, norepeat, round e space
- c) stretch, norepeat, round e none
- d) stretch, repeat, round e spaced
- e) Nenhuma das alternativas anteriores está correta.



**5**

# **Bordas e sombras**

## **Mãos à obra!**

SCANIA TDA  
59.704.907-67



**IMPACTA**  
EDITORA

# Laboratório 1

## A - Criando bordas e sombreamento

1. Abra na pasta projeto os arquivos **smartphone.css**, **tablet.css**, **desktop.css**, e adicione as declarações CSS conforme a tabela a seguir:

Seletores	Propriedade	Valor
#logo-css3	border	dashed 3px #000;
	position	relative
	float	left
input[type='search']	background-color	#eee
	border	None
	border-radius	15px
#button-lupa	border	none
.descricao	background-color	#e0e0e0
	box-shadow	10px 10px 10px #888
#secoes	border	Solid 1px #e0e0e0
	border-radius	10px
#para-voce	border-radius	10px

# Trabalhando com fontes

6

- ✓ Fontes;
- ✓ Propriedades para fontes;
- ✓ Tamanho e formatações;
- ✓ A regra @font-face;
- ✓ Efeito em fontes;



**IMPACTA**  
EDITORA

## 6.1. Fontes

Um dos grandes desafios para o desenvolvimento web no decorrer dos anos, tem sido a limitação no uso de fontes para textos. Essa limitação ocorre devido ao fato de os navegadores não possuírem suporte aos diversos tipos de fontes para texto. Designers utilizam as inúmeras possibilidades quando criam layouts completos com as mais variadas famílias de fontes. Entretanto, para refletir essa criação no projeto final em HTML, era necessário criar imagens com as diferentes fontes desejadas.

Com a CSS3 temos a possibilidade de carregar fontes externas que não são nativas do navegador ou do sistema operacional do usuário. Neste capítulo aprenderemos sobre as declarações para o uso de fontes, formatação de estilo de fontes, a regra `@font-face`, restrições para fontes pagas e efeitos em fontes.

## 6.2. Propriedades para fontes

Para determinar a fonte a ser utilizada no documento, utilizamos a propriedade **font-family**, com os seguintes valores:

- Nome da fonte;
- Família da fonte.

Exemplo:

```
font-family: arial, sans-serif;
```

As fontes podem ser com serifa, como a **Times New Roman**, onde os traços são detalhados nas extremidades, ou podem ser sem serifa - do francês **sans-serif**, que significa que a fonte tem um traço mais reto sem os detalhes nas extremidades. Temos então a **ARIAL**, que é um exemplo de fonte sem serifa.

## 6.3. Tamanho e formatações

Para determinar o tamanho da fonte, utilizamos a propriedade **font-size**, que possui os seguintes valores:

Valor da propriedade font-size	Medida	Descrição
<b>xx-small</b>	9 pixels	Fonte do tamanho mínimo legível
<b>x-small</b>	10 pixels	Fonte de tamanho pequeno legível
<b>small ou smaller</b>	12-13px	Fonte de tamanho padrão
<b>medium</b>	16px	Fonte de tamanho médio
<b>large</b>	18px	Fonte de tamanho acima da média
<b>larger</b>	19px	Fonte grande
<b>x-large</b>	24px	Fonte utilizada para grande destaque
<b>xx-large</b>	32px	Fonte grande para destaque sem significados semânticos
<b>%</b>	0% - 1000%. Abaixo de 100% reduz o tamanho, acima de 100% aumenta.	Tamanho em porcentagem, onde 100% é o tamanho padrão
<b>px, pt, em, cm</b>	Especificado em números.	Unidades reconhecidas pela CSS, onde pixel é o mais utilizado

É possível determinar o peso da fonte, se a fonte terá um comportamento normal ou mais espesso, e quanto mais espessa ela deve ser. Determinamos isso pela propriedade **font-weight**:

Valor da propriedade <b>font-weight</b>	Descrição
<b>normal</b>	Fonte normal sem alterações
<b>bold</b>	Negrito
<b>bolder</b>	Equivalente ao negrito, mas determina uma fonte mais espessa
<b>100-900</b>	Intensidade do negrito, embora a diferença maior seja percebida a partir de 400 e 700. Abaixo ou acima desse valor, o resultado apresentado será equiparado aos limites entre estes valores. Por exemplo: o valor 400 apresentará basicamente o mesmo resultado que o valor 100, porém algumas fontes permitem observar esta diferença.

A propriedade **font-style** determina um estilo para a formatação de um texto, sendo que esses valores podem ser:

Valor da propriedade <b>font-style</b>	Descrição
<b>normal</b>	Fonte normal sem alterações
<b>italic</b>	Formata a fonte como itálico, quando esta possui uma versão em itálico, porém quando a fonte não possuir o padrão itálico, seu comportamento se assemelha ao valor oblique
<b>oblique</b>	Inclina a fonte tornando-a semelhante ao itálico, então neste caso este valor irá inclinar a fonte original não necessariamente carregando sua versão em itálico

## 6.4. A regra @font-face

Uma das regras mais importantes da CSS3, a regra `@font-face` permite carregar uma fonte diferente das que estão instaladas por padrão no navegador do usuário. No entanto, para isso é necessário possuirmos o arquivo da fonte que desejamos utilizar. Podemos trabalhar com vários formatos de fontes, e ainda otimizar o desempenho do uso de fontes externas.

Para otimizar o uso de fontes, podemos definir que o navegador deverá buscar a fonte em questão na máquina do usuário, e caso não a encontre, poderá carregá-la remotamente.

Veremos agora os tipos de fontes que podemos carregar, e o suporte de cada navegador a estes tipos. Veremos também como determinar a prioridade de carregamento da fonte, local ou remota.

### 6.4.1. Tipos de fontes

Os vários formatos de fontes permitem a sua utilização em praticamente todos os navegadores modernos, alguns suportam um padrão mas não outro, e alguns suportam vários padrões. Na tabela abaixo encontramos os formatos de fontes disponíveis e os navegadores que suportam tais extensões.

Formato	Ext.	Suporte
Embedded Open Type (EOT)	.eot	Internet Explorer em todas as versões.
TrueType & Open Type	.ttf ; .otf	IE 9+, Firefox, Chrome, Opera, Safari, iOS Safari 4.2+, Android 2.2+
Web Open Font Format	.woff	IE 9+, Firefox, Chrome, Opera, Safari, iOS Safari 5.0
Scalable Vector Graphics (SVG)	.svg	Chrome, Safari, Opera, iOS Safari, Android 3.0+

# CSS3

---

Vejamos um exemplo do uso de fontes tradicionais com as suas respectivas formatações:

- Arquivo: cap6\_01.html

```
1 <!doctype html>
2 <html lang="pt-BR">
3 <head>
4     <meta charset="UTF-8">
5     <title>Trabalhando com Fontes</title>
6     <link rel="stylesheet" href="css/estilo6_01.css">
7 </head>
8 <body>
9     <main>
10         <header>
11             <h1>Utilizando Fontes</h1>
12         </header>
13         <p>
14             Formatando um parágrafo.
15             <strong>Trecho em Negrito pela marcação</strong>
16         </p>
17         <p>
18             Formatando outro parágrafo.
19             <span class="negrito">Trecho em Negrito pelo css</span>
20         </p>
21     </main>
22 </body>
23 </html>
```

- Arquivo: estilo6\_01.css

```
1 @charset "UTF-8";
2
3 body{
4     font-family: arial, sans-serif;
5 }
6 p{
7     font-size: 15px;
8     font-style: oblique;
9 }
10
11 .negrito{
12     font-weight: bold;
13 }
```

Veja o resultado:

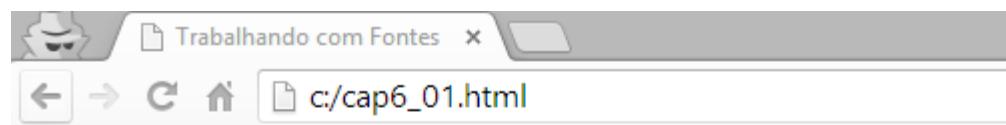


## 6.4.2. Usando fontes locais

A regra `@font-face` permite determinar uma fonte externa a ser utilizada pelo navegador. Entretanto, suponhamos que o usuário que acessa um site já possua esta fonte instalada em seu sistema operacional, e por extensão em seu navegador. Neste caso, ao invés de realizar o download da fonte e sua renderização, podemos utilizar a fonte local. Vejamos um exemplo com a fonte Calibri:

```
1 @charset "UTF-8";
2
3 @font-face{
4     font-family: 'calibri';
5     src:local('calibri');
6 }
7 body{
8     font-family: calibri, sans-serif;
9 }
10 p{
11     font-size: 15px;
12     font-style: oblique;
13 }
14
15 .negrito{
16     font-weight: bold;
17 }
```

Produzindo o seguinte resultado:



## Utilizando Fontes

*Formatando um parágrafo. Trecho em Negrito pela marcação*

*Formatando outro parágrafo. Trecho em Negrito pelo css*

Neste caso, o uso da propriedade **@font-face** poderia ser omitido, uma vez que a fonte já está instalada no sistema operacional.

### 6.4.3. Usando fontes remotas

Caso a fonte não exista na máquina do usuário, teremos que fazer uma referência à sua localização no servidor web com base no endereço raiz do site. Assim, utilizamos um caminho relativo até o arquivo da fonte.

- Arquivo: **estilo6\_1.css**

```
1 @charset "UTF-8";
2
3 @font-face{
4     font-family: 'sports-world';
5     src: url('../font/Sports World-Regular.ttf');
6 }
7 body{
8     font-family: calibri, sans-serif;
9 }
10 p{
11     font-size: 15px;
12     font-style: oblique;
13 }
14 h1{
15     font-family: sports-world;
16     font-size: 50px;
17     color: #597800;
18 }
19 .negrito{
20     font-weight: bold;
21 }
```

Veja o resultado:



*Formatando um parágrafo. Trecho em Negrito pela marcação*

*Formatando outro parágrafo. Trecho em Negrito pelo css*

Vejamos mais um exemplo do uso de fontes com o uso de uma imagem de fundo:

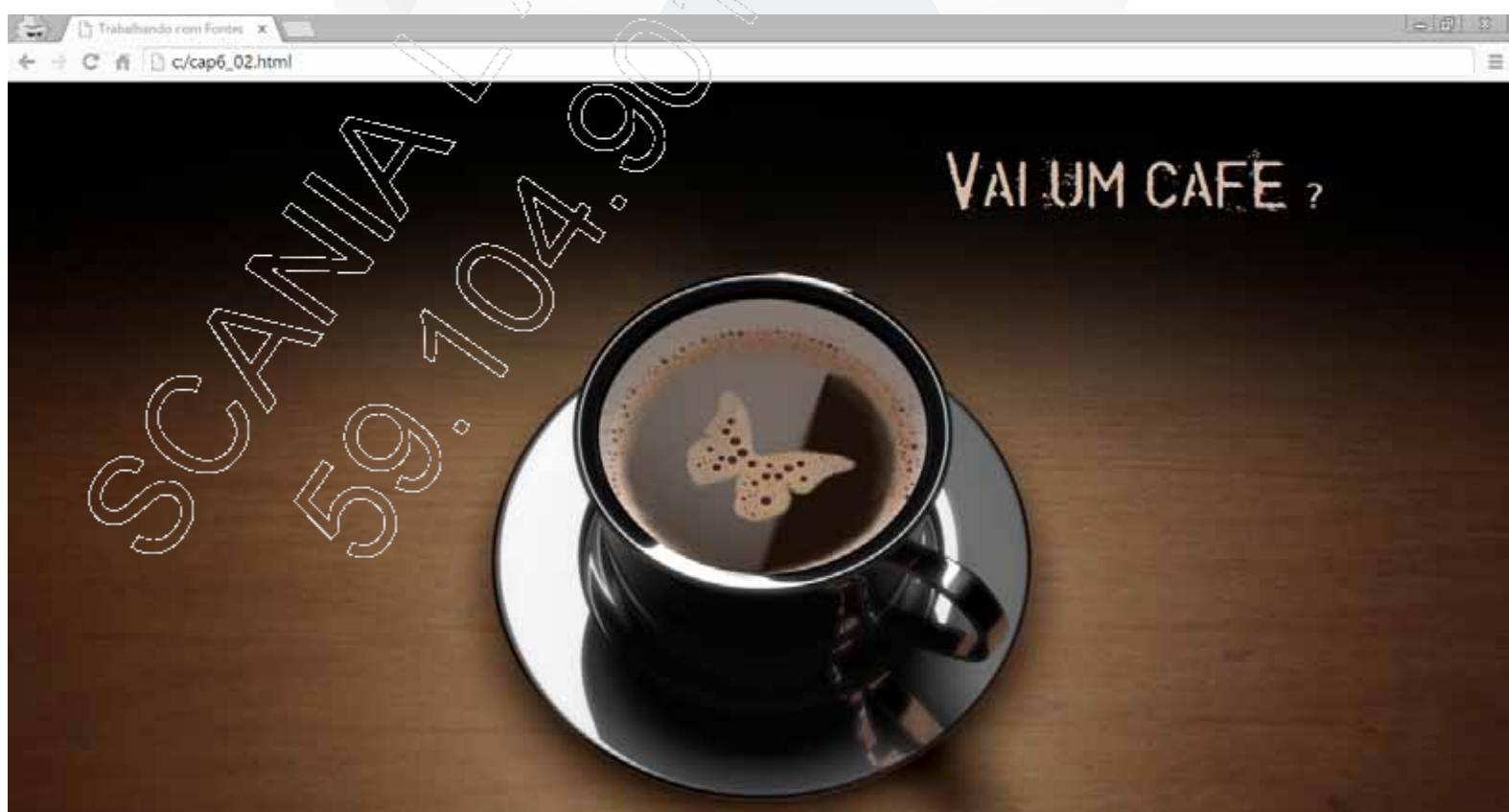
- Arquivo: cap6\_02.html

```
1 <!doctype html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8"> .
5   <title>Trabalhando com Fontes</title>
6   <link rel="stylesheet" href="css/estilo6_02.css">
7 </head>
8 <body>
9   <main>
10    <h1> Vai um cafe ?
11    </h1>
12  </main>
13 </body>
14 </html>
```

- Arquivo: **estilo6\_02.css**

```
1 @charset "UTF-8";
2 @font-face{
3     font-family: 'coldcoffee';
4     src:url('../font/coldcoffee.ttf');
5 }
6 body{
7     background-color: #000000;
8     background: url("../images/coffee-fundo.jpg") no-repeat;
9     background-size: 100%;
10}
11 h1{
12     font-family: coldcoffee;
13     font-size: 30px;
14     color: #CCB39F;
15     position: absolute;
16     left: 850px;    top: 30px;
17 }
```

Veja o resultado:



## 6.4.4. Usando fontes públicas

Com a popularização do uso da regra `@font-face`, algumas empresas criaram repositórios públicos para armazenar fontes gratuitas, que podem ser utilizadas pelos desenvolvedores web. São chamados de rede de distribuição de conteúdo (do inglês Content Distribution Network - CDN). Você também pode chamar uma fonte de algum destes CDNs, em vez de armazená-la em seu servidor web.

### 6.4.4.1. Usando um serviço de hospedagem de fontes

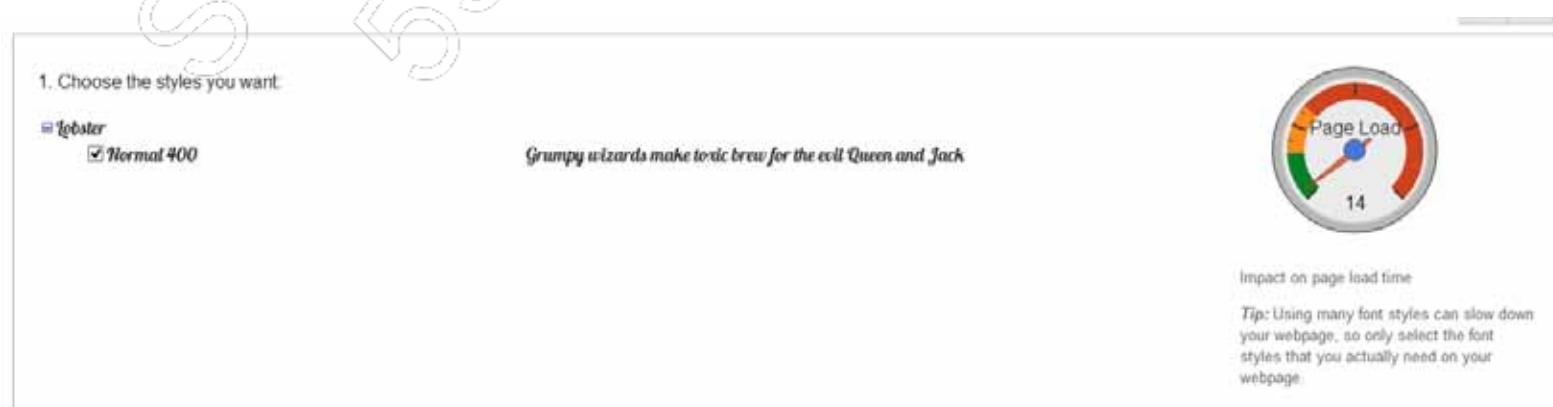
Um dos mais comuns é o [www.google.com/webfonts](http://www.google.com/webfonts). Após acessar o site, basta escolher a fonte desejada e clicar no botão azul “Add to Collection”, localizado no canto direito logo ao lado da fonte escolhida.



Essa fonte será adicionada à sua coleção para download:



Clique então na opção “Use” que aparece no canto inferior direito, e você será redirecionado para a tela de download, onde verá a velocidade de download para a(s) fonte(s) selecionada(s), além das opções para implementá-las em seu código.



Neste caso, o impacto na velocidade do carregamento da página será mínimo, porém quando você seleciona muitas fontes, e fontes com muitos recursos (itálico, regular, negrito) pode haver impacto no desempenho de carregamento da página.

