MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS EXATAS
DEPARTAMENTO DE EXPRESSÃO GRÁFICA
Professor: Paulo Henrique Siqueira
Disciplina: Fundamentos do Webdesign I

PARTE 1: HTML

O design de páginas de internet é similar à editoração eletrônica. A criação de sites deve aproveitar ao máximo os recursos da Web, respeitando suas limitações. O objetivo principal do estudo de Web Design é a criação de layouts interessantes que chamem a atenção de visitantes.

1. TAGS DE TEXTO

Vamos começar com as tags mais comuns de HTML para editoração de páginas. Vamos utilizar o aplicativo **Bloco de Notas** e um navegador para visualização das páginas (Chrome, Internet Explorer, Firefox, Safari, Opera).

Sites para utilizarmos cores em código hexadecimal são os seguintes:

http://html-color-codes.info/Codigos-de-Cores-HTML/

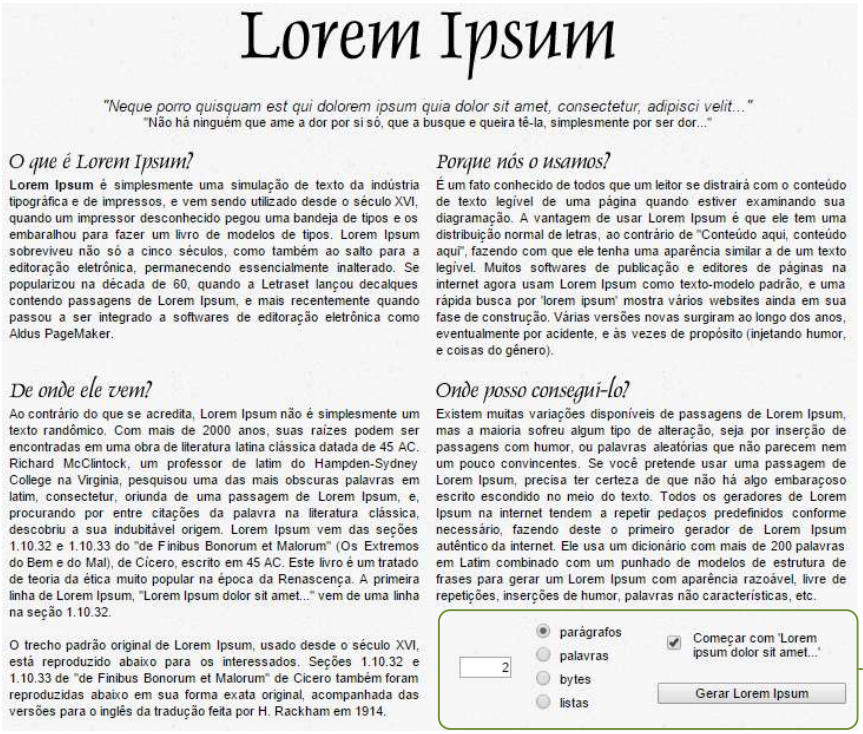
https://color.adobe.com/pt/create/color-wheel/

https://celke.com.br/artigo/tabela-de-cores-html-nome-hexadecimal-rgb

https://html-color.codes

Com o objetivo de criar apenas layouts, sem se preocupar com os conteúdos de texto de um site, podemos acessar o site:

http://br.lipsum.com/



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque vestibulum bibendum posuere. Nulla blandit fermentum mi. In euismod dolor ligula, sit amet pellentesque ipsum venenatis a. Fusce a ipsum diam. Fusce sem magna, tempus fringilla arcu nec, interdum ultrices turpis. In euismod massa sollicitudin finibus ultricies. Nullam faucibus elit non semper porta. Curabitur nec leo urna. Phasellus dignissim consequat dui, dictum maximus turpis tristique at. Vivamus cursus, leo ac sagittis blandit, est sem lacinia nibh, id sagittis ipsum dui facilisis augue. Fusce at nulla et justo lacinia egestas in eu ante. Fusce lacinia pulvinar urna ut venenatis. Fusce ac lorem eget tellus malesuada lobortis nec in neque. Sed sit amet pharetra leo.

Proin finibus diam at imperdiet condimentum. Nullam at dolor dolor. Phasellus a nisi id enim semper facilisis. Cras at dui turpis. Suspendisse convallis nisl leo, vel placerat est venenatis at. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur tristique arcu in turpis tempor tincidunt. Ut varius nisl sed ligula convallis varius. Proin efficitur eleifend laoreet. Etiam orci nulla, ornare in fringilla eu, lacinia in turpis.

Se colocarmos este texto no **Bloco de Notas** e salvar o arquivo com extensão htm, a visualização será a seguinte:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque vestibulum bibendum posuere. Nulla blandit fermentum mi. In euismod dolor ligula, sit amet pellentesque ipsum venenatis a. Fusce a ipsum diam. Fusce sem magna, tempus fringilla arcu nec, interdum ultrices turpis. In euismod massa sollicitudin finibus ultricies. Nullam faucibus elit non semper porta. Curabitur nec leo urna. Phasellus dignissim consequat dui, dictum maximus turpis tristique at. Vivamus cursus, leo ac sagittis blandit, est sem lacinia nibh, id sagittis ipsum dui facilisis augue. Fusce at nulla et justo lacinia egestas in eu ante. Fusce lacinia pulvinar urna ut venenatis. Fusce ac lorem eget tellus malesuada lobortis nec in neque. Sed sit amet pharetra leo. Proin finibus diam at imperdiet condimentum. Nullam at dolor dolor. Phasellus a nisi id enim semper facilisis. Cras at dui turpis. Suspendisse convallis nisl leo, vel placerat est venenatis at. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur tristique arcu in turpis tempor tincidunt. Ut varius nisl sed ligula convallis varius. Proin efficitur eleifend laoreet. Etiam orci nulla, ornare in fringilla eu, lacinia in turpis.

Para separar os parágrafos usaremos a nossa primeira tag: ** **

** Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque vestibulum bibendum posuere. Nulla blandit fermentum mi. In euismod dolor ligula, sit amet pellentesque ipsum venenatis a. Fusce a ipsum diam. Fusce sem magna, tempus fringilla arcu nec, interdum ultrices turpis. In euismod massa sollicitudin finibus ultricies. Nullam faucibus elit non semper porta. Curabitur nec leo urna. Phasellus dignissim consequat dui, dictum maximus turpis tristique at. Vivamus cursus, leo ac sagittis blandit, est sem lacinia nibh, id sagittis ipsum dui facilisis augue. Fusce at nulla et justo lacinia egestas in eu ante. Fusce lacinia pulvinar urna ut venenatis. Fusce ac lorem eget tellus malesuada lobortis nec in neque. Sed sit amet pharetra leo. **

** Proin finibus diam at imperdiet condimentum. Nullam at dolor dolor. Phasellus a nisi id enim semper facilisis. Cras at dui turpis. Suspendisse convallis nisl leo, vel placerat est venenatis at. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur tristique arcu in turpis tempor tincidunt. Ut varius nisl sed ligula convallis varius. Proin efficitur eleifend laoreet. Etiam orci nulla, ornare in fringilla eu, lacinia in turpis. **

A visualização da página com estas tags fica assim:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque vestibulum bibendum posuere. Nulla blandit fermentum mi. In euismod dolor ligula, sit amet pellentesque ipsum venenatis a. Fusce a ipsum diam. Fusce sem magna, tempus fringilla arcu nec, interdum ultrices turpis. In euismod massa sollicitudin finibus ultricies. Nullam faucibus elit non semper porta. Curabitur nec leo urna. Phasellus dignissim consequat dui, dictum maximus turpis tristique at. Vivamus cursus, leo ac sagittis blandit, est sem lacinia nibh, id sagittis ipsum dui facilisis augue. Fusce at nulla et justo lacinia egestas in eu ante. Fusce lacinia pulvinar urna ut venenatis. Fusce ac lorem eget tellus malesuada lobortis nec in neque. Sed sit amet pharetra leo.