Para aplicar o estilo, temos as opções a seguir:

- Incluir o elemento **link** fazendo uma referência ao site do Google Fonts



The screenshot shows the Google Fonts API interface. At the top, there are tabs for "Standard", "@import", and "Javascript". Below that, a section titled "3. Add this code to your website" contains the CSS link element: <link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet' type='text/css'>. To the right of the code, there is an "Instructions" box stating: "To embed your Collection into your web page, copy the code as the first element in the <head> of your HTML document." Below the instructions is a link "» See an example".

```
<link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet' type='text/css'>
```

- Importar o arquivo CSS, dentro de seu arquivo de folha de estilos:

```
@import url(http://fonts.googleapis.com/css?family=Lobster);
```

- Criar o elemento de referência da CSS via JavaScript:

Naturalmente a referência via JavaScript consome mais processamento, e sua utilização precisa ser avaliada com critério.

```
<script type="text/javascript">
  WebFontConfig = {
    google: { families: [ 'Lobster::latin' ] }
  };
  (function() {
    var wf = document.createElement('script');
    wf.src = ('https:' == document.location.protocol ? 'https' :
      'http') +
      '://ajax.googleapis.com/ajax/libs/webfont/1/webfont.js';
    wf.type = 'text/javascript';
    wf.async = 'true';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(wf, s);
  })(); </script>
```

Vejamos um exemplo com a segunda implementação, utilizando a regra **@import**:

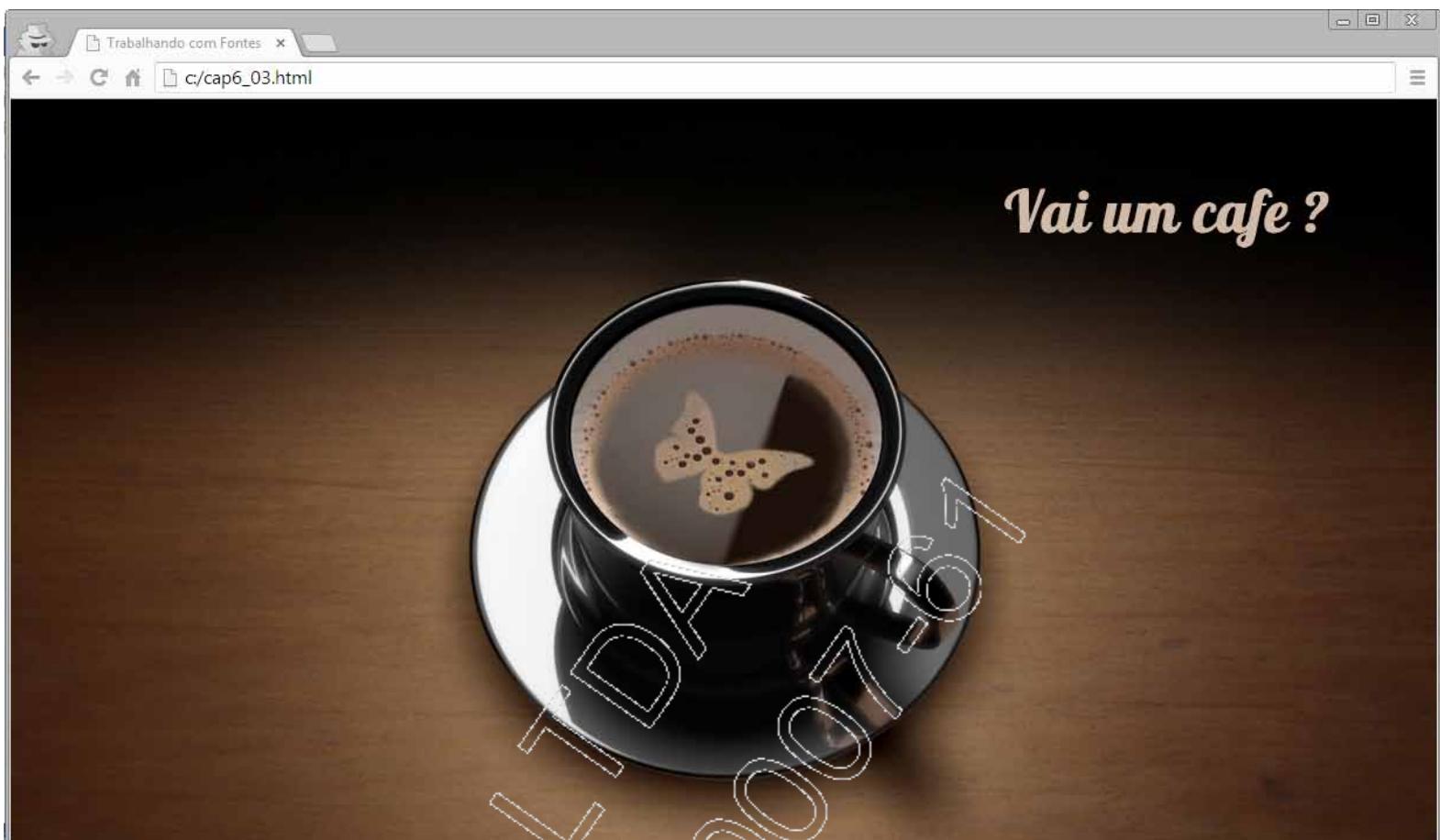
- Arquivo: **estilo6\_03.css**

```
1 @charset "UTF-8";
2
3 @import url('http://fonts.googleapis.com/css?family=Lobster');
4
5 body{
6     background-color: #000000;
7     background: url("../images/coffee-fundo.jpg") no-repeat;
8     background-size: 100%;
9 }
10 h1{
11     font-family: Lobster;
12     font-size: 50px;
13     color: #CCB39F;
14     position: absolute;
15     left: 850px;    top: 30px;
16 }
```

- Arquivo: **cap6\_03.html**

```
1 <!doctype html>
2 <html lang="pt-BR">
3 <head>
4     <meta charset="UTF-8">
5     <title>Trabalhando com Fontes</title>
6     <link rel="stylesheet" href="css/estilo6_03.css">
7 </head>
8 <body>
9     <main>
10        <h1>
11            Vai um cafe ?
12        </h1>
13    </main>
14 </body>
15 </html>
```

Veja o resultado:



## 6.4.5. Restrições com fontes pagas

Utilizar fontes públicas como as do servidor em [google.com/webfonts](http://google.com/webfonts) é mais vantajoso do que hospedá-las em seu próprio servidor. Muitas dessas fontes são pagas ou são gratuitas para uso pessoal, porém colocá-las em seu servidor web poderá estar em desacordo com os termos de licença para uso das fontes, pois seu servidor pode funcionar como hospedeiro para download ilegal de fontes.

Verifique a licença da fonte antes de sugerí-la a um cliente ou utilizá-la em seu projeto. Nos servidores públicos existem inúmeras opções úteis e interessantes para serem utilizadas. Caso esteja utilizando seu site em um ambiente interno da empresa, tal como uma intranet, você pode ter um leque maior de possibilidades restringindo o acesso dos usuários diretamente à pasta de fontes. Esse mecanismo não impede o download da fonte, o que ocorre a partir do momento que a aplicação web a solicita, mas dificulta o acesso direto a elas por parte do usuário.

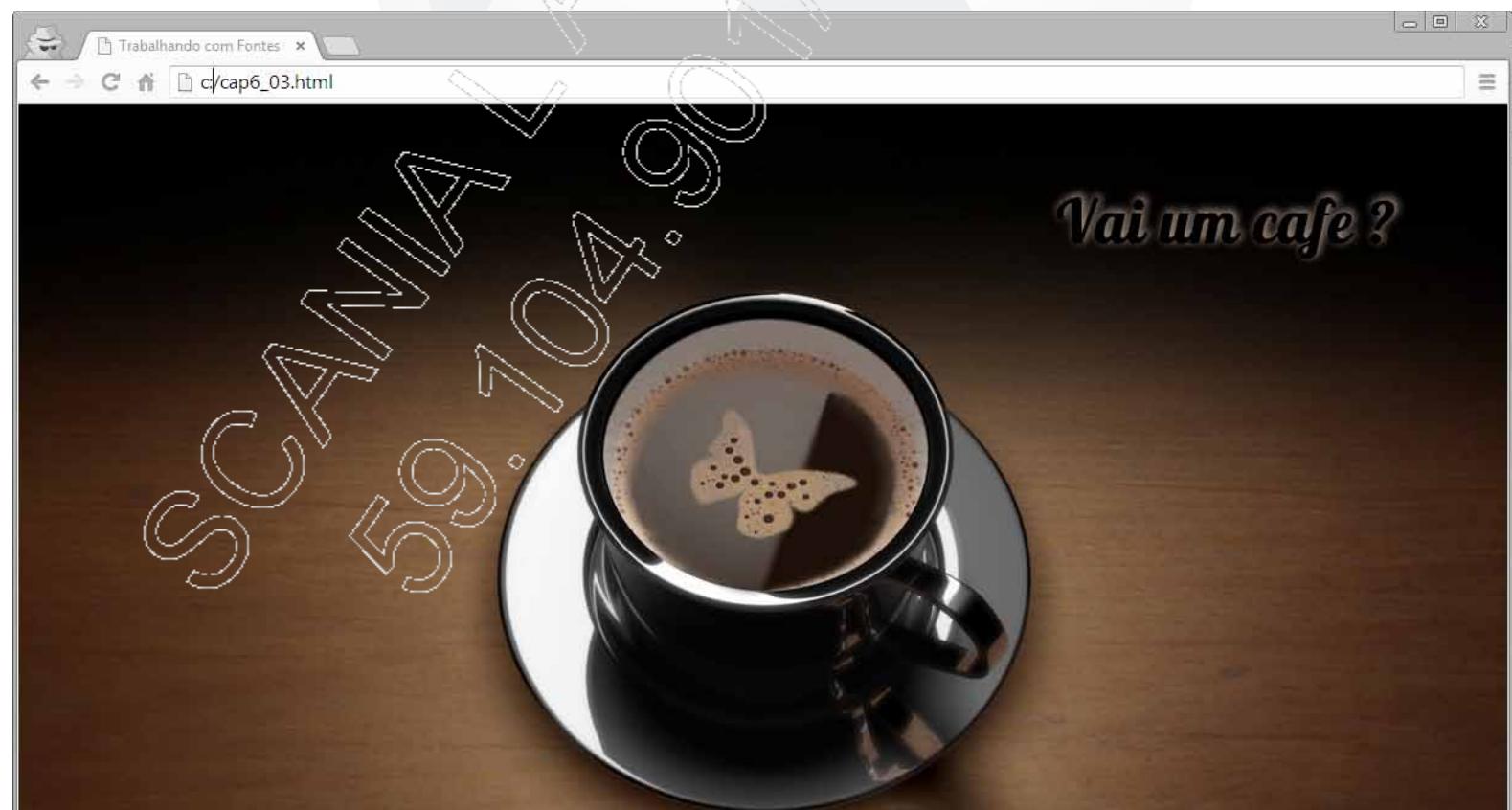
## 6.5. Efeito em fontes

Uma das principais propriedades para colocar efeitos no texto, é a propriedade **text-shadow**, que já foi explicada no capítulo 5.

Veja o exemplo anterior com uma sombra no formato **blur**.

```
10 h1{  
11     font-family: Lobster;  
12     font-size: 50px;  
13     color: none;  
14     position: absolute;  
15     left: 850px;    top: 30px;  
16     text-shadow: 2px -2px 10px #AE957E;  
17 }
```

Veja o resultado:



# Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- A propriedade **font-family** é responsável por determinar um tipo de fonte para o seletor em questão;
- A propriedade **font-weight** é responsável por determinar um tipo de espessura para a fonte em questão;
- A propriedade **font-size** determina o tamanho da fonte, podendo ser medida em pixels;
- A **@font-face** é uma regra que permite carregar uma fonte diferente do padrão do sistema operacional, podendo ser local ou remota com o atributo **src**;
- Podemos utilizar fontes públicas para não termos que armazenar o arquivo de fontes em nosso servidor web. Além disso, levar em consideração o tipo de licença da fonte ao utilizá-la é uma preocupação necessária.

6

# Trabalhando com fontes

## Teste seus conhecimentos

SCANDA  
50.  
TDA  
907-67



**IMPACTA**  
EDITORA

**1. Qual declaração CSS precisamos utilizar na regra @font-face para determinar o caminho relativo de uma fonte em seu servidor web?**

- a) local e url
- b) src e local
- c) @charset e local
- d) src e url
- e) @font-face

**2. Qual regra utilizamos para determinar o uso de uma fonte diferente do padrão do navegador?**

- a) @important
- b) @charset
- c) @font-face ou @import
- d) @font-face ou @charset
- e) @import ou @charset

### 3. Qual regra e atributo devemos utilizar para carregar uma fonte chamada “courier” localizada na máquina do usuário?

- a) @font-face {font-family: "courier"; src: url('courier');}
- b) @font-face {font-family: "courier"; src: local('courier')}
- c) @font-style {font-family: "courier"; src: local('courier')}
- d) @font-name {font-family: "courier"; src: url('courier')}
- e) @font-name {font-family: "courier"; src: local('courier')}

### 4. Para carregar uma fonte chamada Kite One de um servidor do Google Web Fonts, como podemos prosseguir?

- a) @import url(http://fonts.googleapis.com/css?family=Kite+One);
- b) @import local(http://fonts.googleapis.com/css?family=Kite+One);
- c) @import url(fonts://fonts.googleapis.com/css?family=Kite+One);
- d) @charset url(http://fonts.googleapis.com/css?family=Kite+One);
- e) @import local(fonts.googleapis.com/css?family=Kite+One);



6

# Trabalhando com fontes

## Mãos à obra!

SCANIA 114.000-67  
59.704.901  
SCANIA 114.000-67



**IMPACTA**  
EDITORA

## Laboratório 1

### A - Importando uma fonte

1. Na pasta projeto, abra a pasta **css**, e dentro de cada um dos três arquivos (**desktop.css**, **tablet.css** e **smartphone.css**) adicione no início do arquivo a importação da fonte **Open Sans** diretamente do Google Fonts.
2. Adicione ao seletor **body** a declaração **font-family: 'Open Sans', sans-serif**.



# Produzindo um layout responsivo

7

- ✓ Etapas da criação de um layout;
- ✓ Web design responsivo.

SCANIA LTDA  
59.704.900/10007-67



**IMPACTA**  
EDITORA

## 7.1. Introdução

Após conhecer as propriedades e valores que formam as declarações CSS, vamos considerar como planejar, projetar e criar um layout responsivo, que seja adaptável a vários dispositivos que o usuário esteja utilizando.

Pensar de modo responsivo no desenvolvimento web é uma constante, e ajuda a criar sites e aplicações web simples e limpas.

Neste capítulo aprenderemos a planejar um layout desde a sua concepção e ideia inicial, passaremos pelo wireframe e os diversos tipos de dispositivos que o mercado oferece tais como desktop, tablet e smartphone. Entenderemos o conceito de media queries e como aplicá-los em nosso projeto.

## 7.2. Etapas da criação de um layout

Para iniciar a criação de um layout, seja ele para um website ou para um web app, é necessária a realização de alguns passos fundamentais, que vão desde a reunião com o cliente que patrocina o projeto, com o levantamento das necessidades e análise de requisitos, até rascunhos em papel para criar um esboço do que será desenvolvido. A seguir, colocamos alguns passos que devem existir num projeto de layout bem elaborado: não são regras rígidas, mas que irão ajudar a criar um layout profissional e que enriqueça a experiência do usuário.

## 7.2.1. Briefing

O primeiro passo é montar um briefing, ou um roteiro com o conteúdo que desejamos criar, e é neste momento que precisamos saber que tipo de informações o layout irá possuir. É um produto ou serviço? Será um comércio eletrônico ou outro tipo de conteúdo? Para exemplificar os passos que teremos a partir de agora, iremos utilizar como exemplo um layout para um site ou web app que exiba a previsão do tempo.

Neste caso algumas perguntas que ajudariam a montar o briefing:

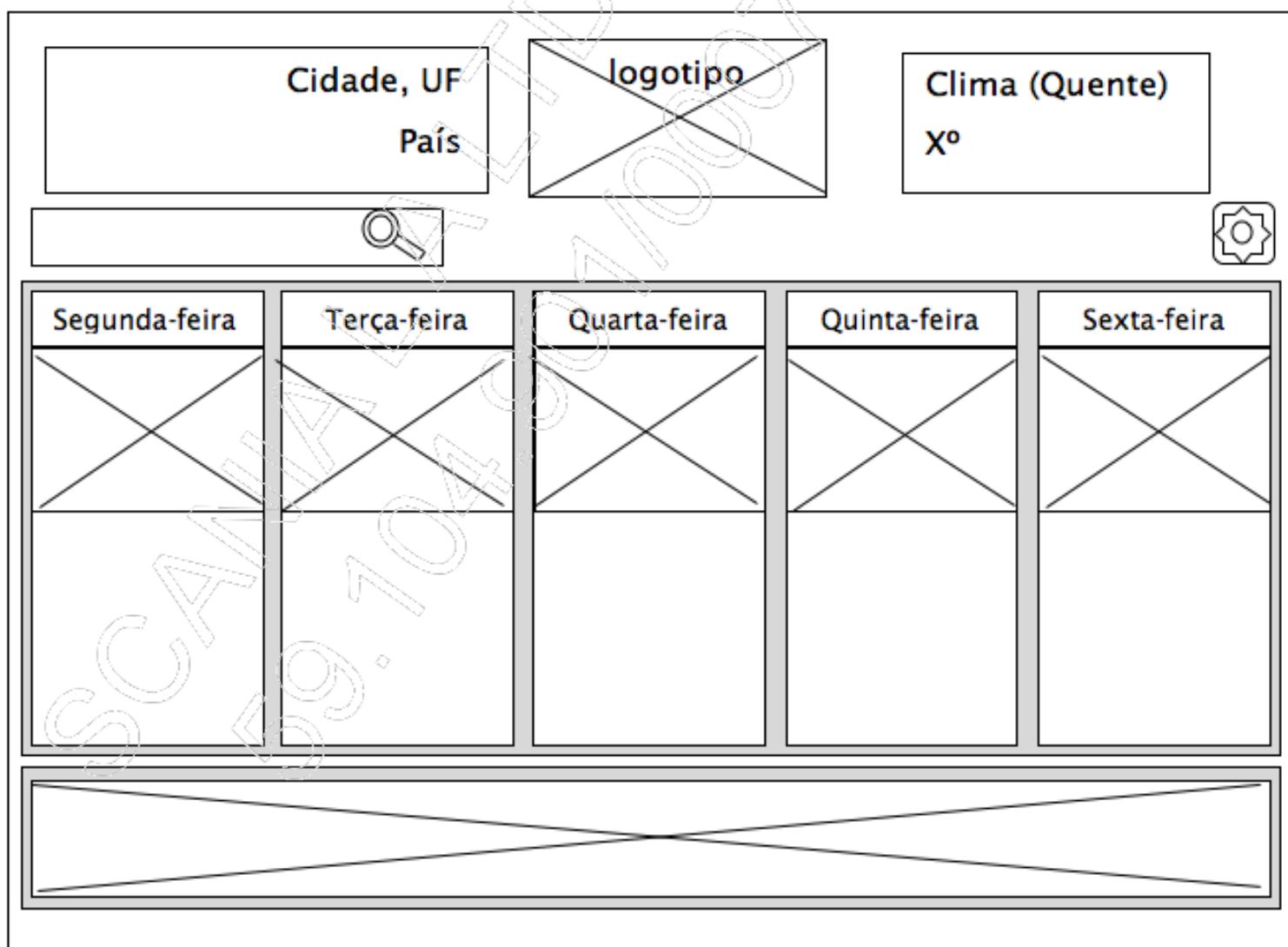
Pergunta	Resposta
Quais informações não podem faltar?	Nomes da cidade, do estado (ou da província) e do país, temperatura em graus Celsius e um logotipo que simbolize o clima (por exemplo, um sol para um clima quente), além de uma área para configurações.
Será informada a previsão para outros dias?	Sim, pelo menos para mais cinco dias.
Será possível realizar pesquisas para outras localidades?	Sim, a busca deve ficar em destaque.
Haverá publicidade?	Sim, discretamente.

Criar perguntas objetivas para montar o briefing é essencial para ir ao próximo passo, pois caso perguntas importantes não sejam feitas, é possível que o layout fique comprometido, e muitos ajustes terão de ser realizados.

## 7.2.2. Wireframe

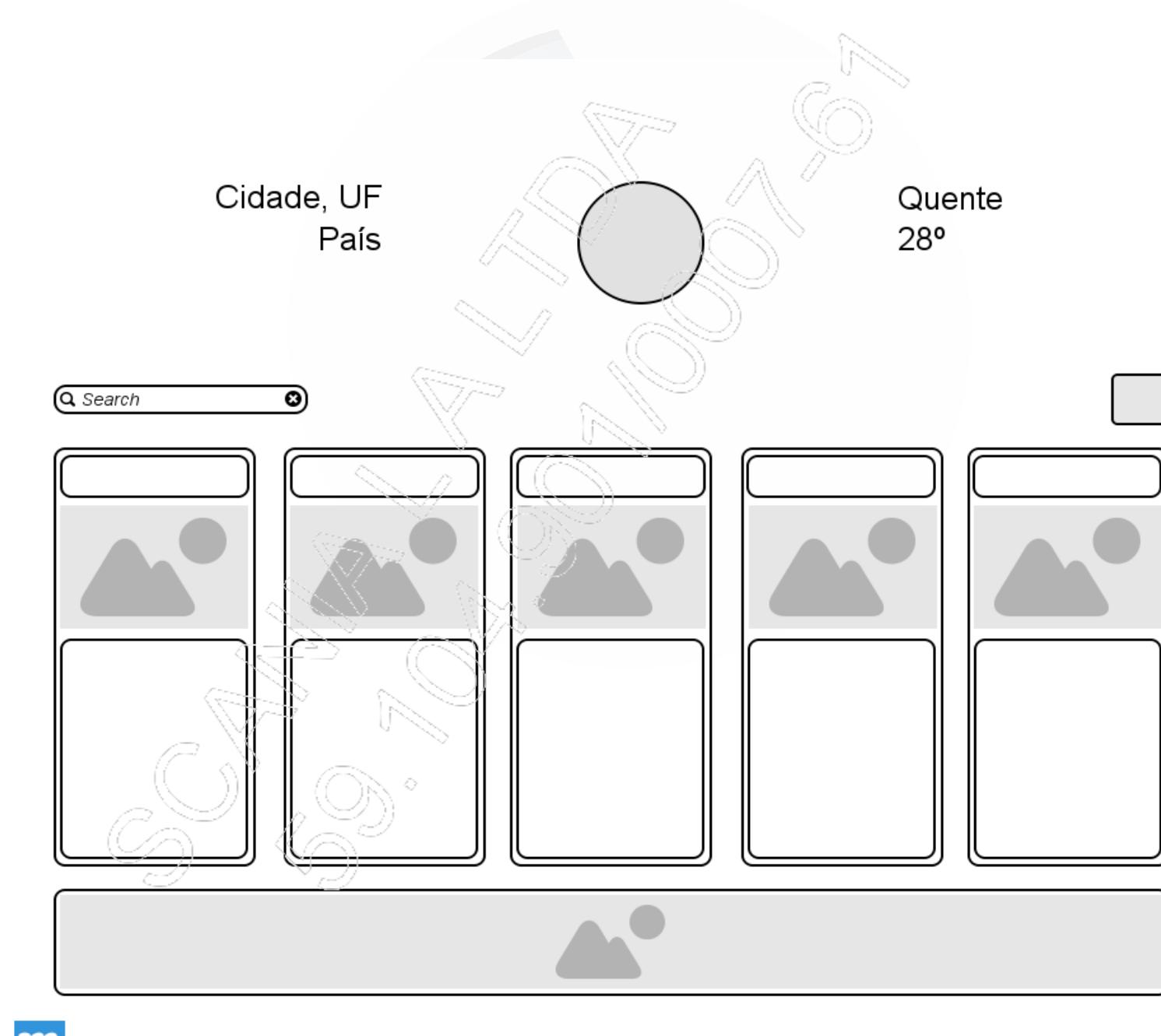
Após a criação de um briefing que responda às principais perguntas para a produção do layout, o próximo passo é criar um guia visual básico, sem muitos detalhes, para mostrar a estrutura do site. Com este guia conseguimos ter uma ideia de onde deverão ficar imagens, textos, logotipo e a disposição das informações, bem como a interação de uma página com outra. Damos a esse esboço o nome de **wireframe** ou **web wireframe**.

Vejamos um modelo de um wireframe baseado no briefing criado:



## Produzindo um layout responsivo

Agora, se utilizamos uma ferramenta para produzir wireframes, podemos deixar o layout com um aspecto mais profissional. O exemplo a seguir utiliza uma ferramenta online, a **Moqups** ([moqups.com](http://moqups.com)), muito útil para visualizar os wireframes.



Online wireframes and prototypes powered by Moqups. Visit us at <http://moqups.com>

## 7.2.3. Dimensões

Neste ponto precisamos pensar no tipo de dispositivo no qual nosso site ou web app irá rodar, uma vez que existem inúmeros tipos à disposição do usuário, e que atualmente a quantidade de acessos à internet por dispositivos móveis já superou o acesso por meio do desktop ou do notebook.

Veremos a partir de agora o que significa **viewport**, quais as dimensões dos principais dispositivos, além de protótipo de seus tamanhos para a criação de layout.

## 7.2.4. Viewport

As dimensões de resolução da CSS são medidas de uma forma diferente da resolução física do aparelho. A isso chamamos de **css pixels**.

Para adequar o conteúdo a vários dispositivos, utilizamos como referência a tag meta com o **name="viewport"**, que determina a área visível do conteúdo em várias dimensões diferentes.

Alguns sites que acessamos por meio do dispositivo móvel acabam mostrando um layout semelhante ao desktop, só que em miniatura, exigindo que o usuário tenha que dar zoom no dispositivo para exibir o conteúdo de uma forma mais legível. Este problema pode ser minimizado utilizando uma tag meta com o atributo **name="viewport"** conforme o exemplo a seguir:

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

A tag meta viewport possui também o atributo **content**, que recebe a largura do viewport e a escala de zoom. Quando atribuímos o valor **device-width** para a propriedade width, determinamos que ocupe a largura do dispositivo, e a propriedade **initial-scale=1** permite que o tamanho do conteúdo seja exibido no tamanho normal em 100%. Para termos uma ideia, quando colocamos o **initial-scale=1.5** o conteúdo será exibido com 50% de aumento de zoom, seja ele texto ou imagem.

O viewport utiliza a medida em css pixels, facilitando o trabalho dos desenvolvedores, uma vez que os dispositivos móveis possuem inúmeros tipos de resolução, e projetar para o viewport resolve este problema. Quando temos um aparelho de alta resolução como o iPhone 5, por exemplo, que possui resolução de 640pixels de largura física no modo retrato, ele irá exibir em 320px no viewport como todos os outros iPhones.

## 7.2.4.1.A regra @viewport

Com a nova regra **@viewport**, podemos transferir o controle do viewport, que fica na meta tag, diretamente para a folha de estilo. E assim como a meta tag, também utilizamos aqui a propriedade **width: device-width**.

```
1 @charset "UTF-8";
2
3 @viewport {
4   width: device-width;
5   zoom: 1;
6 }
```

A largura está utilizando a largura disponível do dispositivo. E **zoom: 1** permite utilizar a escala normal de zoom. Caso mudarmos para **zoom: 2** haverá então um aumento de 100% no tamanho do viewport.

### 7.3. Web design responsivo

Desde smartphones, passando por tablets, notebooks e desktops, os usuários acessam sites por meio dos mais variados dispositivos e navegadores. Pensando em como criar um design para web que se adaptasse aos mais diferentes tipos de dispositivos, Ethan Marcotte utilizou em 2010 o termo **Responsive Web Design (RWD)** em um artigo notório no site [www.alistapart.com](http://www.alistapart.com), para designar a criação de sites que sejam adaptáveis aos mais diversos dispositivos. Desde então o Responsive Web Design, ou simplesmente design responsivo, tem estado na ordem do dia da área.

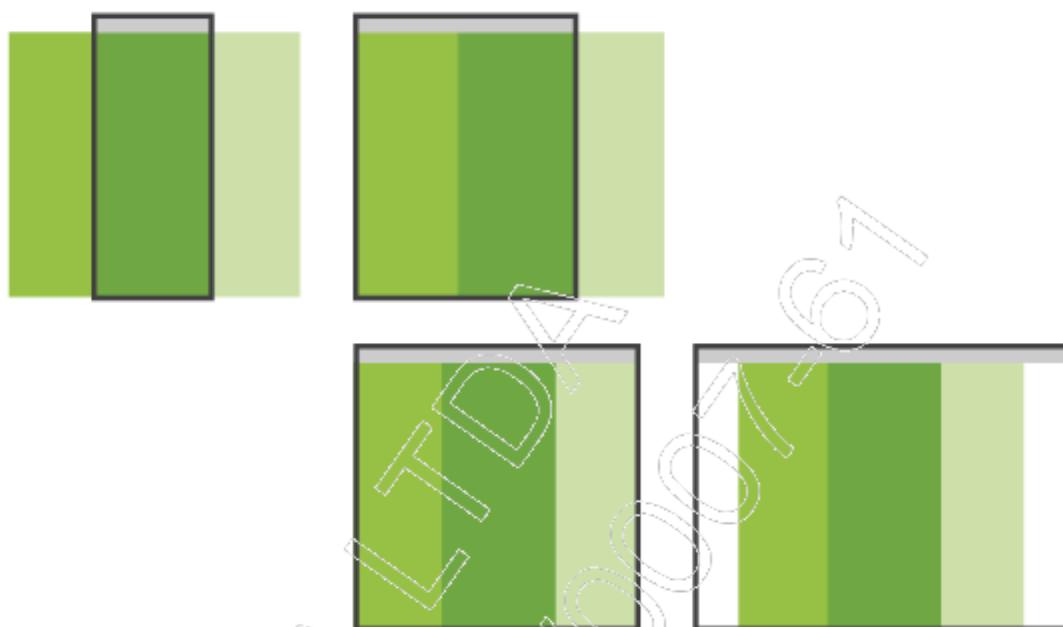
Para que um site seja responsivo, ele precisa ser desenvolvido levando-se em conta um conjunto de fatores, que separados são apenas recursos úteis, porém juntos realizam verdadeiras obras de arte.

A seguir, alguns fatores que ajudam a produzir layout de forma responsiva.

- **Grades fluidas (Fluid grid):** A capacidade de projetar os elementos com dimensões relativas, como por exemplo % ao invés de utilizar tamanhos fixos, além de permitir que quando o viewport é menor as informações que eram maiores fiquem empilhadas como se fossem uma única coluna.



Ou também podemos usar espaços que não são observados na tela para projetar nossa estrutura, dessa forma ganhamos um espaço que o usuário não está visualizando, e sem atrapalhar a sua experiência.



Um exemplo dessa estrutura é o site do [facebook.com](http://facebook.com) para celular smartphone:



- **Imagens flexíveis:** são as que também têm suas dimensões em unidades relativas, de modo a prevenir que escapem de dentro de seu elemento HTML;
- **Media queries:** permitem que a página use diferentes regras de estilo CSS com base nas características do dispositivo onde o website está sendo exibido, sendo mais comum a largura de visualização do navegador web.

Vejamos como trabalhar com a regra **@media** para criar inúmeros tipos de media queries e tornar nosso site responsivo.

## 7.3.1. Media queries

A regra **@media** permite que possamos aplicar estilos e regras com algumas condições, e possui as seguintes propriedades:

Tipo de Mídia	Descrição
<b>all</b>	Utilizado para todos os tipos de mídia
<b>aural</b>	Utilizado para microfones e sintetizadores de som
<b>braille</b>	Utilizado para dispositivos de reação ao tato em braile
<b>embossed</b>	Utilizado para impressoras em braile
<b>handheld</b>	Utilizados para pequenos dispositivos (antigos)
<b>print</b>	Utilizado para impressoras
<b>projection</b>	Utilizado para projetores e slides
<b>screen</b>	Utilizado para telas de computador (mais comum)
<b>tty</b>	Utilizado para mídia que utiliza uma grade de caracteres de tamanho fixo como terminais
<b>tv</b>	Utilizado para televisão

Para utilizarmos a regra `@media`, podemos determinar em qual resolução aquele bloco estará disponível. Vejamos no exemplo a seguir:

```
@media (min-width: 1200px) {  
}
```

Neste caso, todas as declarações CSS que forem adicionadas dentro do bloco acima, somente terão efeito quando a largura mínima do dispositivo for 1200px. Assim, essa declaração irá funcionar independentemente de qual dispositivo o usuário acessse, seja uma TV, uma impressora ou uma tela comum.

Para determinar que as declarações do bloco devem ser ativadas quando a largura mínima do dispositivo for 1200px, e esse dispositivo for uma tela de computador, tablet ou smartphone, podemos adicionar mais uma condição que seja aplicada quando a tela for de um computador e a largura mínima for a desejada, como mostra a imagem a seguir:

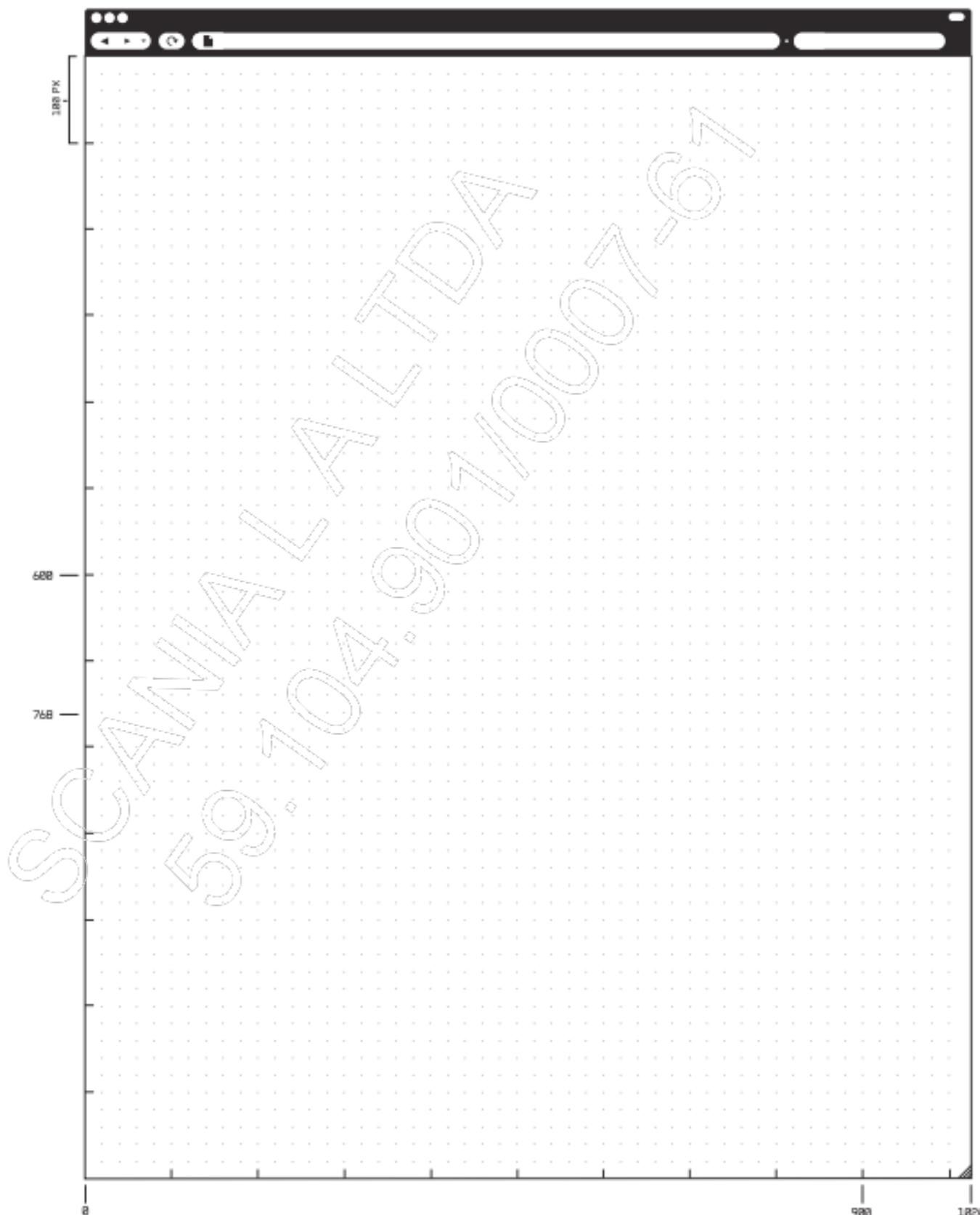
```
@media only screen and (min-width: 1200px) {  
}
```

Vejamos agora quais as medidas ideais para cada dispositivo, bem como um protótipo do dispositivo para criarmos nossos layouts.

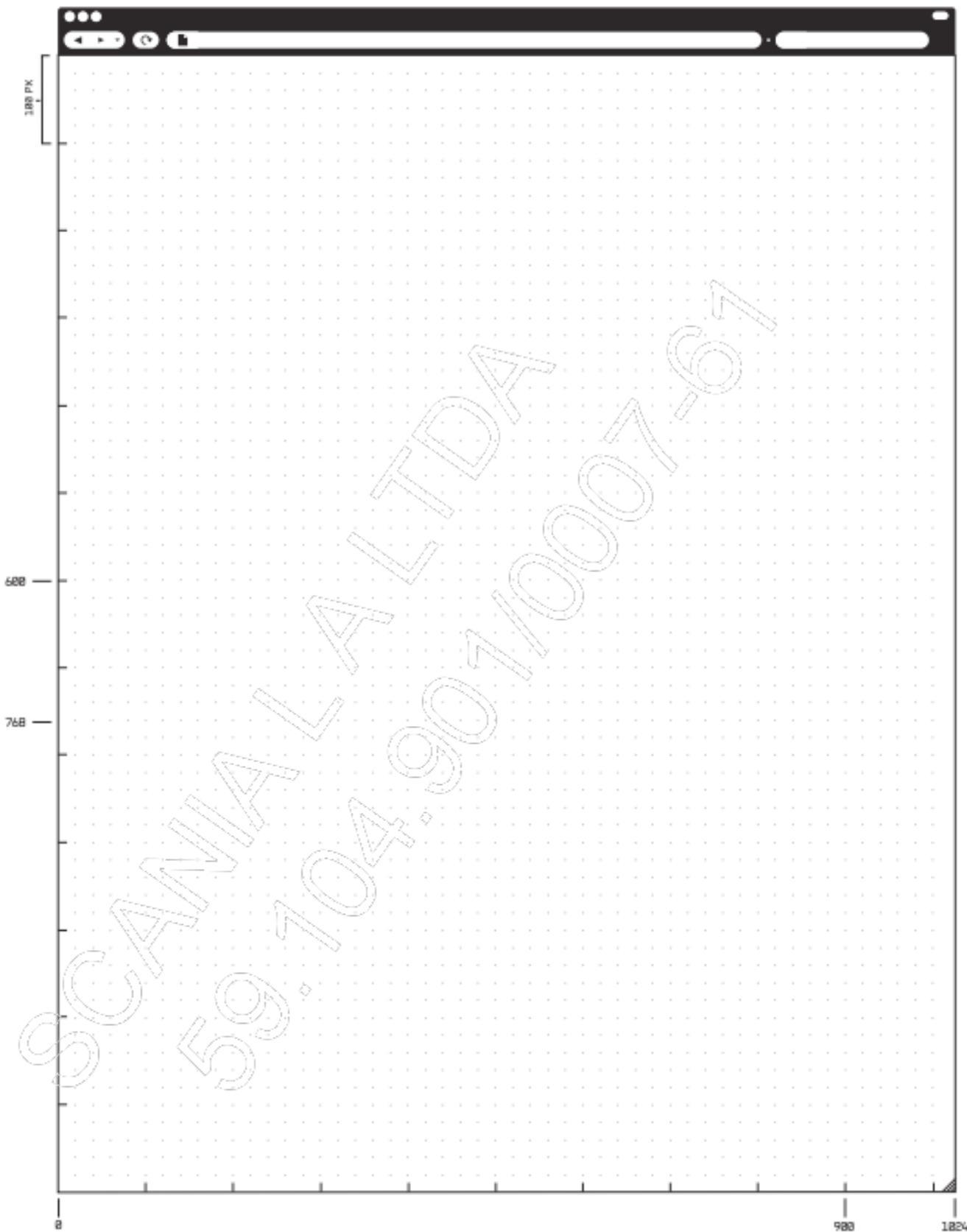
## 7.3.1.1. Projetando para desktop

A fim de criar projetos que funcionem bem nos computadores tradicionais desktop e notebooks, podemos utilizar as seguintes condições com a regra `@media`:

Largura	Regra
<b>800 - 1024px</b>	<code>@media (min-width: 768px) and (max-width: 992px) { ... }</code>
<b>1200px</b>	<code>@media (min-width: 1200px) { ... }</code>