Proin finibus diam at imperdiet condimentum. Nullam at dolor dolor. Phasellus a nisi id enim semper facilisis. Cras at dui turpis. Suspendisse convallis nisl leo, vel placerat est venenatis at. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur tristique arcu in turpis tempor tincidunt. Ut varius nisl sed ligula convallis varius. Proin efficitur eleifend laoreet. Etiam orci nulla, ornare in fringilla eu, lacinia in turpis.

A estrutura básica de uma página em HTML é a seguinte:

```
<!DOCTYPE html>
<html>
<head>
      <title>TÍTULO DA PÁGINA</title>
</head>
<body>
      CONTEÚDO DA PÁGINA
</body>
</html>
```

A tag meta pode ser usada para colocar palavras-chave, autores e descrição da página. Esta tag pode ser colocada no cabeçalho da página.

```
<meta name="author" content="AUTOR DA PÁGINA">
<meta name="keywords" content="HTML, PÁGINA">
<meta name="description" content="Minha primeira página">
<meta charset="UTF-8">
```

Para mudar a cor de fundo da página, ou colocar uma imagem de fundo podemos colocar os atributos bgcolor ou background na tag <body>:

```
<body bgcolor="gray">
<body background="imagens/fundo.jpg">
<body text="green">
```

Vamos criar uma pasta para as imagens

O atributo text serve para padronizar a cor do texto da página

Dentro de uma tag podemos colocar atributos de cores, fontes, alinhamentos e planos de fundo. Vamos tentar trocar os atributos de uma tag de parágrafo:

```
<p align="left">Parágrafo à esquerda</p>
<p align="right">Parágrafo à direita</p>
<p align="center">Parágrafo ao centro</p>
<p align="justify">Parágrafo justificado</p>

<p><font size="5">Fonte tamanho 5</font></p>
<p><font color="blue" size="3">Texto em azul de tamanho 3</font></p>
<p><font face="Verdana">Texto com fonte Verdana</font></p>
```

As tags <hr> e
 servem para definir uma linha horizontal e uma quebra de linha, respectivamente. Podemos colocar atributos de cores e tamanho na tag <hr> e estas tags não precisam de fechamento (tipo </br>).

```
<hr>
<hr width="50%">
<br>
```

Para alterar a altura e a cor da barra <hr>, usamos as propriedades size e color, respectivamente:

```
<hr size="10px" color="red">
```

As tags <h1>, <h2>,... <h6> são usadas para definir cabeçalhos, com tamanhos diferenciados dos que usamos em parágrafos normais.

```
<h1>Cabeçalho de tamanho 1</h1>
<h2 align="right">Cabeçalho de tamanho 2 alinhado à direita</h2>
```

A tag <blockquote> cria um bloco de texto em destaque, com margem diferente do parágrafo normal. Já as tags <i>, <b> e <u> servem para deixar o texto em itálico, negrito e sublinhado, respectivamente.

```
<p><u>Parágrafo normal com texto sublinhado</u></p>
<blockquote><i>Bloco de texto <b>destacado</b> em itálico</i></blockquote>
```

A tag <strike> serve para deixar uma palavra riscada. Outras tags que podemos usar são <del> e <s>

```
<p><strike>Palavra</strike> riscada</p>          ou          <p><del>Palavra</del> riscada</p>
```

2. TAGS DE LISTAS

Para criar listas numeradas ou com marcadores vamos criar uma nova página e colocar o título de LISTAS. As tags são <ul> para marcadores e <ol> para as listas numeradas. Os atributos de alinhamento e cores funcionam do mesmo jeito que fizemos com as tags anteriores.

```
<!DOCTYPE html>
<html>
<head>
	<title>LISTAS</title>
	<meta name="author" content="AUTOR DA PÁGINA">
	<meta name="keywords" content="HTML, LISTAS">
	<meta name="description" content="Minha primeira página">
	<meta charset="UTF-8">
</head>
<body>
	<h3>Minha página de listas</h3>
	<ul>
		<li>Item amarelo</li>
		<li>Item azul</li>
		<li>Item verde</li>
	</ul>
	<ol>
		<li>Item cinza</li>
		<li>Item branco</li>
		<li>Item preto</li>
		<li>Item vermelho</li>
	</ol>
</body>
</html>
```

UTF-8 ou ISO-8859-1

Para modificar os atributos das listas com marcadores usamos type com os valores circle, disc ou square:

```
<ul type="square">
	<li>Item amarelo</li>
	<li>Item azul</li>
	<li>Item verde</li>
</ul>
```

Para modificar os atributos das listas numeradas usamos type com os valores A ou I, e podemos escolher também em que número a lista começa a contar com o atributo start:

```
<ol type="A" start="4">
	<li>Item cinza</li>
	<li>Item branco</li>
	<li>Item preto</li>
	<li>Item vermelho</li>
</ol>
```

3. IMAGENS

Para inserir imagens vamos criar uma nova página e colocar o título de IMAGENS. A tag é <img> seguida de atributos de localização, alinhamento e espaçamento das imagens inseridas.

O atributo que mostra o caminho que a imagem salva será mostrada é **src (source)**, o espaçamento horizontal é **hspace** e o espaçamento vertical **vspace**.

O atributo **align** serve para deixar o texto alinhado no topo (**top**), na base (**bottom**), no centro (**middle**) à esquerda (**left**) ou direita (**right**). Sem este atributo, o texto fica alinhado na base do texto.

O atributo **width (largura)** é muito importante, pois deixa a imagem com um tamanho fixo (se usado com pixels) ou em percentual, que pode dimensionar a imagem de acordo com o tamanho de tela do dispositivo usado para visualizar a página.

O atributo **height (altura)** também pode ser usado para dimensionar a imagem.

Outras tags que podem auxiliar na inserção de imagens são <figure> e <figcaption>.
Segue abaixo um exemplo de página com inserções de imagens.

```
<!DOCTYPE html>
<html>
<head>
	<title>IMAGENS</title>
	<meta name="author" content="AUTOR DA PÁGINA">
	<meta name="keywords" content="HTML, IMAGENS">
	<meta name="description" content="Minha primeira página">
	<meta charset="ISO-8859-1">
</head>
<body>
	<h3>Minha página com imagens</h3>
	<p>Vamos testar se a imagem está alinhada com o texto.</p>

	<p>Esta imagem <img src="imagens/figura1.jpg" width="30%" align="left"> está alinhada à esquerda
	do texto.</p>

	<p>Esta imagem <img src="imagens/figura2.jpg" width="80%" align="middle" hspace="30" vspace="60">
	está alinhada no meio do texto.</p>

	<figure align="center">
		<img src="imagens/figura3.jpg" width="100%">
		<figcaption>Esta é a legenda da imagem</figcaption>
	</figure>
</body>
</html>
```

4. LINKS, ÁUDIO, VÍDEO e LISTAS SELETORAS

Os **links** de uma página utilizam a tag **<a>**, que se refere a âncoras de hipertexto ou hiperlinks. Esta tag pode ser usada para referência de uma página do site que você criou ou de uma página externa.