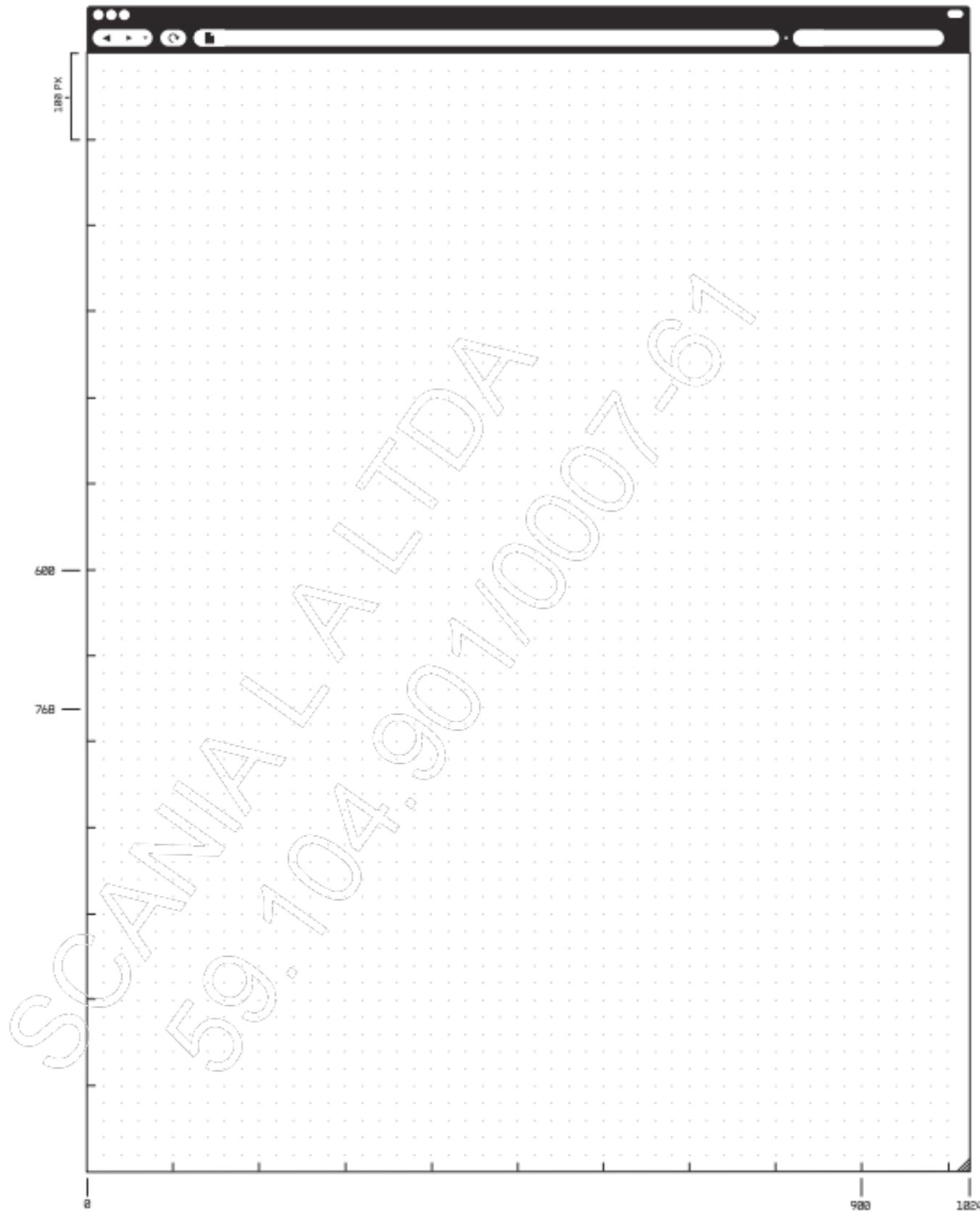


# Produzindo um layout responsivo



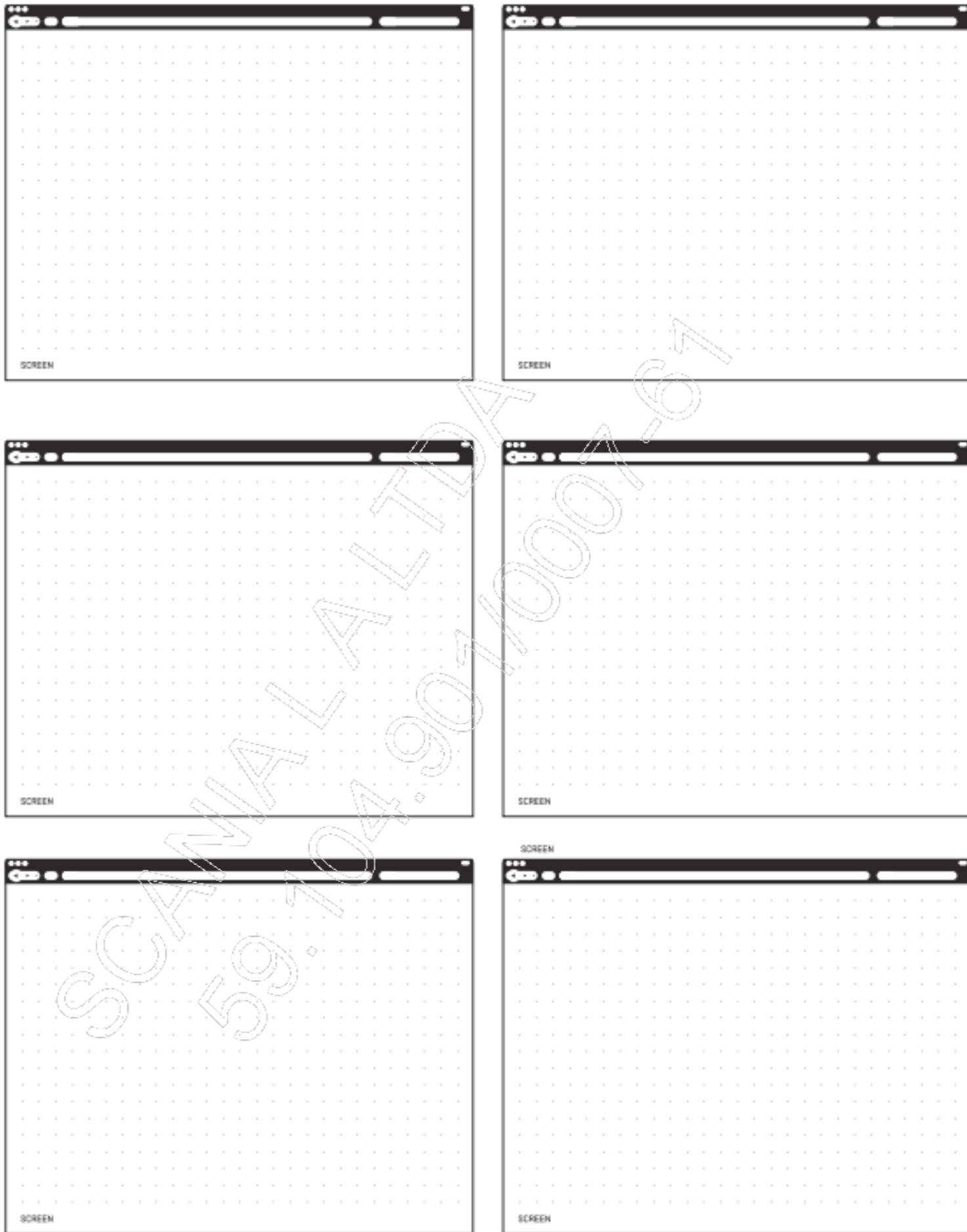
# CSS3

---



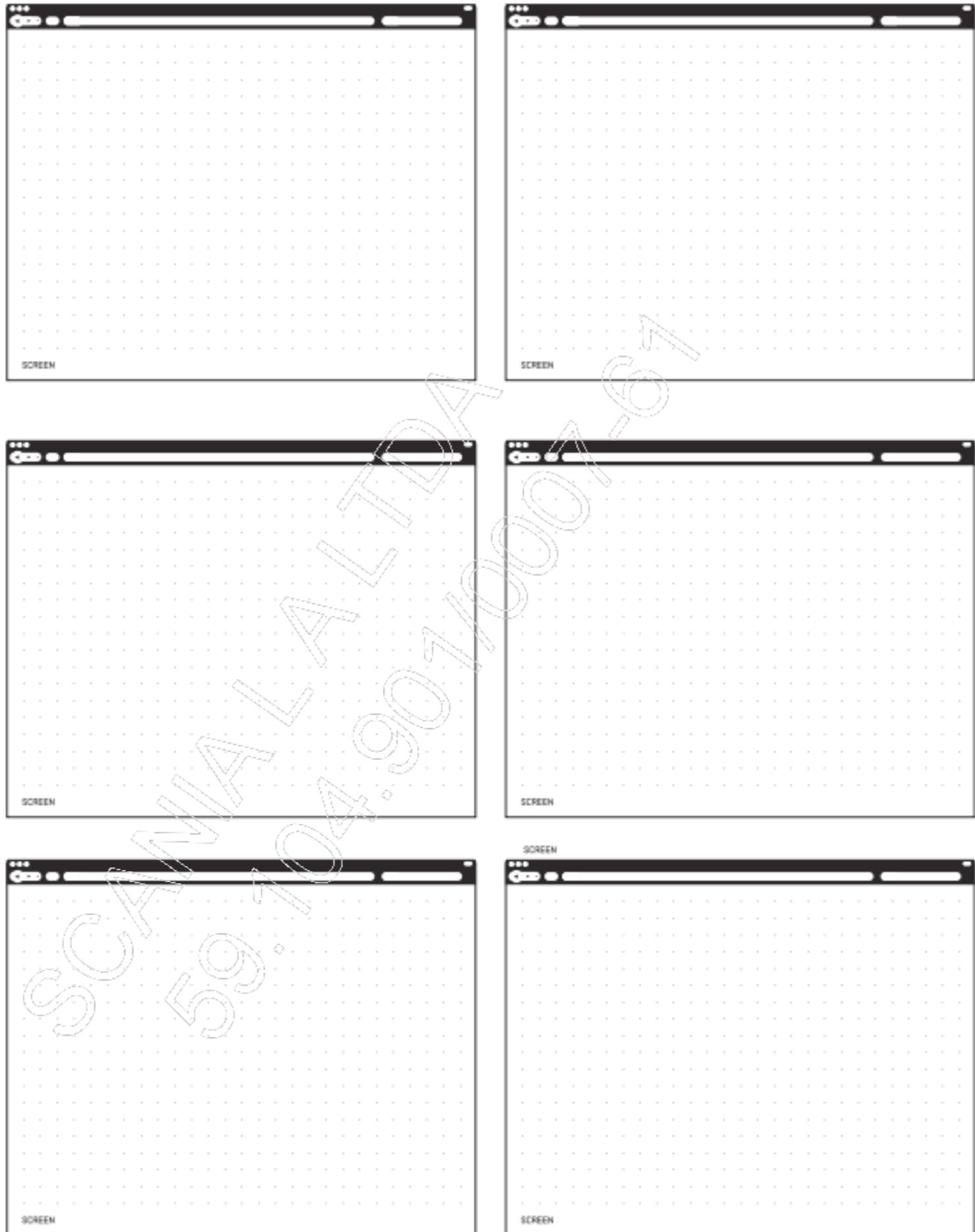
# Produzindo um layout responsivo

7



# CSS3

---



## 7.3.1.2. Projetando para TV e dispositivos de alta definição

Quando projetamos um layout para TV ou dispositivos de alta resolução como o iPad Retina e outros com uma definição com quase o dobro de qualidade do comum, além de trabalharmos com uma dimensão maior, os pixels nessas telas mudam de tamanho. Muitas vezes chegam a ter o dobro do tamanho, já que uma imagem que tenha 2 pixels de borda, na realidade nos dispositivos de alta resolução elas podem ter 3 ou até 4 pixels físicos.

A CSS tem uma propriedade para tratar esse tipo de situação, que é chamada de **min-device-pixel-ratio**, onde o valor padrão é 1, e que significa a dimensão comum. Caso fosse preciso um aumento de 50% do pixel ratio, a propriedade ficaria **-webkit-min-device-pixel-ratio: 1.5;** equivalente a 144dpi, em vez do padrão 96dpi de resolução. Ou podemos ainda utilizar a nova propriedade **resolution** com suas variações, **min-resolution** e **max-resolution**.

A fim de criar projetos que funcionem bem em TV e dispositivos de alta definição, podemos utilizar as seguintes condições com a regra **@media**:

Alta Definição	Regra
<b>1200px+ ou Retina display</b>	<code>@media (min-width: 1200px), (-webkit-min-device-pixel-ratio: 1.25), (min-resolution: 120dpi) { ... }</code>

## 7.3.1.3. Projetando para tablet

Atualmente existem inúmeros modelos de tablet, alguns com 10" e outros com 7", mas como mencionado no início deste capítulo, a área de conteúdo do site ou viewport é o tamanho utilizado para medir a resolução do conteúdo, e tanto nos tablets como nos celulares smartphones, temos dois modos de visualização.

- **Portrait – Retrato**

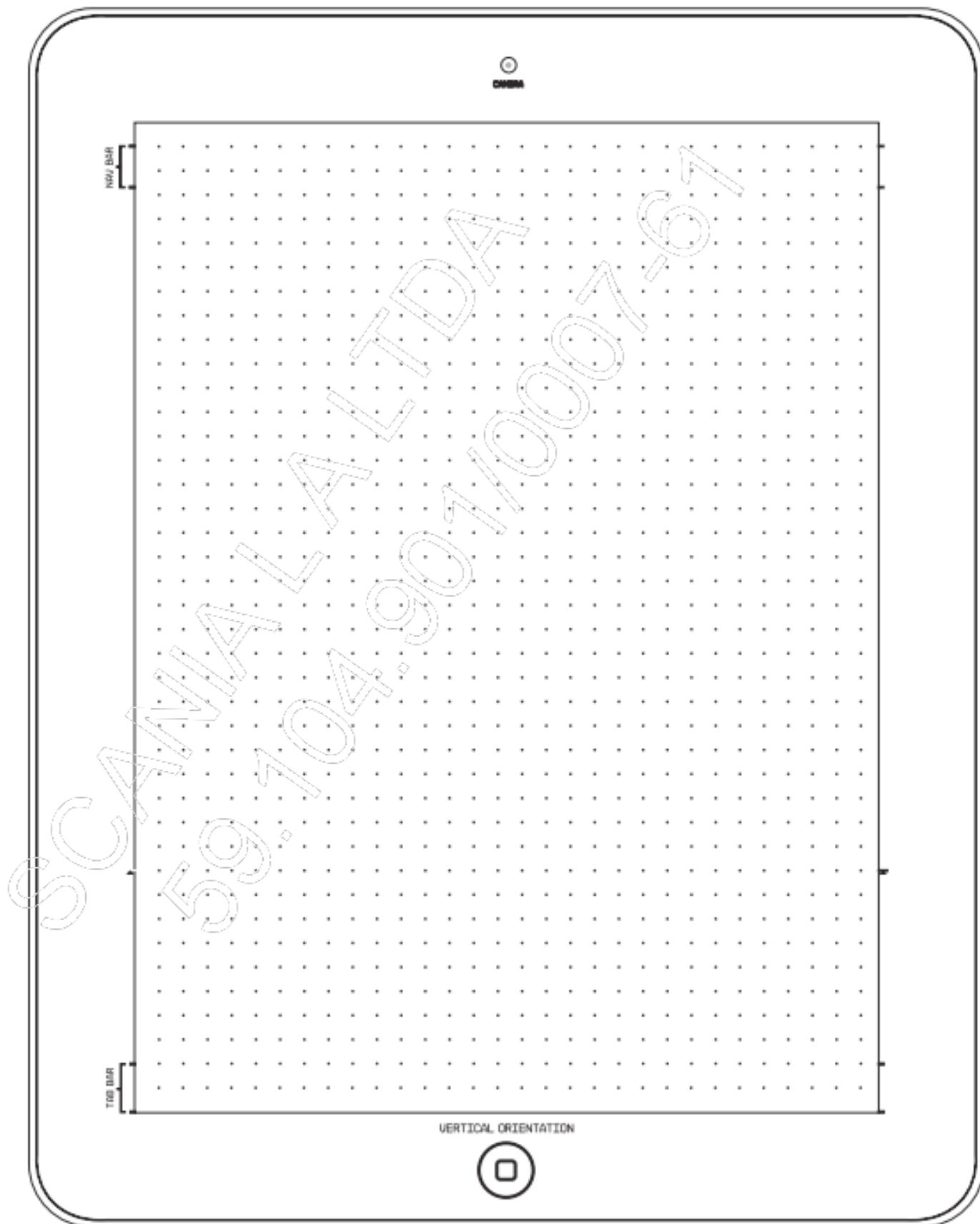
O tablet ou smartphone é utilizado na vertical, e neste caso sua largura pode possuir as seguintes dimensões:

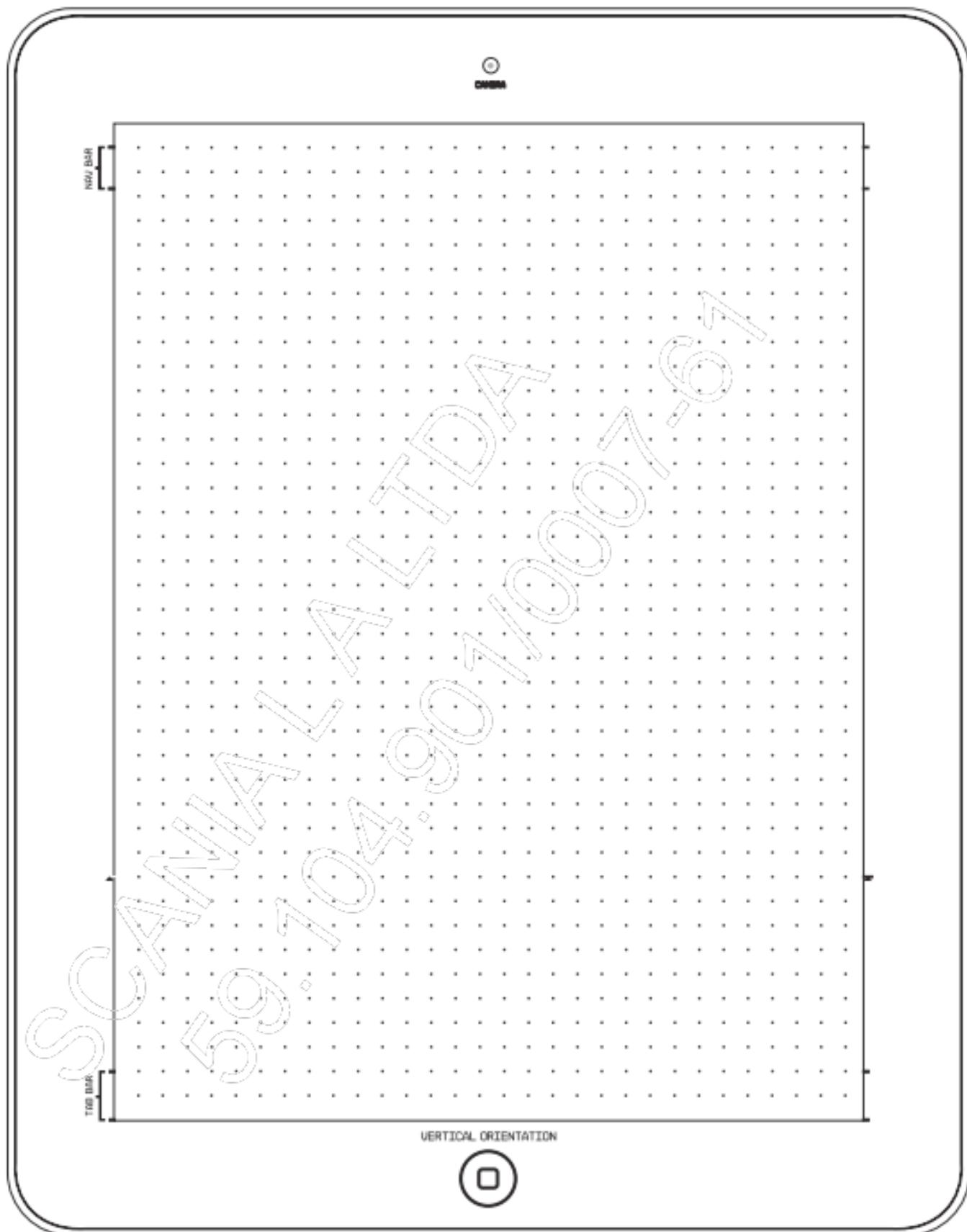
Largura	Regra
<b>Máximo 800</b>	<code>@media only screen and (max-width: 767px) { ... }</code>

- **Landscape – Paisagem**

O tablet ou smartphone é utilizado na horizontal, e neste caso sua largura pode possuir as seguintes dimensões:

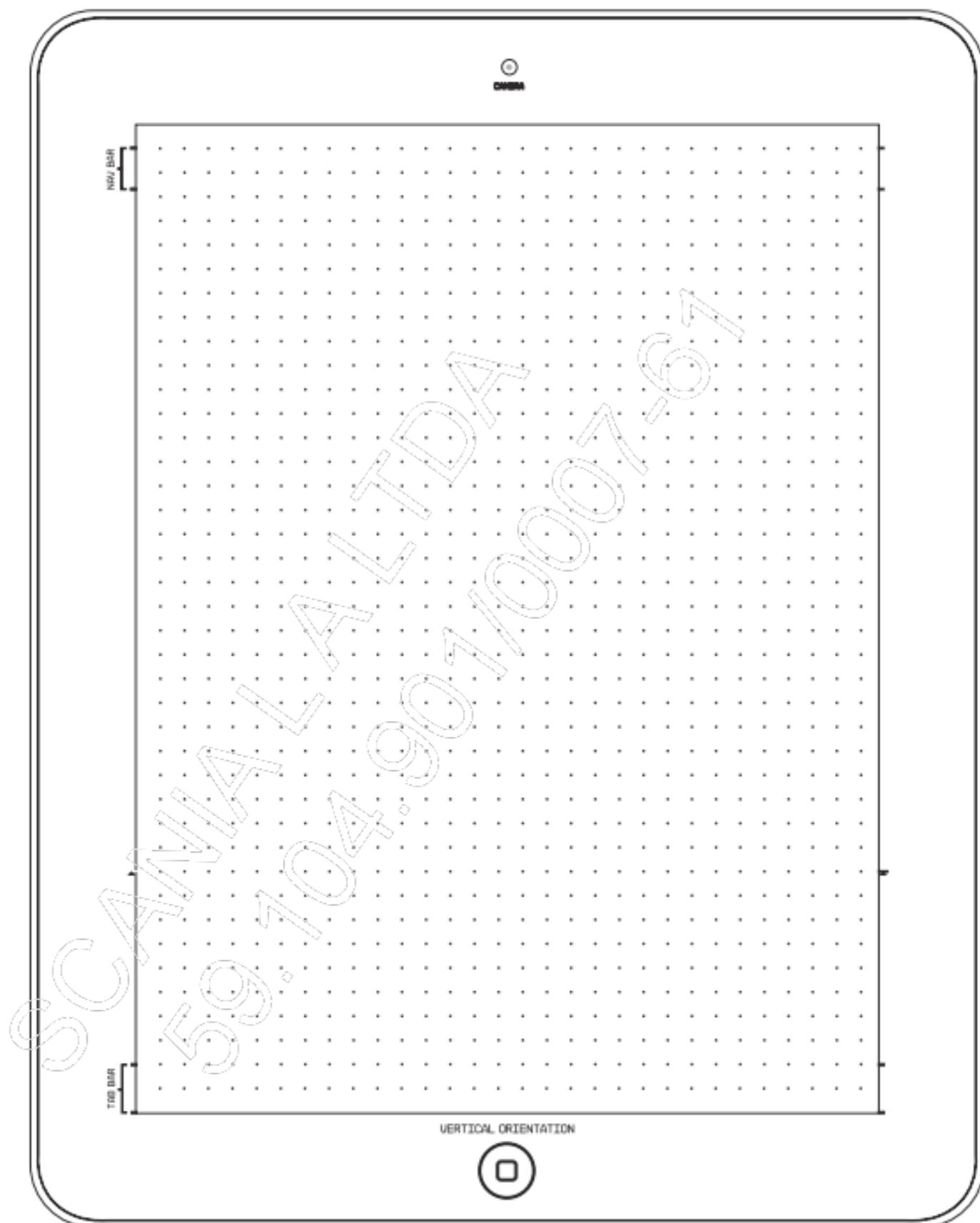
Largura	Regra
<b>800 - 980px</b>	@media (min-width: 768px) and (max-width: 979px) { ... }

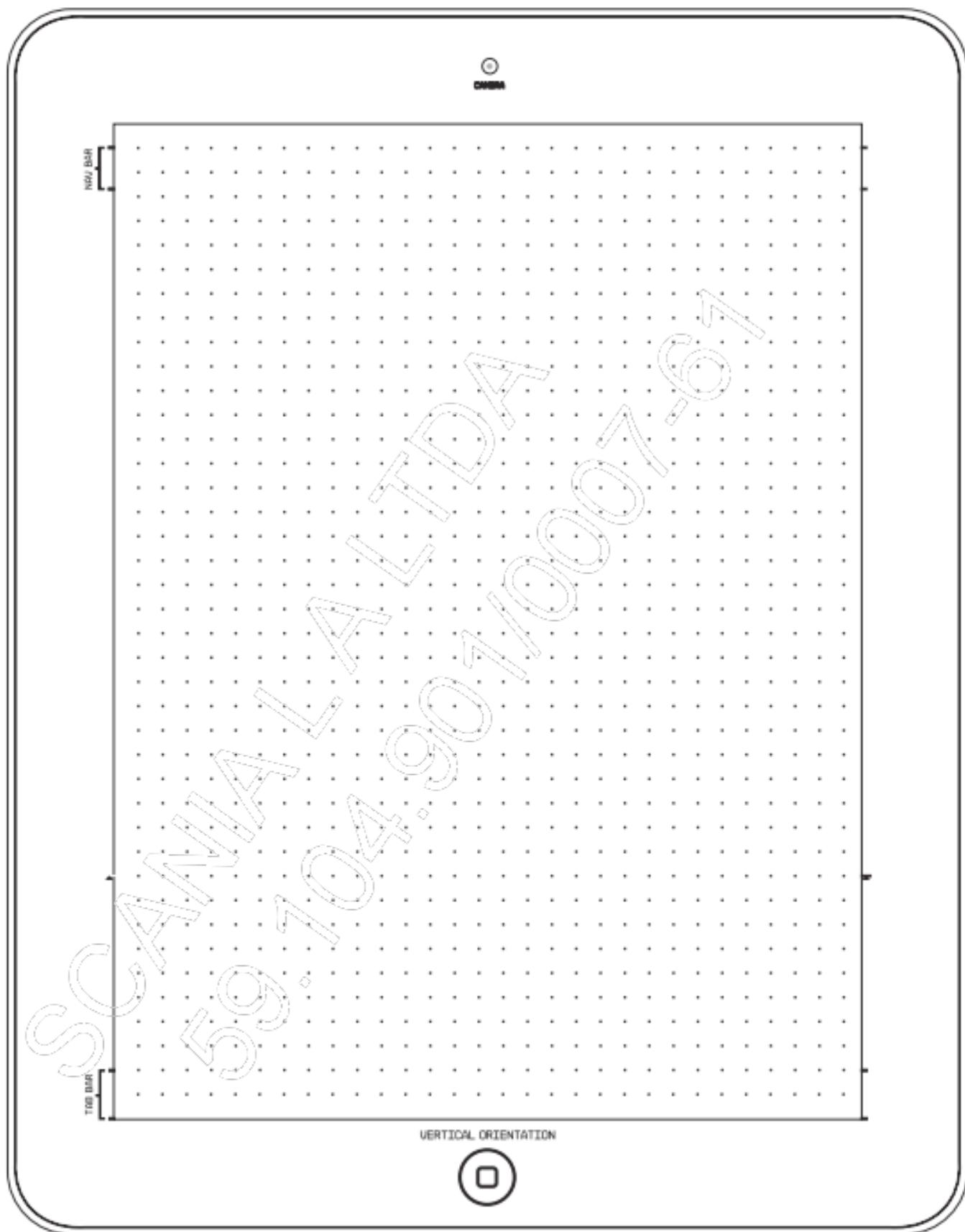




# CSS3

---





## 7.3.1.4. Projetando para smartphone

Assim como os tablets, hoje existem inúmeros tipos de smartphones, alguns de alta definição e outros mais simples. Porém, como mencionado no início deste capítulo, a área de conteúdo do site ou o viewport é o tamanho utilizado para medir a resolução do conteúdo, e tanto nos tablets como nos celulares smartphones, temos dois modos de visualização.

- **Portrait – Retrato**

O smartphone é utilizado na vertical, e neste caso sua largura pode possuir as seguintes dimensões:

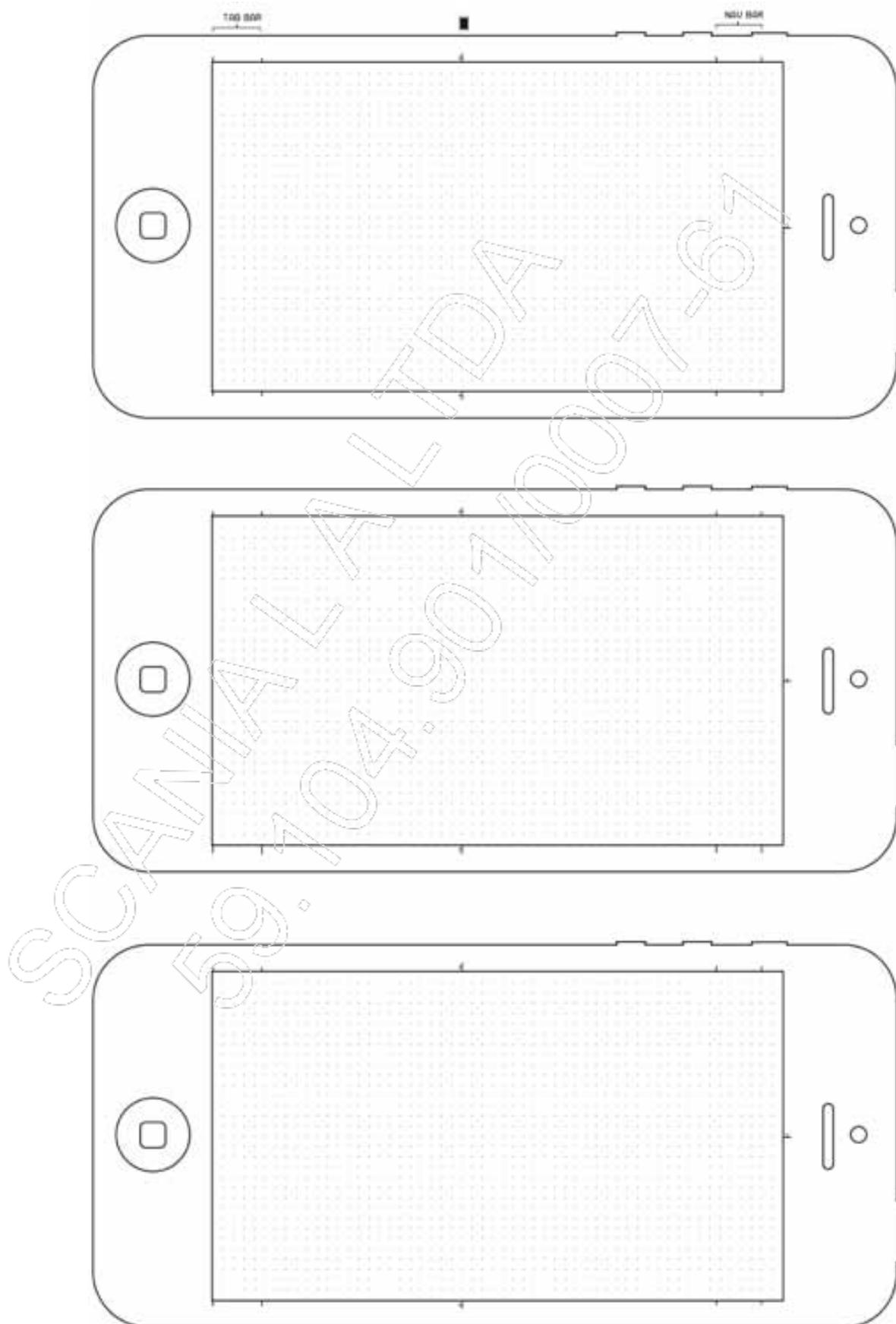
Largura	Regra
<b>Máximo 480px</b>	@media (max-width: 480px) { ... }

# Produzindo um layout responsivo

- **Landscape – Paisagem**

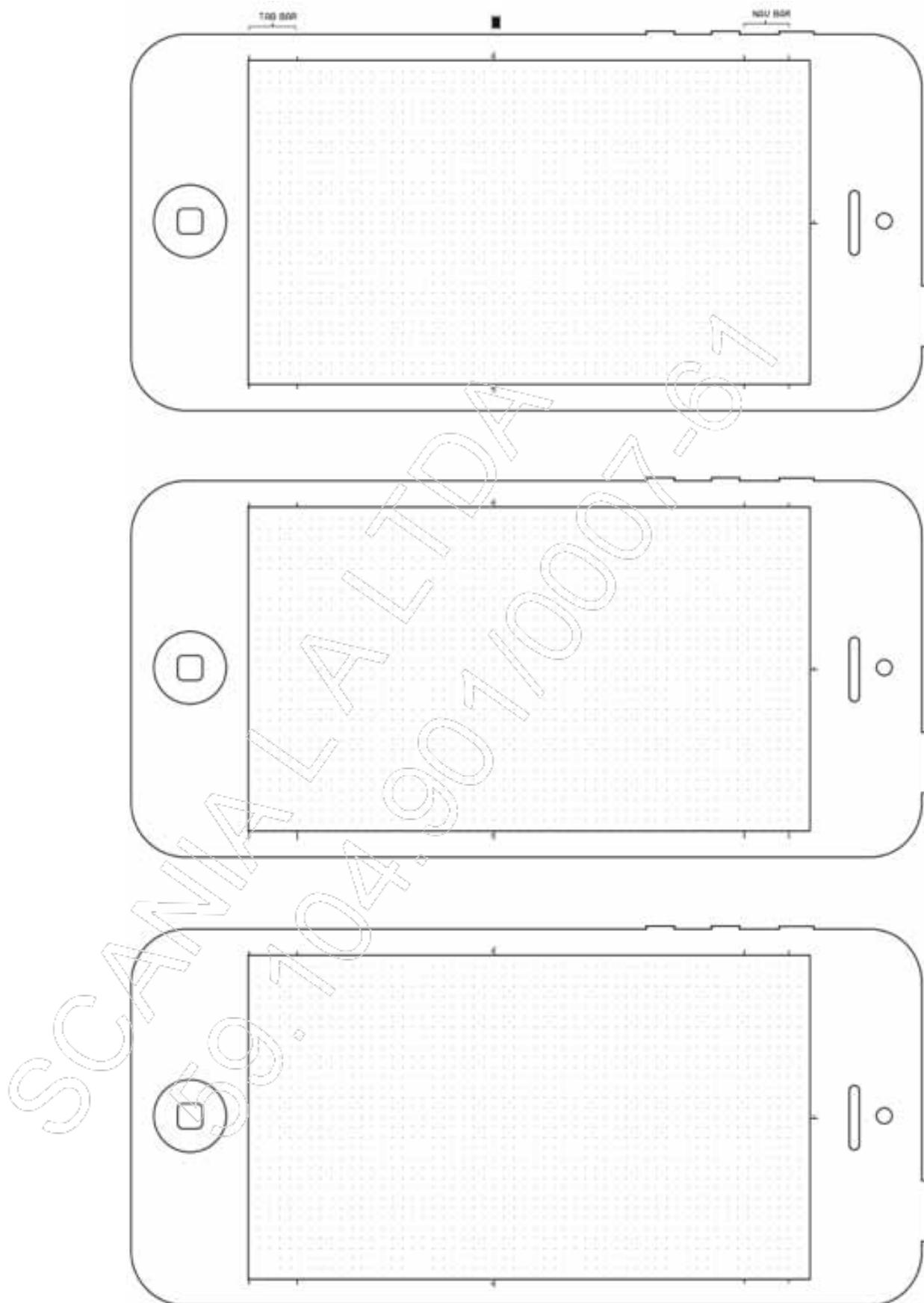
O smartphone é utilizado na horizontal, e neste caso sua largura pode possuir as seguintes dimensões:

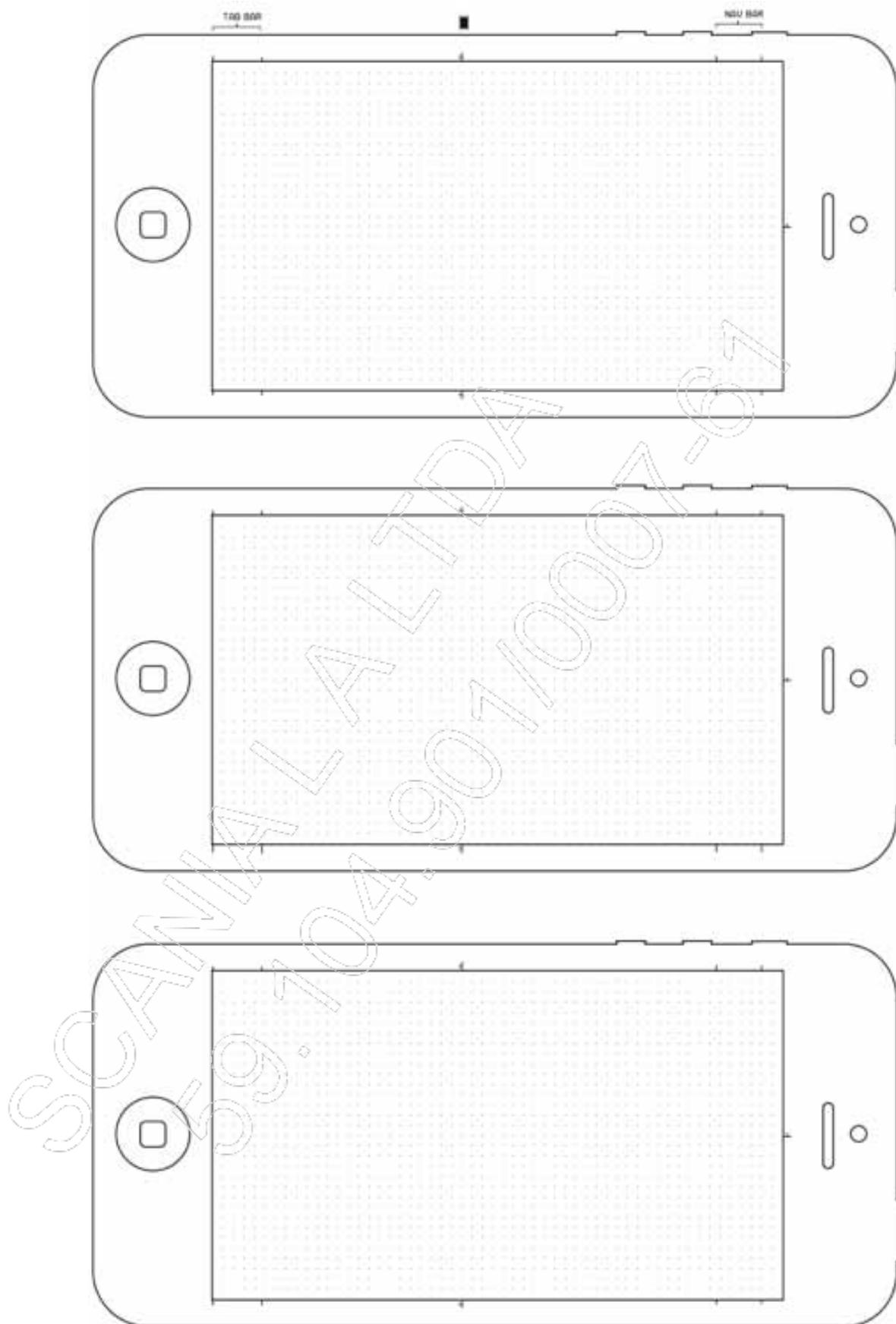
Largura	Regra
<b>Máximo 800px</b>	@media (max-width: 767px) { ... }