O atributo **href** é o endereço do link e **target** é o destino do link quando for aberto pelo visitante do seu site. Quando for omitido, o target define que a página do link será aberta na mesma aba da página que contém o link. Caso queira que ela abra em uma nova aba, basta usar o valor "_blank" para o target.

O link a seguir define que a pagina1.htm será aberta em uma nova aba. Vale lembrar que para o link funcionar, esta página deve estar no mesmo diretório da página que contém o link.

```
<a href="pagina1.htm" target="_blank" title="Minha página">Página 1</a>
```

O link a seguir define que a página do Google será aberta na mesma aba da página do link.

```
<a href="http://www.google.com.br" title="Google">Página do Google</a>
```

Para mudar a cor do texto de links e de links visitados, basta colocar os atributos **link** e **vlink** na tag **<body>** da página:

```
<body link="green" vlink="red">
```

Os arquivos de **áudio** e **vídeo** podem ser inseridos em uma página de forma similar com tags **<source>** **<audio>** e **<video>**.

```
<audio src="audio/musica1.mp3" controls autoplay></audio>
```

O atributo **loop** pode ser usado para repetir a música.

Nem todos os navegadores conseguem abrir os arquivos mp3, logo podemos criar uma tag com arquivos mp3 e ogg.

```
<audio controls autoplay>
	<source src="audio/musica1.mp3">
	<source src="audio/musica1.ogg">
</audio>
```

Se em nenhum dos casos o áudio funcionar, você pode criar um link para download do arquivo de áudio depois de fechar a tag <audio> ou criar uma mensagem em um parágrafo dentro da tag de áudio dizendo que o navegador não reconhece os arquivos de áudio.

Para inserir um vídeo salvo em seu computador usamos a tag <video>, de forma similar à tag <audio>.

```
<video src="videos/video1.mp4" controls></video>
```

Os atributos de dimensionamento (largura e altura), autoplay, preload (auto ou none) e loop podem ser usados nesta tag de vídeo.

Para que todos os navegadores reconheçam e possam executar seus vídeos, os formatos devem ser ogv, mp4 e webm. A tag completa para vídeos fica da seguinte forma:

```
<video width="100%" height="100%" controls autoplay>
	<source src="videos/video1.mp4" type="video/mp4">
	<source src="videos/video1.ogv" type="video/ogv">
	<source src="videos/video1.webm" type="video/webm">
</video>
```

Caso o navegador não reconheça os arquivos, você pode criar um link para download dos arquivos ou colocar um parágrafo dentro da tag vídeo com uma mensagem dizendo que o navegador não reconhece os vídeos.

Os vídeos da internet podem ser inseridos em uma página HTML através de links ou usando a tag **<iframe>**, que cria uma tela de exibição do vídeo embutida na página. Os atributos são os mesmos da tag **<img>**.

Um atributo adicional chama-se allowfullscreen, que permite a exibição do vídeo em tela cheia. Outro atributo chama-se frameborder, que cria uma borda na tela do vídeo.

```
<iframe width="420" height="315" src="https://www.youtube.com/embed/1cQh1ccqu8M" frameborder="1"
allowfullscreen></iframe>
```

Atividade 1: Crie uma página com um tema comum. Esta página deve conter banner, imagens, uma lista, imagem de fundo e formatações de texto com cores e alinhamentos.

5. TABELAS

A criação de tabelas nas páginas HTML permite algumas aplicações interessantes, que vão além de uma simples tabela de dados. Muitas vezes podemos usar a estrutura de tabela para criar um layout de página, uma paleta de cores ou galerias de imagens e vídeos.

As principais tags são <table>, que cria a tabela, <tr> que define uma linha da tabela, <td> que define uma célula da tabela e <th> que define uma célula de título da tabela.

Os atributos de alinhamento, cores e texto são os mesmos que vimos para parágrafos.

Na primeira linha da tabela podemos definir a largura de cada coluna, fontes, alinhamentos, cores e bordas (**border**).

Os atributos **rowspan** e **colspan** servem para mesclar linhas ou colunas, respectivamente. Um exemplo de estrutura simples de tabela com 3 linhas e 4 colunas é o seguinte:

```
<!DOCTYPE html>
<html>
<head>
	<title>TABELAS</title>
	<meta name="author" content="AUTOR DA PÁGINA">
	<meta name="keywords" content="HTML, TABELAS">
	<meta name="description" content="Minha primeira página">
	<meta charset="ISO-8859-1">
</head>
<body>
	<h3>Minha página com tabelas</h3>
	<table border="2" bordercolor="red" cellpadding="10" cellspacing="5">
		<tr bgcolor="green">
			<th>Título 1</th>
			<th>Título 2</th>
			<th>Título 3</th>
			<th>Título 4</th>
		</tr>
		<tr>
			<td>Linha 2, Coluna1</td>
			<td>Linha 2, Coluna2</td>
			<td>Linha 2, Coluna3</td>
			<td>Linha 2, Coluna3</td>
		</tr>
		<tr>
			<td>Linha 3, Coluna1</td>
			<td>Linha 3, Coluna2</td>
			<td>Linha 3, Coluna3</td>
			<td>Linha 3, Coluna3</td>
		</tr>
	</table>
</body>
</html>
```

Os atributos **cellpadding** e **cellspacing** servem para definir a margem interna e externa de cada célula da tabela. O atributo **bordercolor** muda a cor da borda.

Exercício 1: Crie uma galeria de imagens usando uma tabela no fim da página criada na Atividade 1. Para cada imagem colocada, crie um link para que abra em uma nova aba. Assim temos uma galeria de imagens bem simples, com abertura da imagem completa em uma nova aba.

Exercício 2: Crie uma galeria de vídeos no site da Atividade 1, com arquivos mp4 ou vídeos do youtube. Crie também uma galeria de áudios neste site.

Exercício 3: Crie uma linha de título, que ocupe a primeira linha da tabela. Crie uma tabela com fundos de imagens com as informações da tabela a seguir.

GRUPO	ALIMENTO	MEDIDA	QUANTIDADE
VERDURA	Hortaliça crua	180g	1 pires cheio
	Hortaliça cozida	140g	1 pires cheio
LEGUMES	Cenoura ou abóbora	80g	1 unidade
CEREAIS	Pão	50g	1 unidade
	Bolacha	35g	6 unidades
	Cereais	35g	5 col. (sopa) de aveia
FRUTAS	Variadas	160g	1 maçã, 1 banana, 1/2 papaya 2 col. (sopa) de passas 1 xíc. (chá) de uvas 1 manga pequena
LEGUMINOSAS	Ervilha ou Grão de Bico	80g	1/2 xícara (chá)
	Feijão ou Lentilha	85g	1/2 xícara (chá)
LATICÍNIOS	Leite	250ml	1 copo (pequeno)
	Iogurte líquido	200g	1 pote
PROTEÍNAS	Carnes ou pescados cozidos	25 g	80g de carne de boi, porco, frango ou peixe ou 1 ovo
ÓLEOS	Manteiga ou margarina	5g	1 colher (sopa)
	Azeite ou óleo	10g	1 colher (sobremesa)

Exercício 4: Crie uma página que contém a tabela a seguir.