# CSS3

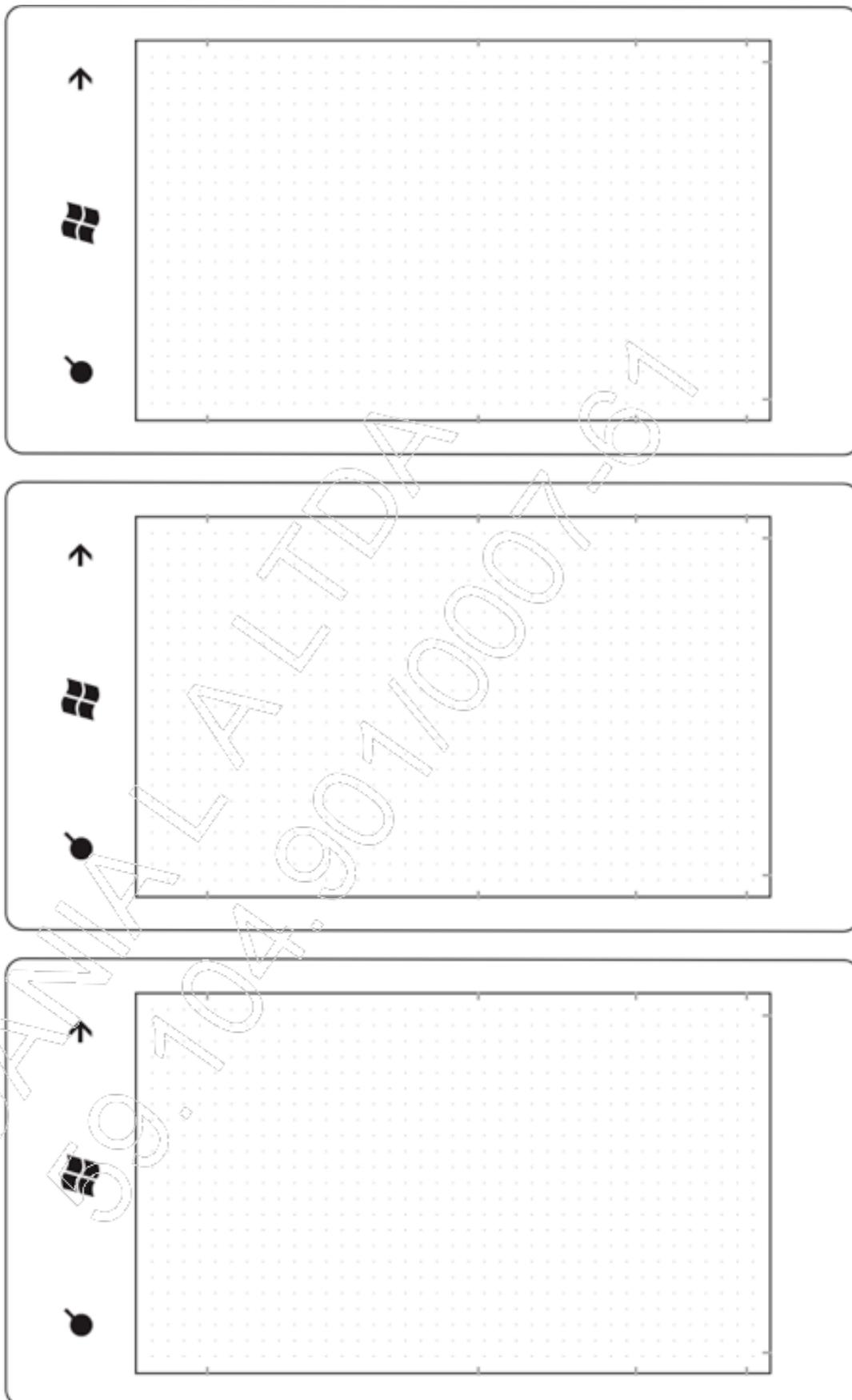
---

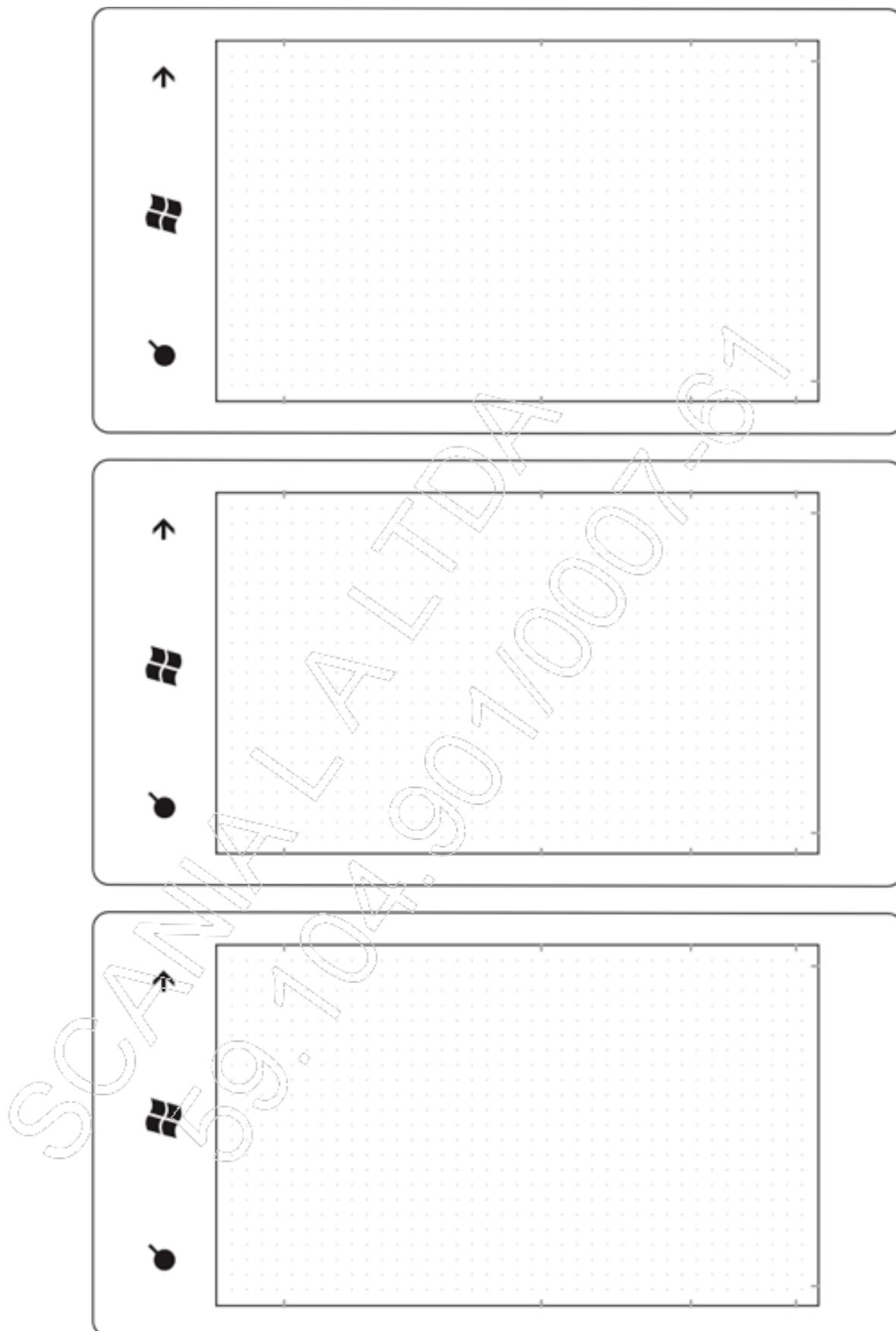




## CSS3

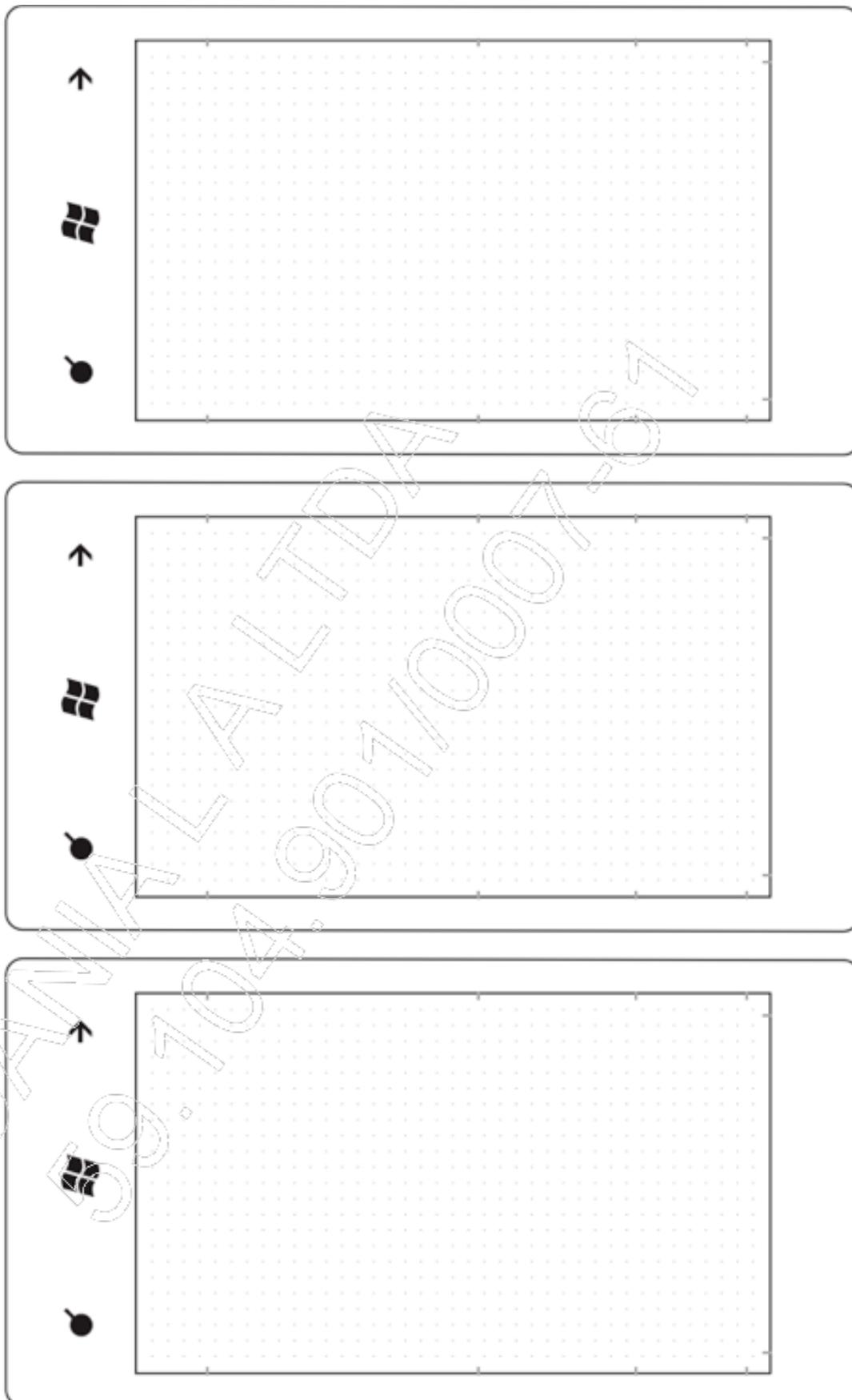
---





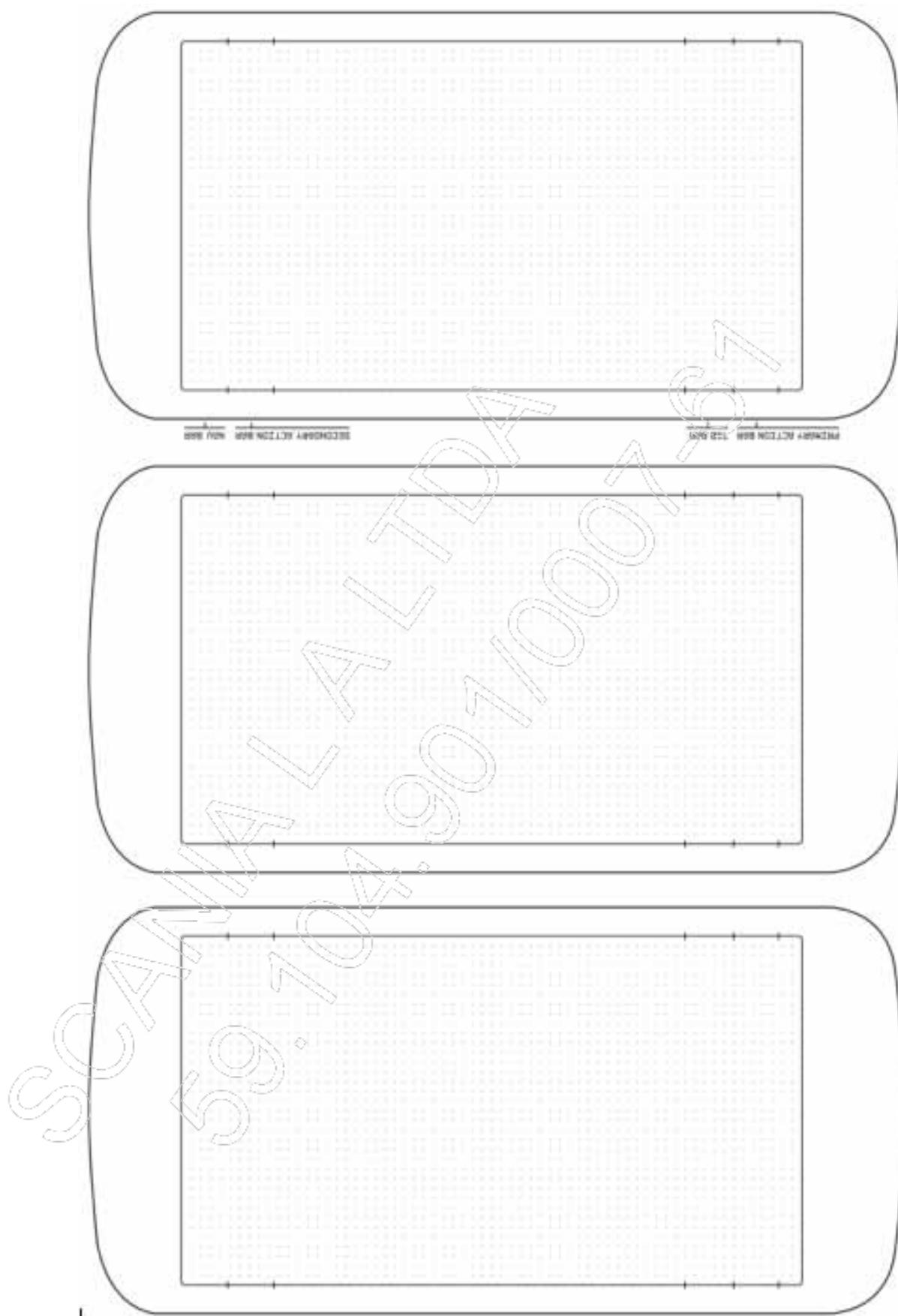
## CSS3

---



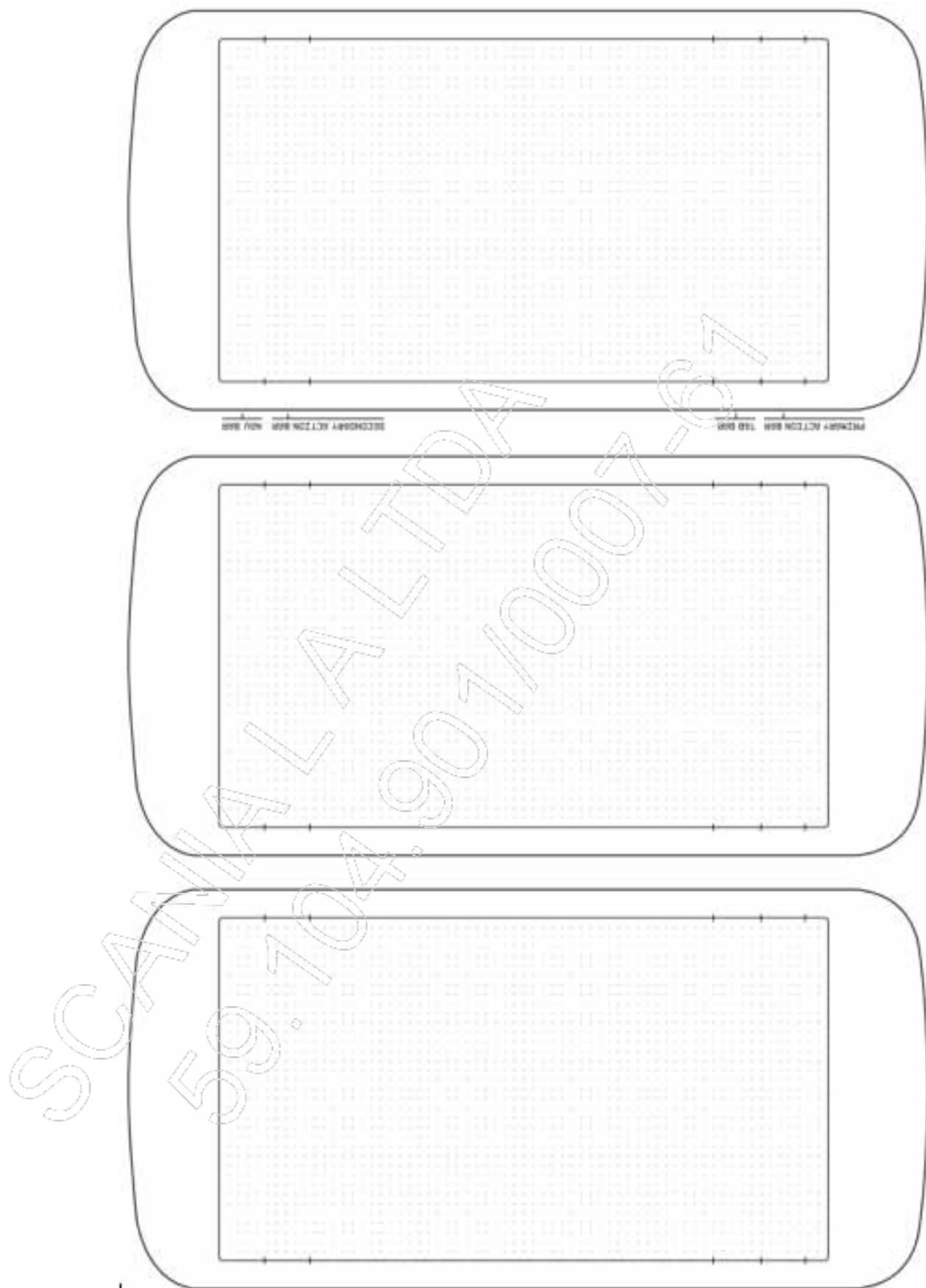
# Produzindo um layout responsivo

7



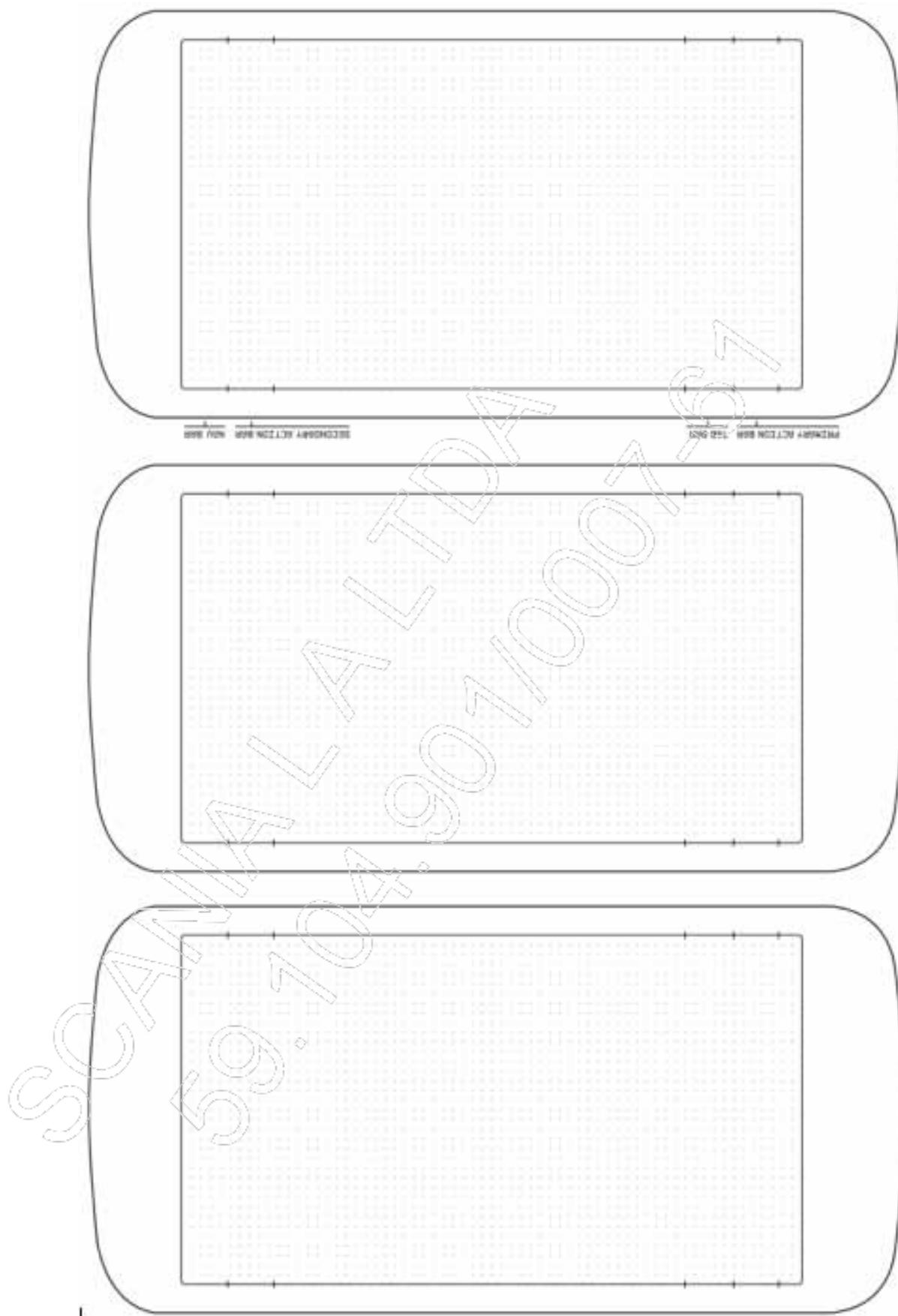
## CSS3

---



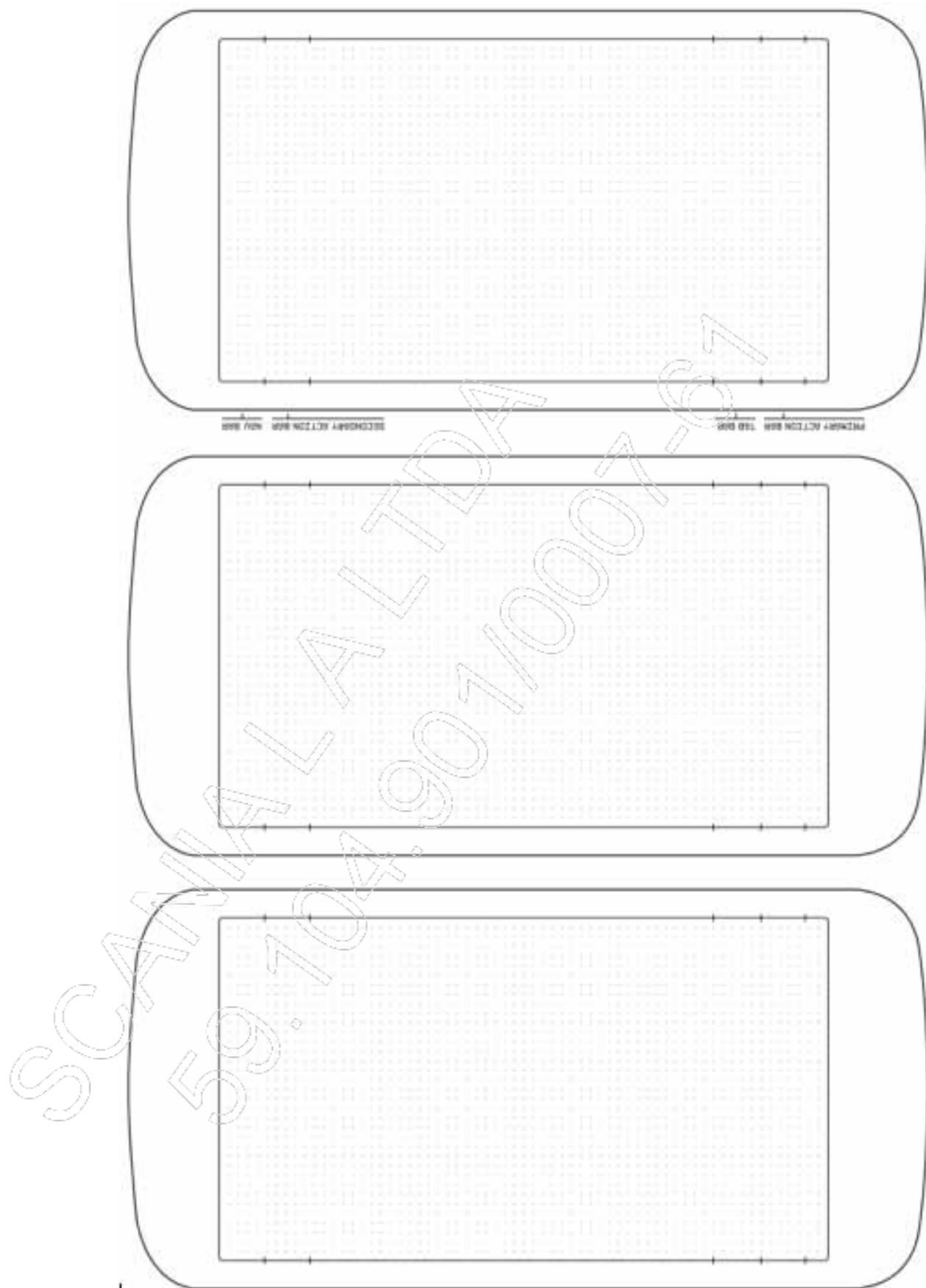
# Produzindo um layout responsivo

7



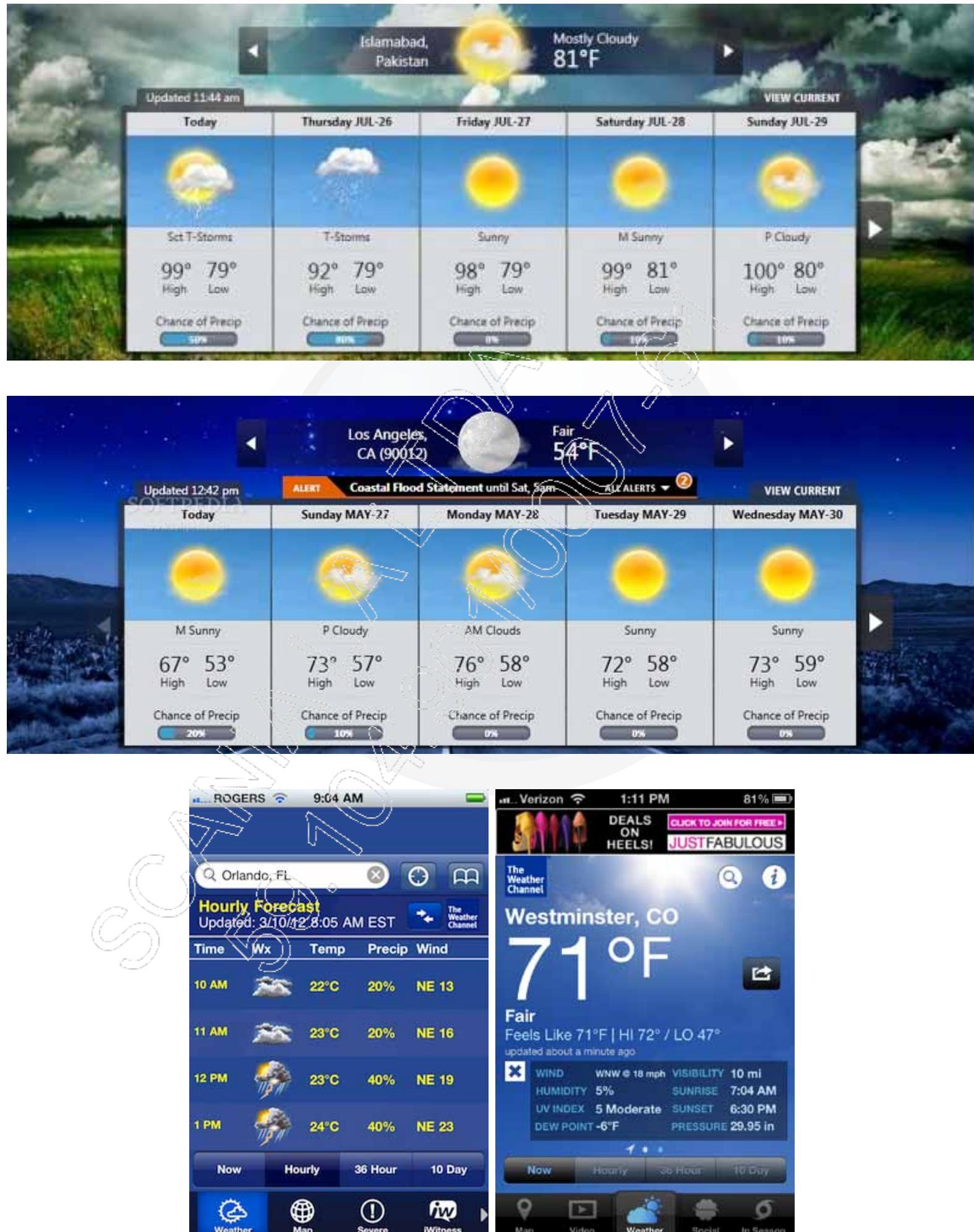
# CSS3

---



# Produzindo um layout responsivo

Após a criação do arquivo HTML, teremos o resultado que foi criado no Wireframe no início do capítulo:





Estes modelos nos quais visualizamos o resultado final do projeto são chamados de **mockups**. De responsabilidade do criador do layout, eles produzem uma grande impressão no cliente final, uma vez que ele está vendo o layout, mesmo que sem vida, diretamente no dispositivo.

Vejamos um exemplo prático do uso das media queries:

- Arquivo: cap7.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Projetando um layout responsivo</title>
6     <link rel="stylesheet" type="text/css" href="css/estilo_cap7.css">
7 </head>
8 <body>
9     <main class="fundo-transparente">
10         <section class="topo">
11             <div id="cidade-pais">
12                 Parauapebas
13                 Brasil
14             </p>
15             </div>
16             <div id="icone"></div>
17         </section>
18         <div id="clima">
19             <p>Límpio</p>
20             <p class="temperatura">23°</p>
21         </div>
22     </main>
23 </body>
24 </html>
```

- Arquivo: estilo\_cap7.css

```
1 @charset "UTF-8";
2 @import url('http://fonts.googleapis.com/css?family=Nixie+One|Pompiere|Felipa');
3 body, main{
4     margin: 0;
5     padding: 0;
6     font-size: 14px;
7 }
8 main{
9     min-width: 99%;
10    min-height: 99%;
11    position: relative;
12 }
13 @viewport {
14     width: device-width;
15     zoom: 1;