Região	Porcentagem de pessoas que fumam diariamente		Porcentagem de ex-fumantes que fumavam diariamente	
	Homens	Mulheres	Homens	Mulheres
Brasil	18,9	11,6	16,1	10,7
Norte	17,2	8,6	14,5	10,2
Nordeste	18,1	10,4	14,7	12,1
Sudeste	19,0	12,3	16,5	9,9
Sul	21,1	13,9	17,9	10,4
Centro-Oeste	19,0	10,7	17,2	11,0

Fonte: Brasil, Tabaco e Câncer de Pulmão: O que dizem os Números, André Medici, junho de 2013.

Exercício 5: Crie uma página com a paleta de cores básicas de HTML mostrada a seguir. Cada nome de cor pode ser colocado em uma célula com sua respectiva cor de fundo, ou em uma célula separada, logo abaixo da sua respectiva cor de fundo. Crie uma célula de título da página e adicione mais uma linha com as seguintes cores: GreenYellow, MediumVioletRed, PaleTurquoise, PapayaWhip, SandyBrown, ForestGreen, FireBrick.

Red	Blue	Green	Yellow	Orange	Purple	Pink
Chocolate	Cyan	Gray	Black	White	Beige	Gold
Aqua	Azure	Bisque	Magenta	Coral	Chatreuse	Salmon
Aquamarine	Blanch	Dalmond	Brown	Burlywood	Cadet	Chat
CadetBlue	Blanchedalmond	Cornsilk	Darkblue	Goldenrod	Indigo	Lavander
Darkgoldenrod	Floral	Gainsboro	Hotpink	Khaki	Lawngreen	Turquoise
Indian	Indianred	Korchid	Darkorchid	Honeydew	Dimgray	Ivory
Blueviolet	Crimson	Lemonchiffon	Deepink	Lavanderblue	Lightpink	Lightblue
Lightgreen	FloralWhite	Darkolivegreen	Dodgerblue	Ghostwhite	Lightcoral	Deepskyblue

Fonte: Web Design para iniciantes, Neilon Marcio.

Para usar os alinhamentos horizontais nas células de uma tabela usamos o atributo **align**. Para o alinhamento vertical, podemos usar o atributo **valign (bottom, top, middle)**.

Vamos criar um primeiro site com as páginas em formato de tabela para construir seus layouts. Esta técnica servirá como base para criarmos os sites com Folhas de Estilos CSS que veremos adiante.

Atividade 2: Crie uma página de restaurante, usando um layout com banner, informações de cardápio e preços no site. Utilize imagens e a tabela criada no Exercício 3 para criar uma segunda página do site, com mesmo layout da primeira. Crie links para que os visitantes consigam acessar as duas páginas.

A criação de **formulários e listas seletoras** em HTML utiliza o elemento **input** para validar informações digitadas pelos visitantes de sua página. Podemos criar listas de seleção e validar campos de email, telefone e outros elementos contidos em formulários.

Segue abaixo um exemplo de lista seletora com as tags **<select>** e **<option>**, e o formato padrão de uma linha visível (primeira opção da lista).

```
<select id="marca" class="texto" name="carro">
        <option value="fiat">Fiat</option>
        <option value="ford">Ford</option>
        <option value="volkswagen">Volkswagen</option>
        <option value="renault">Renault</option>
</select>
```

Para que deixar um item pré-selecionado usamos a propriedade **selected**. Para deixar um item desabilitado usamos a propriedade **disabled**.

```
<select id="marca" class="texto" name="carro">
        <option value="fiat">Fiat</option>
        <option value="ford" selected>Ford</option>
        <option value="volkswagen">Volkswagen</option>
        <option value="renault" disabled>Renault</option>
</select>
```

Para deixar a lista seletora em um formato de lista com mais itens visíveis, usamos a propriedade **size**.

```
<select id="marca" class="texto" name="carro" size="3">
        <option value="fiat">Fiat</option>
        <option value="ford" selected>Ford</option>
```

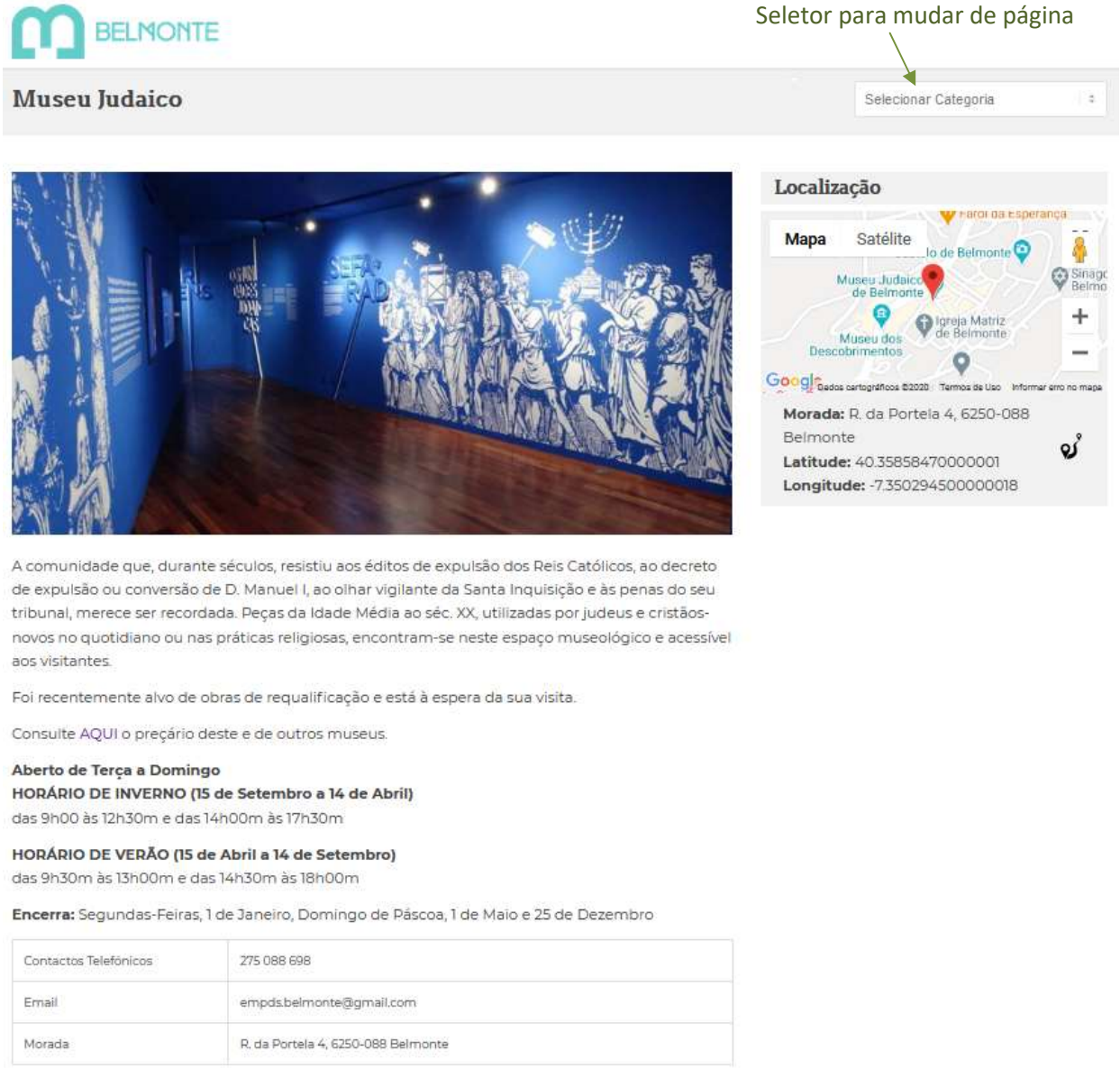
```
        <option value="volkswagen">Volkswagen</option>
        <option value="renault">Renault</option>
</select>
```

Podemos incluir um rótulo (label) que fique posicionado próximo da lista. Um exemplo interessante é do uso de javascript para inserir opções de imagens ou vídeos dentro de um iframe. Segue abaixo o exemplo com 3 fotos, e a caixa seletora é usada para mudar as imagens do iframe.

```
<label for="imagens">Escolha uma imagem da galeria:</label>
<select id="fotos" class="midia" name="imagens" onchange="javascript:iframe1.location=this.value">
        <option value="imagens/foto1.jpg">Foto 1</option>
        <option value="imagens/foto2.jpg">Foto 2</option>
        <option value="imagens/foto3.jpg">Foto 3</option>
        <option value="imagens/foto4.jpg">Foto 4</option>
</select>

<iframe src="imagens/foto1.jpg" name="iframe1" width="70%" height="400px"></iframe>
```

Atividade 3: Crie um site com as páginas com imagens e descrições de 5 pontos turísticos, com layout em formato de tabela (exemplo a seguir). As informações podem incluir preço de ingresso, endereço, link para o site oficial de cada ponto escolhido e link para o mapa. Em cada página, crie um botão de seleção que mudará o conteúdo da página de um ponto turístico para outro. Coloque fotos e a localização de cada ponto turístico com o iframe do Google Maps.



6. CSS – CASCATING STYLE SHEET

O padrão de formatação CSS (Cascating Style Sheet), também chamado de folha de estilos, permite a formatação de itens de uma página de uma só vez. Este padrão também permite que diversas páginas tenham um mesmo padrão para determinadas tags HTML.

A sintaxe é a seguinte:

```
objeto {propriedade:valor;}
```

Exemplos:

```
body {background:#E0F8F7;}
table {border:1px solid green;}
```

Fundo da página com a cor

Bordas de tabelas com 1px de largura, linha contínua e na cor verde

O CSS pode ser inserido na página de três formas: inline (logo após a tag), interno (dentro da tag de cabeçalho da página) ou externo (um arquivo com extensão **.css** referenciado na tag de cabeçalho).

Exemplo inline:

```
<p style="color:#01DF01; font-size:18px;">Texto verde com fonte tamanho 18</p>
```

Exemplo interno (dentro da tag <head>):

```
<style type="text/css">
        p {color:#01DF01; font-size:18px;}
</style>
```

Exemplo externo (dentro da tag <head>):

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

No caso de css em arquivo externo, basta escrever dentro do arquivo **estilo.css** todas as formatações css:

```
p {color:#01DF01; font-size:18px;}
```

Para elementos das tags do HTML usamos sempre o nome da tag (p, body, select, img, h1, h2, a, audio,...).

As **unidades** usadas para CSS incluem algumas ótimas opções. Além das unidades pixels (**px**), ponto (**pt**) e de percentual (**%**), podemos utilizar mais 3 opções descritas a seguir.

A unidade **em** herda o tamanho do elemento "pai" multiplicado por um número. No exemplo a seguir, o tamanho da fonte de um parágrafo torna-se 24px (pois a unidade 2em multiplica o tamanho da tag "pai" por 2), enquanto que as células **td** de cada tabela ficam com as fontes de tamanho 6px. Esta regra vale para as tags imediatamente abaixo às tags "pai".

```
body {font-size:12px;}
p {font-size:2em;}
td {font-size:0.5em;}
```

No caso de várias tags aninhadas, como em uma tag de célula de tabela **td** que sempre está dentro de uma tag **table**, as medidas podem ser multiplicadas várias vezes. No exemplo a seguir, a tag **td** tem tamanho de fonte multiplicado por 4, ou seja, 48px.

```
body {font-size:12px;}
table, td {font-size:2em;}
```

A unidade **vw** (viewport width) usa um percentual relativo ao tamanho da largura total da janela do navegador. Já a unidade **vh** (viewport height) usa um percentual relativo ao tamanho da altura total do navegador.

Estas unidades podem ser usadas de forma parecida com o percentual em tamanhos de fontes, iframes, alturas de imagens e outros elementos.

```
img {width:20vw;}
iframe {width:100vw; height:100vh;}
```

Os atributos de fonte podem ser usados de forma separada, como mostra o exemplo a seguir:

```
table {font-size:14px; font-family:Verdana, Tahoma, Courier; font-style:italic; font-weight: bold;}
```

Quando for usada a propriedade font, a ordem dos elementos é a seguinte:

```
font:font-style font-weight font-size font-family;

p {font:italic bold 15px Courier, Verdana;}
```

Exercício 7: Crie uma página com layout em formato de tabela para formatar elementos usando CSS.

Uma propriedade que podemos usar no CSS define o corpo da página centralizado à tela. Neste caso, podemos definir a largura da página menor do que 100% e as margens da seguinte maneira:

```
body {width:85%; margin:auto;}
```

Para configurar o texto de uma página com parágrafo indentado, podemos usar a propriedade **text-indent** com qualquer unidade mostrada anteriormente. No exemplo a seguir, o primeiro parágrafo que aparece na página fica sem indentação, e os próximos ficam com **1.5em** de indentação.

```
p {margin-bottom:10px; text-align:left;}
p + p {text-indent:1.5em;}
```

Quando criamos vários elementos do mesmo tipo e queremos que todos tenham uma determinada formatação (por exemplo, parágrafos, cabeçalhos h1, divs, img,...) criamos as classes (**class**) para estes grupos de tags.

Por exemplo, uma tabela e um parágrafo podem compartilhar das mesmas formatações colocando-se em suas tags a mesma classe:

```
<p class="fonteVermelha">Parágrafo com fonte vermelha e fundo azul</p>
<table class="fonteVermelha">
        ...
</table>
```

E no CSS colocamos as propriedades da classe **fonteVermelha**:

```
<style type="text/css">
        .fonteVermelha {color:red; background:blue;}
</style>
```

Quando usamos os identificadores (**id**), a formatação do CSS deve ser usada exclusivamente para um determinado elemento. Os **ids** não podem se repetir em uma página. Caso estejam repetidos, o padrão é que a página coloque a formatação apenas no primeiro elemento que aparecer com **id**.

Por exemplo, para criar um parágrafo com texto em vermelho e outro com texto em verde, colocamos **ids** em cada parágrafo no HTML:

```
<p id="paragrafoVermelho">Parágrafo com fonte vermelha</p>
<p id="paragrafoVerde">Parágrafo com fonte verde</p>
```

No CSS colocamos as propriedades de cada parágrafo:

```
<style type="text/css">
	#paragrafoVermelho {color:red;}
	#paragrafoVerde {color:green;}
</style>
```

Para modificarmos uma propriedade de um elemento que está dentro de uma tag do HTML criamos um seletor encadeado. Por exemplo, para mudar a cor do negrito de um parágrafo, usamos a seguinte tag HTML com respectivo css:

```
<p>Este é o meu <b>principal</b> parágrafo</p>
```

Dentro da tag <head>:

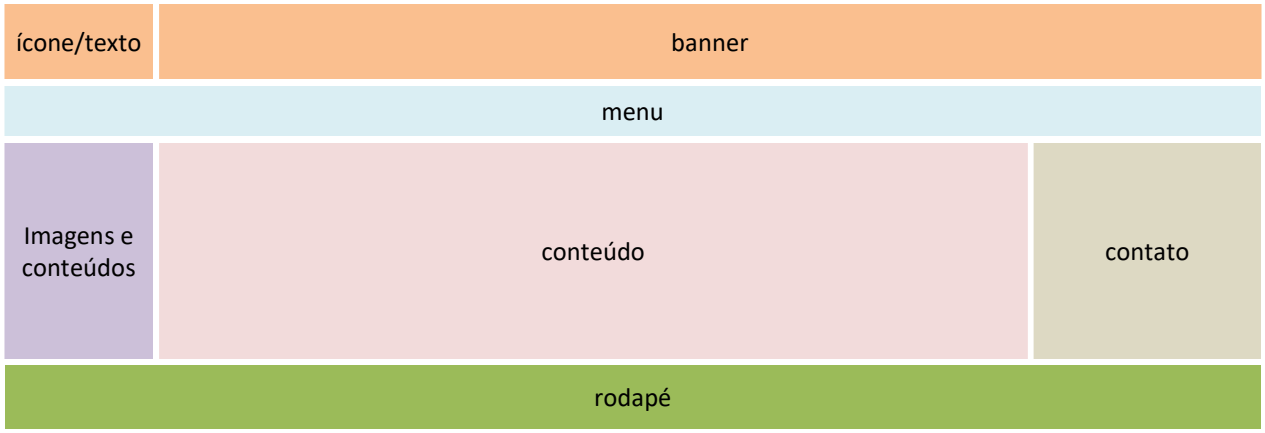
```
<style type="text/css">
	p b {color: blue;}
</style>
```

A mesma ideia vale para **id** e **class**. Para facilitar as formatações sempre usamos as tags ** ** para separar elementos da página com atributos diferentes. Para criar uma página com menu, banner, logo e corpo de texto, podemos criar 4 tags ** ** com os respectivos conteúdos e seus ids ou classes. Desta forma, a formatação do CSS ficará mais simples e eficiente. As pseudo-classes do CSS podem ser usadas para criar propriedades que não são editáveis em HTML. Estes seletores têm formatos sempre separados por dois pontos :

```
a:hover {font-weight:bold;}
a:link {color:red; text-decoration:none;}
a:visited {color:green;}
```

Vamos criar um site com formato em **tabela**, com layout abaixo, para a próxima Atividade.

Caixa/table



Podemos usar a estrutura simplificada a seguir para montar o layout, criando classes para o CSS.

```
<!DOCTYPE html>
<html>
<head>
	<title>LAYOUT COM CSS</title>
	<meta name="author" content="AUTOR DA PÁGINA">
	<meta charset="UTF-8">
	<style type="text/css">
		table {color:magenta;}
	</style>
</head>
<body>
	<table>
		<tr>
			<td>Ícone</td>
			<td colspan="2">Banner</td>
		</tr>
```

```
		<tr>
			<td colspan="3">menu</td>
		</tr>
		<tr>
			<td>imagens</td>
			<td>conteúdo</td>
			<td>contato</td>
		</tr>
		<tr>
			<td colspan="3">rodapé</td>
		</tr>
	</table>
</body>
</html>
```

Ao criar um background para a página na tag **body**, podemos fixar a imagem usando as seguintes propriedades do CSS:

```
body {background-image:url(imagens/fundo.png); background-repeat:no-repeat;
	background-position:right top; background-attachment:fixed;}
```

Desta forma, a imagem de fundo fica fixa ao rolarmos a página, posicionada na parte de cima da tela, alinhada horizontalmente à direita. Experimente outras configurações no site que estamos construindo.

Porém, se você quiser que a imagem de fundo "role" junto com o conteúdo, basta usar a propriedade:

```
background-attachment:scroll;
```

Se quiser repetições na horizontal ou na vertical, as respectivas propriedades são as seguintes:

```
background-repeat:repeat-x;
ou
background-repeat:repeat-y;
```

Se você quiser ampliar a imagem para cobrir a tela toda, basta usar a propriedade background-size:

```
background-size:cover;
```

Outras opções são **auto** (usa o tamanho original da imagem) e **contain** (cobre verticalmente toda a tela).

As células da tabela serão os "boxes" de conteúdos do nosso site. Podemos definir no CSS propriedades de fundo com imagens ou cores. Podemos criar fundos com transparência, para dar algum destaque à imagem de fundo da página, colocado no **body**. Neste caso, usamos a propriedade de cor **rgba(x,y,z,w)**, com atributos de cor que variam de 0 a 256 para as 3 primeiras variáveis, e a última varia entre **0** e **1** e define a transparência da cor (0 transparente e 1 opaco).

No exemplo a seguir, definimos as células da tabela com fundos azuis e 70% de transparência, com margens internas de 10px:

```
td {background-color:rgba(0,0,256,0.7); padding:10px;}
```

Para criar o banner, selecione uma imagem e coloque-a como imagem de fundo da célula com classe definida como **banner**. Defina propriedades de fontes, cores e do background para esta classe. Assim, o título da página pode ser criado por cima da imagem.

```
.banner {background:url(imagens/banner.jpg) no-repeat center; background-size:cover;}
```

Um efeito que pode ser usado no texto do banner é o preenchimento da fonte de uma cor, e o contorno com outra cor. Os comandos para espessura e cor de um contorno de fonte são os seguintes:

```
-webkit-text-stroke-width:1px; -webkit-text-stroke-color:red;
```

Outro efeito que podemos criar no texto do banner é de sombra com a propriedade text-shadow:

```
text-shadow:medida1 medida2 medida3 cor;
```

onde medida1 e medida2 são as distâncias à direita e abaixo do texto, medida3 é o raio para um efeito blur (esfuminho) da sombra e a cor em rgb, hexadecimal ou código HTML.

No exemplo a seguir, criamos uma sombra com 2px à direita do texto, -2px abaixo do texto, raio de efeito blur de 5px e cor vermelha para os títulos h1 da página.

```
h1 {text-shadow:2px -2px 5px red;}
```

Para criar mais de uma sombra no mesmo texto, você pode colocar as medidas da outra sombra logo após as medidas da sombra anterior:

```
text-shadow:medida1 medida2 medida3 cor1, medida4 medida5 medida6 cor2;
```

Podemos criar um ícone, ou usar imagens pictográficas do HTML. Escolha uma imagem para usar como ícone:

http://www.degraf.ufpr.br/docentes/paulo/webdesign/pictograph.html

Agora vamos criar um menu horizontal usando a tag **<nav>** que indica uma lista de links internos ou externos. Vamos criar uma **classe menu** para esta tag para colocar suas propriedades CSS.