16 }
17 .fundo-transparente{
18     position: absolute;
19     margin: 3% 3%;
20     background-color: rgba(15, 13, 06, 0.3);
21     min-width: 93%;
22     min-height: 92%;
23 }
24 .topo, #cidade-pais, #icone{
25     position: relative;
26 }
```

# CSS3

```
28
29 @media only screen and (max-width: 480px) {
30     body{
31         background-image: url("../images/fundo-smartphone.jpg");
32         background-size: 100%;
33         background-repeat: no-repeat;
34     }
35     #cidade-pais{
36         margin-left: 25%;
37         color: #eee;
38         font-size: 200%;font-family: 'Pompiere', cursive;
39         font-weight: bold;
40     }
41     #clima{
42         position: absolute;
43         bottom: 0; margin-left: 2%;
44         font-family: arial, sans-serif;
45         color: #eee;font-size: 49px;
46         font-family: 'Felipa', cursive;
47     }
48     #clima p{
49         padding: 0; margin: 0;
50     }
51     .temperatura{
52         font-size: 120px; padding: 0; margin: 0 2%;
53         font-family: 'Nixie One', cursive;
54     }
55 }
```

```
57 @media only screen and (min-width: 481px) {
58     body{
59         background-image: url("../images/fundo-tablet.jpg");
60         background-repeat: no-repeat;
61     }
62     #cidade-pais{
63         margin-left: 20%;
64         position: relative;
65         width: 115px;color: #eee;
66         font-size: 200%;
67         font-family: 'Pompiere', cursive;
68         font-weight: bold;
69         text-align: right;
70         float: left;
71     }
72     #icone{
73         width:150px;height: 200px;
74         background-image: url('../images/sol.png');
75         background-repeat: no-repeat;float: left;
76     }
77     #clima{
78         margin: 30px 0px;
79         color: #eee;font-size: 200%;
80         font-family: 'Pompiere', cursive;
81         font-weight: bold;
82     }
83     #clima p {
84         margin: 0;padding: 0;
85     }
86 }
```