Dentro desta tag criamos a lista dos itens do menu com as tags de listas que já utilizamos: **<ul>** e **<li>**.

```
<nav class="menu">
	<ul>
		<li><a href="site1.htm">Primeiro Item</a></li>
		<li><a href="site2.htm">Segundo Item</a></li>
		<li><a href="site3.htm">Terceiro Item</a></li>
	</ul>
</nav>
```

Seguem abaixo as propriedades CSS do menu. A linha da tabela **<tr>** que contém o menu pode ter a classe **menu** para ser formatada no CSS. Podemos colocar borda nesta linha **<tr>** e mudar a cor de fundo para o menu:

```
.menu {background:yellow; border-top:blue 1px solid;}
```

Em cada item **<li>** do menu podemos definir: margens internas (padding) e externas (margin), largura de cada botão, borda e alinhamento.

```
.menu li {margin:7px; padding:10px; border:1px red solid; width:100px; text-align:center;}
```

Se você quiser deixar uma barra lateral em cada item do **menu**, basta modificar a propriedade **border**:

```
border-right:1px red solid;
```

Para remover a barra do último item da lista, usamos a propriedade **last-child** do CSS:

```
.menu li:last-child {border-right:none;}
```

Se você usar as barras à esquerda, a propriedade css fica assim:

```
border-left:1px red solid;
```

Para remover a barra do primeiro item do menu, usamos a propriedade **first-child**:

```
.menu li:first-child {border-left:none;}
```

Cada item **<li>** com seu respectivo link **<a>** pode ter definições de cor de texto, margem e cor de fundo.

```
.menu li a {color:#666; padding:10px; text-decoration:none;}
```

Para criar um efeito de sombra no texto ao passar o cursor do mouse em cada item do menu, podemos usar a propriedade hover de cada link do menu:

```
.menu li a:hover {color:blue; text-shadow:1px 1px 2px white;}
```

Assim como o efeito de sombra no texto, podemos criar um efeito de sombra em cada item do menu com a propriedade box-shadow:

```
box-shadow:medida1 medida2 medida3 cor;
```

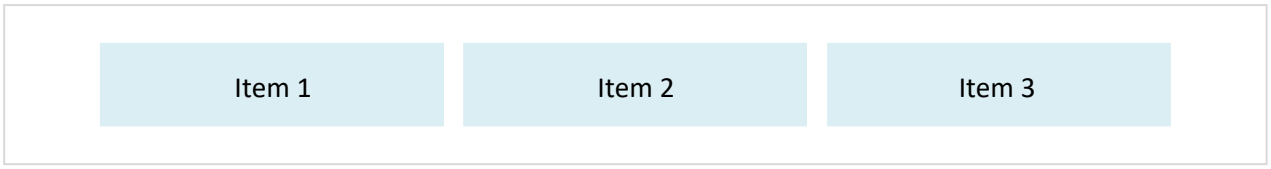
que possui as mesmas informações da propriedade text-shadow. Experimente colocar esta propriedade em outros elementos da página, como o **body**, **td** ou parágrafos **p**.

O formato de visualização da lista **<ul>** pode ser definido com a propriedade de **display block** ou **flex.** Utilizaremos o **display flex**, que é mais fácil de configurar no CSS. A lista **<ul>** do menu, não precisa ter o formato padrão de lista e as margens internas e externas podem ser nulas (senão uma margem padrão de listas será definida).

```
.menu ul {list-style-type:none; margin:0; padding:0; display:flex; justify-content:center; flex-direction:row;}
```

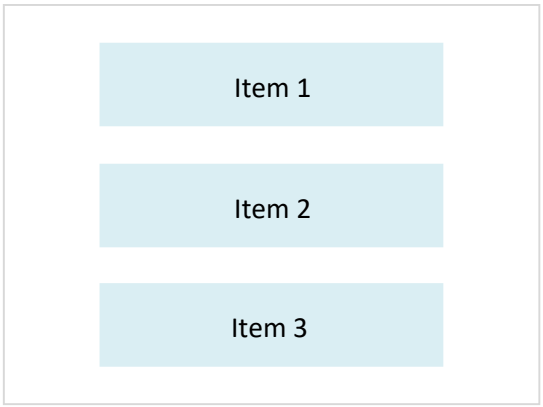
O display **flex** faz com que a lista do menu ocupe o espaço da célula **<td>** com a classe **menu.** Para centralizar o menu, vamos usar o atributo em CSS chamado **justify-content**, que pode ter os valores **flex-start** (equivalente a left), **flex-end** (equivalente a right), **center**, **space-between** (outros objetos da mesma linha ficam com mesmo espaçamento, sem margens) ou **space-around** (outros objetos da mesma linha ficam com mesmo espaçamento e margens).

Os itens do menu ficarão dessa forma:



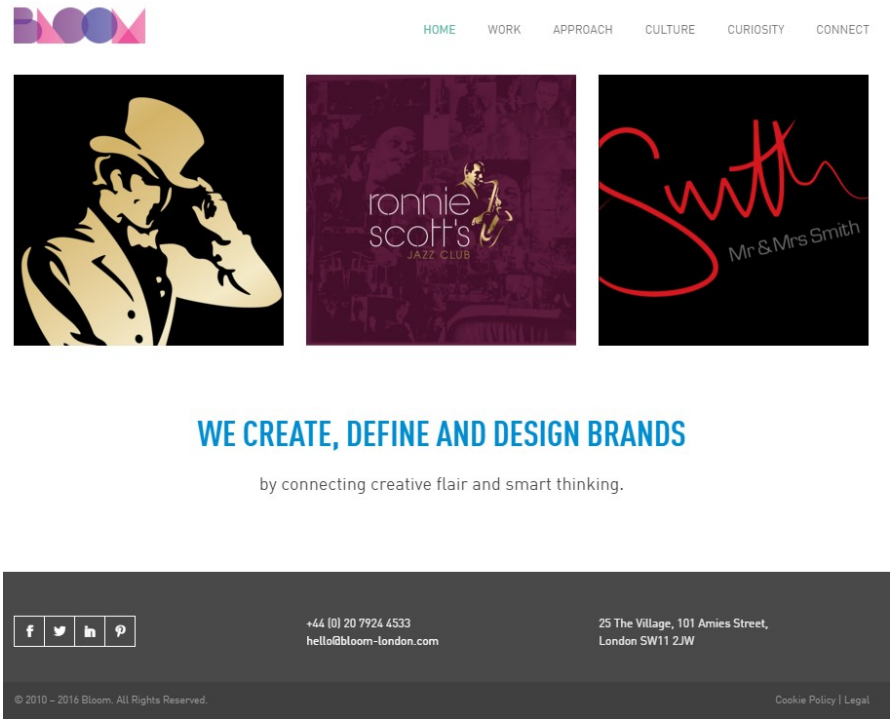
Para mudar a disposição dos itens para o formato vertical, basta modificar a propriedade:

```
flex-direction:column;
```

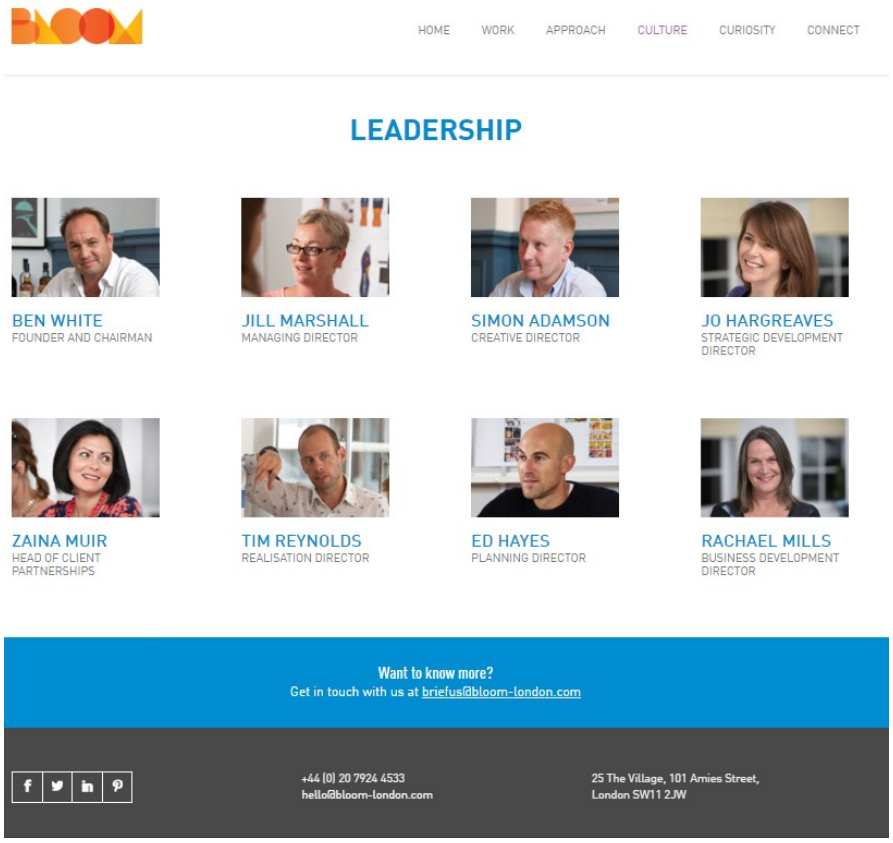


Assim podemos usar este atributo **display:flex** para criarmos menus horizontais e verticais.