```
87
88 @media only screen and (min-width: 1024px) {
89
90   body{
91     background-image: url("../images/fundo.jpg");
92     background-size: 100%;
93     background-repeat: no-repeat;
94   }
95   #icone{
96     width:250px;
97     height: 200px;
98     background-image: url('../images/sol.png');
99     background-repeat: no-repeat;
100    float: left;
101  }
102 }
103 #cidade-pais{
104   margin-left: 35%;
105 }
```

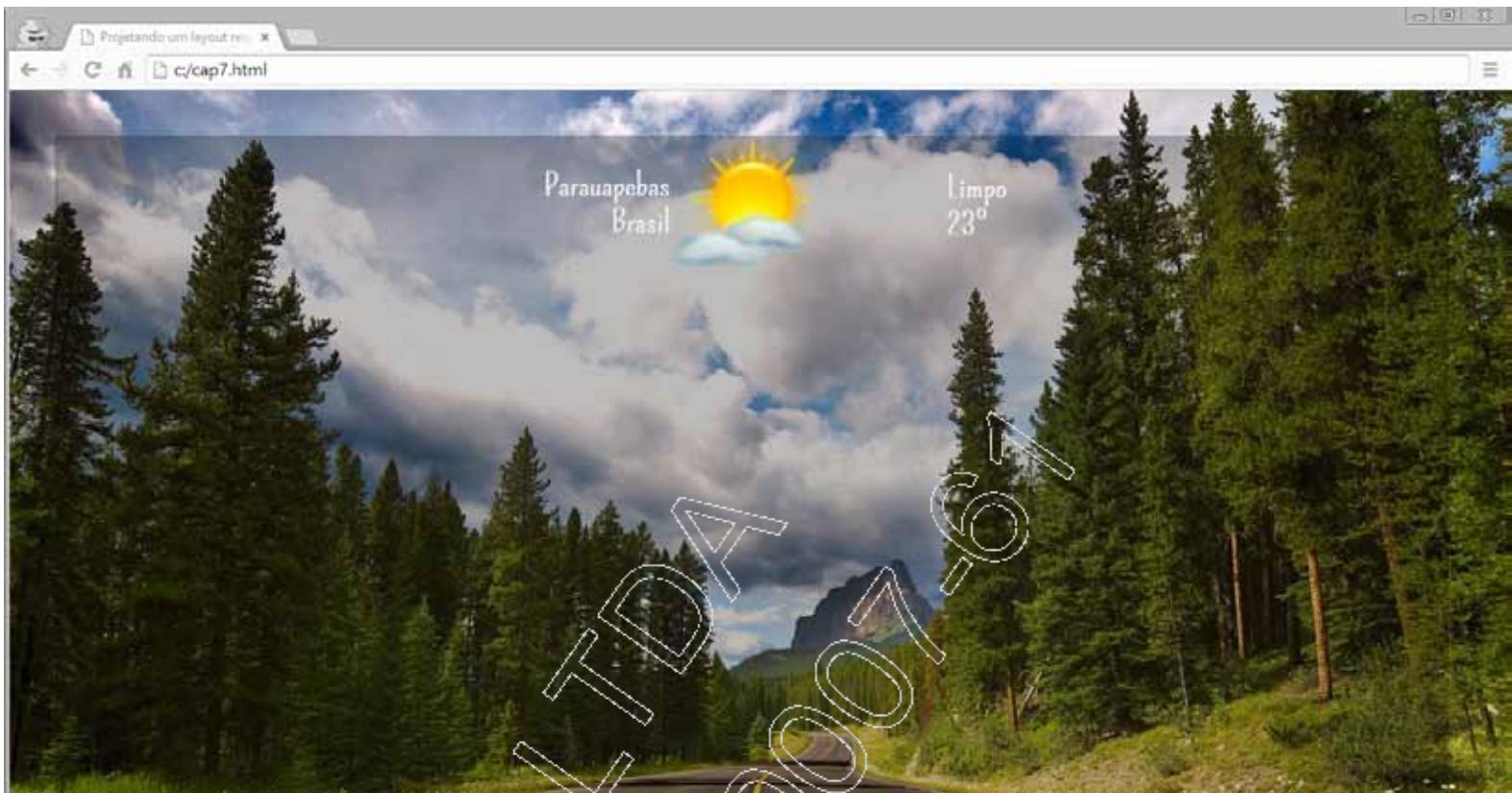
```
87
88 @media only screen and (min-width: 1024px) {
89
90   body{
91     background-image: url("../images/fundo.jpg");
92     background-size: 100%;
93     background-repeat: no-repeat;
94   }
95   #icone{
96     width:250px;
97     height: 200px;
98     background-image: url('../images/sol.png');
99     background-repeat: no-repeat;
100    float: left;
101  }
102 }
103 #cidade-pais{
104   margin-left: 35%;
105 }
```

# CSS3

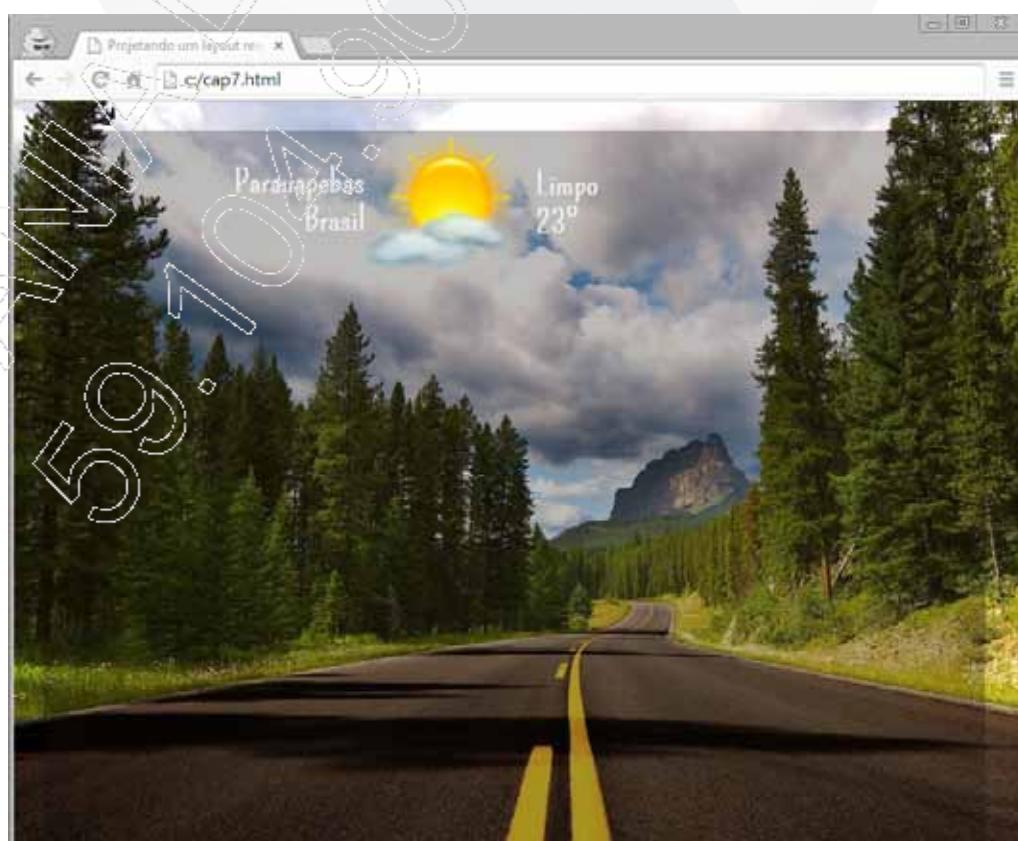
---

Veja o resultado:

- Desktop de alta resolução



- Tablet



- Smartphone



# Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- **Briefing** é um roteiro com o conteúdo que desejamos criar, que é esboçado em reuniões com o cliente para realizar o levantamento das informações relativas ao projeto, e compreende o primeiro passo da criação de um layout.
- **Wireframe** é um guia visual básico, sem muitos detalhes, para mostrar a estrutura do site. Com este guia conseguimos ter uma ideia de onde deverão ficar imagens, textos, logotipo e a disposição das demais informações desejadas;
- Para adequar as **dimensões** do conteúdo a vários dispositivos, utilizamos como referência a tag meta com o **name="viewport"**, que determina a área visível do conteúdo em várias dimensões diferentes;
- Com a nova regra **@viewport**, podemos transferir o controle do viewport que fica na meta tag, diretamente para a folha de estilo;
- Ethan Marcotte utilizou em 2010 o termo Responsive Web Design (RWD) em um artigo notório no site [www.alistapart.com](http://www.alistapart.com), para designar a criação de sites que sejam adaptáveis aos mais diversos dispositivos.
- A regra **@media** permite que possamos aplicar estilos e regras com algumas condições, variáveis conforme o dispositivo exibidor do conteúdo: computadores desktop, TVs, tablets e smartphones.

7

# Produzindo um layout responsivo

Teste seus conhecimentos

SCAM 50°  
TDA 2007-67



**IMPACTA**  
EDITORA

## 1. Qual a sintaxe correta para configurar a largura e escala inicial de zoom usando a tag <meta>?

- a) <meta display="viewport" content="width:device-width" and "initial-scale=1">
- b) <meta name="viewport" content="width=device-width, initial-scale=1">
- c) <meta name="viewport" content="width:device-width, initial-scale;1">
- d) <meta name="viewpoit" content="width:device-width; initial-scale:1">
- e) <@viewport width=device-width, initial-scale=1>

## 2. Qual a sintaxe correta para configurar a largura e escala inicial de zoom do viewport usando a regra @viewport?

- a) @viewport{ width:device-width; initial-scale:1;}
- b) viewport{width=device-width; inicital-scale=1;}
- c) @view-port{}
- d) <meta name="viewpoit" content="width:device-width; initial-scale:1">
- e) <@viewport width=device-width, initial-scale=1>

## 3. Como aplicar estilos que serão usados somente ao imprimir o documento?

- a) @media screen { }
- b) <link rel="stylesheet" type="text/css" href="estilo.css" media="embossed"/>
- c) @media print{ }
- d) @media all { }
- e) <link rel="stylesheet" type="text/css" href="estilo.css" media="screen"/>

## 4. Qual seria uma sintaxe válida para configurar estilos para smartphones?

- a) @media only screen and(min-width: 480px) {}
- b) @media only screen and(maxwidth: 640px) {
- c) @media (min-width: 840px) {
- d) @media only print and (max-width: 480px) {
- e) @media only screen and (max-width: 480px) {

## 5. Quais os dois tipos de posições existentes para tablets e smartphone?

- a) width e height
- b) portrait e landscape
- c) portrait e width
- d) height e landscape
- e) altura e retrato

7

# Produzindo um layout responsivo

Mãos à obra!

SCANNER 50° TDA 007-67



**IMPACTA**  
EDITORA

## Laboratório 1

### A – Importando layouts

1. Utilizando o projeto criado em laboratórios anteriores, crie um arquivo chamado **estilo.css** e importe os arquivos de acordo com a tabela a seguir:

Resolução	Importe o arquivo
max-width: 480px	smartphone.css
Min-width: 481px	tablet.css
Min-width: 1024px	desktop.css

2. Renomeie o arquivo **desktop.html** para **index.html** e chame o arquivo **estilo.css** criado.

# Transições em CSS3

8

- ✓ Introdução;
- ✓ transition-property;
- ✓ transition-duration;
- ✓ transition-timing-function;
- ✓ transition-delay;
- ✓ transition.



**IMPACTA**  
EDITORA

## 8.1. Introdução

As transições em CSS permitem que as mudanças nos valores das propriedades CSS ocorram suavemente e durante um tempo especificado. Esta propriedade, denominada **transitions**, analisa a mudança de valor entre as propriedades e faz com que essa transição, ao invés de ocorrer de forma brusca, ocorra suavemente em um tempo predeterminado.

As propriedades de transição são:

- transition-property;
- transition-duration;
- transition-timing-function;
- transition-delay;
- transition.

Elas estão disponíveis na versão CSS3, e os navegadores que a suportam são:

Navegador	Versão
Internet Explorer	10 ou superior
Mozilla Firefox	21 ou superior
Google Chrome	27 ou superior
Safari	5.1 ou superior
Opera	16 ou superior

Visto que a propriedade transition não funciona em todos os elementos, segue ainda uma listagem dos elementos que suportam seu uso:

Propriedade	Tipo
background-color	Cor
background-position	Medida CSS e porcentagem
border-bottom-color	Cor
border-bottom-width	Medida CSS

Propriedade	Tipo
border-left-color	Cor
border-left-width	Medida CSS
border-right-color	Cor
border-right-width	Medida CSS
border-spacing	Medida CSS
border-top-color	Cor
border-top-width	Medida CSS
bottom	Medida CSS, porcentagem e número inteiro
clip	Um retângulo
color	Cor
font-size	Medida CSS
font-weight	Número inteiro
height	Medida CSS, porcentagem e número inteiro
left	Medida CSS, porcentagem e número inteiro
letter-spacing	Medida CSS
line-height	Medida CSS, porcentagem e número inteiro
margin-bottom	Medida CSS
margin-left	Medida CSS
margin-right	Medida CSS
margin-top	Medida CSS
max-height	Medida CSS, porcentagem e número inteiro
max-width	Medida CSS, porcentagem e número inteiro
min-height	Medida CSS, porcentagem e número inteiro
min-width	Medida CSS, porcentagem e número inteiro

Propriedade	Tipo
opacity	Número inteiro
outline-color	Cor
outline-width	Medida CSS
padding-bottom	Medida CSS
padding-left	Medida CSS
padding-right	Medida CSS
padding-top	Medida CSS
right	Medida CSS, porcentagem e número inteiro
text-indent	Medida CSS, porcentagem e número inteiro
text-shadow	Sombra
top	Medida CSS, porcentagem e número inteiro
vertical-align	Medida CSS
visibility	Visibilidade
width	Medida CSS, porcentagem e número inteiro
word-spacing	Medida CSS
z-index	Número inteiro

Os exemplos deste capítulo serão realizados utilizando o prefixo proprietário, ou seja, as declarações serão: **transition-property**, **transition-duration**, **transition-timing-function**, **transition-delay** e **transition**.

## 8.2. transition-property

Esta propriedade é utilizada para definir sobre qual propriedade o efeito da transição será aplicado. É obrigatória na declaração, pois caso seja omitida, não existirá uma propriedade para se aplicar o efeito da transição. É possível ainda aplicar uma mesma transição para todas as propriedades CSS do elemento, mas neste caso é necessário utilizar o valor `all`.

Veja como é possível utilizar esta propriedade:

```
1 .div1{  
2     transition-property:background-color; /* Transição na cor de fundo */  
3 }  
4 .div2{  
5     transition-property:all; /* Transição em todas as propriedades que  
6      dão suporte a propriedade transition*/  
7 }  
8 .div3{  
9     transition-property:margin-right, text-shadow, color; /* Transição na margem direita,  
10     texto com sombreamento e na cor */  
11 }
```

## 8.3. transition-duration

Esta propriedade é utilizada para definir em segundos a duração do efeito desejado, sendo que o padrão aplicado é 0. É obrigatória na declaração, pois caso seja omitida, assume o valor padrão que é 0 e impossibilita a transição de ser realizada.

Veja como é possível utilizar esta propriedade:

```
1 .div1{  
2     transition-property:margin-left;  
3     transition-duration:1s; /* 0 valor deve ser inserido em segundos */  
4 }  
5 .div2{  
6     transition-property:margin-left, color; /* É possível definir mais de um elemento  
7      para realizar a transição */  
8     transition-duration:1s, 3s; /* é possível definir um tempo para cada  
9      elemento definido na propriedade transition-property */  
10 }  
11 }
```

### 8.4. transition-timing-function

Esta propriedade é utilizada para definir a forma como a transição progride no tempo, ou seja, como o ritmo da transição se comporta durante o efeito. Por padrão, nesta propriedade é utilizada a função **ease**.

Existem duas formas de utilização dessa propriedade. A primeira delas é aplicar alguns valores já predefinidos, que são **linear**, **ease**, **ease-in**, **ease-out** e **ease-in-out**:

```
1 .div1{  
2     transition-timing-function:ease;  
3     transition-timing-function:linear;  
4     transition-timing-function:ease-in;  
5     transition-timing-function:ease-out;  
6     transition-timing-function:ease-in-out;  
7 }
```

A segunda forma é utilizar uma função customizada, especificando quatro coordenadas para definir a **cubic bezier**:

```
1 .div1{  
2     transition-timing-function:cubic-bezier(x1, y1, x2, y2);  
3 }
```

### 8.5. transition-delay

Esta propriedade é utilizada para estabelecer a partir de quanto tempo o efeito da transição vai se iniciar. O valor é inserido em segundos, e o padrão utilizado é 0.

```
1 .div1{  
2     width: 500px;  
3     transition-property:background-color;  
4     transition-duration:2s;  
5     transition-timing-function:linear;  
6     transition-delay:2s;  
7 }
```

## 8.6. transition

Para esta propriedade, existe uma forma abreviada de se declarar todas as propriedades para transição. Basta declarar a propriedade transition para que ela agrupe as quatro propriedades específicas anteriores.

```
1 .div1{  
2     color:red;  
3     transition:margin-left 2s ease-in;  
4 }  
5 .div2{  
6     background-color:#CCC;  
7     transition:background-color 4s linear 1s;  
8 }
```

Exemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style type="text/css">  
    .efeito1{  
        position: relative;  
        left:800px;  
        -moz-transition: left 2s linear;  
        -webkit-transition: left 2s linear;  
        -o-transition: left 2s linear;  
        transition: left 2s linear;  
    }  
    .efeito2{  
        position: relative;  
        left:800px;  
        -moz-transition: left 2s ease;  
        -webkit-transition: left 2s ease;  
        -o-transition: left 2s ease;  
        transition: left 2s ease;  
    }  
    .efeito3{  
        position: relative;  
        left:800px;  
        -moz-transition: left 2s ease-in;  
        -webkit-transition: left 2s ease-in;  
        -o-transition: left 2s ease-in;  
        transition: left 2s ease-in;  
    }
```

```
.efeito4{
    position: relative;
    left:800px;
    -moz-transition: left 2s ease-in-out;
    -webkit-transition: left 2s ease-in-out;
    -o-transition: left 2s ease-in-out;
    transition: left 2s ease-in-out;
}
.efeito5{
    position: relative;
    left:800px;
    -moz-transition: left 2s ease-out;
    -webkit-transition: left 2s ease-out;
    -o-transition: left 2s ease-out;
    transition: left 2s ease-out;
}
.efeito6{
    position: relative;
    left:800px;
    -moz-transition: left 2s linear 2s;
    -webkit-transition: left 2s linear 2s;
    -o-transition: left 2s linear 2s;
    transition: left 2s linear 2s;
}
.efeito7{
    position: relative;
    left:800px;
    -moz-transition: left 2s cubic-bezier(1.0, 4.0, 0, 1.0) 2s;
    -webkit-transition: left 2s cubic-bezier(1.0, 4.0, 0, 1.0) 2s;
    -o-transition: left 2s cubic-bezier(1.0, 4.0, 0, 1.0) 2s;
    transition: left 2s cubic-bezier(1.0, 4.0, 0, 1.0) 2s;
}
</style>
</head>
<body>
    <h1>Treinamento de CSS3</h1>
    <div>
        
    </div>
    <label>Escolha um efeito para aplicar:</label> <br />
    <input type="radio" name="efeito" value="efeito1" checked>
    Linear <br />
    <input type="radio" name="efeito" value="efeito2"> Ease <br />
```

```
<input type="radio" name="efeito" value="efeito3"> Ease-in <br />
<input type="radio" name="efeito" value="efeito4"> Ease-in-out <br />
<input type="radio" name="efeito" value="efeito5"> Ease-out <br />
<input type="radio" name="efeito" value="efeito6"> Delay <br />
<input type="radio" name="efeito" value="efeito6"> Cubic-bezier <br />
<button id="btniniciar">Iniciar</button>
<button id="btnresetar">Resetar</button>
<script type="text/javascript">
var elemento = document.querySelector("div");
// 
document.querySelector("#btniniciar").
addEventListener("click", function() {
    var radios = document.querySelectorAll("input");
    //
    for(var i = 0; i < radios.length; i++) {
        if(radios[i].checked) elemento.className =
radios[i].value;
    }
});
// 
document.querySelector("#btnresetar").
addEventListener("click", function() {
    elemento.removeAttribute('class');
    elemento.removeAttribute('style');
});
</script>
</body>
</html>
```

Resultado:



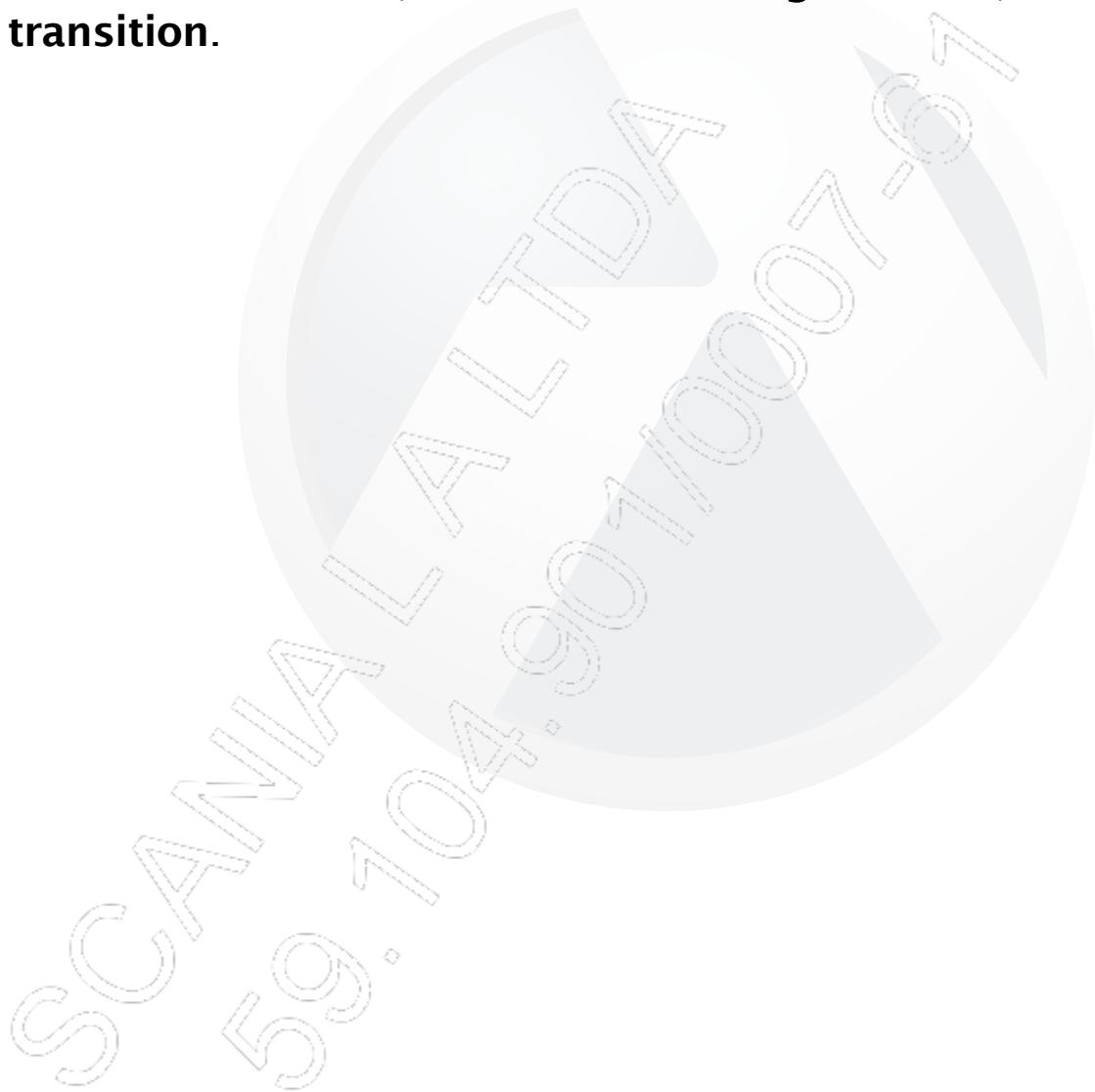
Escolha um efeito para aplicar:

- Linear
- Ease
- Ease-in
- Ease-in-out
- Ease-out
- Delay
- Cubic-bezier

# Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo:

- As propriedades **transition** em CSS permitem que as mudanças nos valores das propriedades CSS ocorram suavemente e durante um tempo especificado.
- Na versão CSS3, as propriedades disponíveis são **transition-property**, **transition-duration**, **transition-timing-function**, **transition-delay** e **transition**.



8

# Transições em CSS3

Teste seus conhecimentos

SCANNING  
50°  
90°  
104°  
100°  
1007-67  
TDA



**IMPACTA**  
EDITORA

## 1. Que função é utilizada para customizar um transição?

- a) function
- b) cubic-bezier
- c) ease
- d) ease-function
- e) new

## 2. Qual o objetivo de se definir o valor “all” na propriedade transition-property?

- a) É utilizado para definir que a transição dê suporte em todas as propriedades que suportam transition.
- b) É utilizado para atribuir um tempo para todas as transições.
- c) É utilizado para definir que a transição não dê suporte em todas as propriedades que suportam transition.
- d) É utilizado para atribuir uma função para todas as transições.
- e) Nenhuma das alternativas anteriores está correta.

**3. Qual das propriedades não é possível utilizar para uma transição?**

- a) transition-queue
- b) transition
- c) transition-duration
- d) transition-delay
- e) Nenhuma das alternativas anteriores está correta.