O site http://bloom-london.com, com sua versão anterior, mostrava a propriedade de um item ativo do menu com outra cor. Outra mudança é com as cores do ícone (que poderemos desenhar e colorir com os conteúdos que veremos nas aulas seguintes), que podem ser diferentes para cada página criada.



Página principal, com destaque para o item HOME do menu.



Página com a equipe de trabalho, com destaque para o item ativo do menu.

Para criar este efeito na página ativa, podemos criar uma classe chamada **.active** dentro da tag html da página ativa. Depois é só colocar as propriedades CSS desta classe que o resultado fica semelhante ao site mostrado acima.

No código HTML da página **index.htm** colocamos na lista do menu a classe **.active**. Podemos fazer o mesmo para cada página do seu site:

```
<nav class="menu">
	<ul>
		<li><a class="active" href="index.htm">Primeiro Item</a></li>
		<li><a href="site2.htm">Segundo Item</a></li>
		<li><a href="site3.htm">Terceiro Item</a></li>
	</ul>
</nav>
```

No CSS criamos as propriedades da classe active:

```
.active {text-shadow:1px 1px 3px black; color:yellow;}
```

Agora nossa página já está com a parte superior completa. O próximo passo é colocar em uma célula imagens na página, na célula do lado esquerdo do conteúdo da página, e na célula do lado direito do conteúdo podemos inserir informações de contato e de redes sociais.

Imagens dos conteúdos	conteúdo	contato

The Starry Night
Vincent van Gogh 1889

Para criar uma espécie de galeria lateral de imagens, podemos criar as tags de imagem com legenda e formatá-las no **css**. Uma classe "imagem" serve para centralizar imagens dentro da classe de "galeria".

```
<td class="galeria">
	<div class="imagem"><img src="imagens/starry.jpg"></div>
	<br>The Starry Night
	<br>Vincent van Gogh 1889
</td>
.galeria {vertical-align:top; text-align:left;}
.imagem {text-align:center;}
.imagem img {width:100%;}
```

Na célula de contato, podemos criar uma função para enviar email com o programa padrão de cada dispositivo com a seguinte **tag**:

```
<a href="mailto:seuemail@provedor.com.br">Envie um email</a>
```

Outras informações que podem ser colocadas são: mapas, endereço e telefone.

Na última coluna da tabela podemos criar os links para redes sociais e colocar informações de contato do webdesigner e o ano de criação de página.

Para colocar os links de redes sociais basta criar ou copiar imagens dos principais sites de redes sociais e criar os links das redes sociais da empresa do site. Cada imagem terá um link agregado com o endereço da rede social, por exemplo:

```
<a href="https://www.facebook.com/pages/UFPR"><img src="imagens/facebook.png"></a>
<a href="http://www.twitter.com/ufpr"><img src="imagens/twitter.png"></a>
<a href="https://www.instagram.com/ufpr_oficial/"><img src="imagens/instagram.png"></a>
```

Para colocar dois links em um mesmo texto ou imagem, usamos um código em **javascript** para criar o segundo link. Como muitos navegadores possuem bloqueio de pop-up, este segundo link provavelmente não abrirá em todos os navegadores. Por isso, o link mais importante deve ficar no **href**. No exemplo a seguir, uma mesma imagem tem os links de facebook e twitter da UFPR:

```
<a href="http://www.twitter.com/ufpr"
onClick="javascript:window.open('https://www.facebook.com/pages/UFPR')">
<img src="imagens/redesSociais.png"></a>
```

Para abrir o segundo link com uma janela nova, sobreposta à janela do primeiro link, colocamos as dimensões da nova janela no código javascript:

```
<a href="http://www.twitter.com/ufpr"
onClick=" javascript:window.open('https://www.facebook.com/pages/UFPR', 'nome', 'width=900px,
height=800px, resizable=yes, scrollbars=yes, status=yes')"><img src="imagens/redesSociais.png"></a>
```

Você também pode colocar imagens com funções de botões para compartilhar sua página nas redes sociais. Basta colocar o link de cada imagem com as respectivas tags abaixo:

```
<a href="http://www.facebook.com/sharer.php?u=endereçoCompletoDoSeuSite"><img
src="imagens/facebook_compartilhar.png"></a>
<a href="https://twitter.com/intent/tweet?url=endereçoCompletoDoSeuSite%2F&amp;
text=MinhaPágina&amp;hashtags=webdesign,redes%20sociais"><img src="imagens/twitter_compartilhar.png"></a>
```

Para compartilhar a nossa página através do whatsapp, basta criar uma imagem e colocar o endereço do site na opção text, conforme exemplo abaixo.

```
<a href="whatsapp://send?text=Veja o meu site. Segue o link: endereçoCompletoDoSeuSite"
target="_blank"><img src="imagens/whatsapp.png"></a>
```

Para finalizar nossa página, basta formatar a célula de rodapé e inserir conteúdos para cada site. Lembre-se de colocar o mesmo cabeçalho com as propriedades CSS em cada página.

Atividade 4: Usando todos os elementos definidos com nosso segundo site de CSS, crie 3 páginas, cada uma sobre um esporte diferente. Coloque os elementos definidos no layout proposto na página 13, com os links que funcionem entre as 3 páginas construídas. Escolha uma página para ser a principal, com nome **index.htm**. Coloque links para redes sociais, imagens, ícone, menu e banner. Utilize as formatações mostradas a partir da página 11.

7. FORMULÁRIOS

Vamos criar uma página de formulário, usando os campos **type** e botões de validação.
Tel – telefone
Search – campo de busca
Email – formatação e validação de email
url – endereço web
datas e horas (date, month, week, time, datetime-local)

Exemplos:

```
<input type="date" name="data" value="11-04-2016"/>
<input type="datetime-local" name="data" value="11-04-2016"/>
<input type="tel" name="telefone" value="" maxlength="9"/>
<input type="number" name="valor" value="10" max="20" min="1" step="0.1"/>
<input type="range" name="valor" value="7" max="40" min="0" step="0.5"/>
<input type="color" name="color"/>
<input name="Procurar" placeholder="Procurar"/>
```

Podemos criar rótulos para os campos de entrada de um formulário usando a tag **label** ou usar uma tabela para organizar o formulário.

```
<label>Nome:</label>
```

Para validar os campos de um formulário, precisamos criar um arquivo em PHP para enviar os dados para o servidor de hospedagem de seu site ou para o seu email. Desta forma, podemos colocar todos os campos do formulário dentro de uma **tag** form com uma chamada para validação de um arquivo PHP.

```
<form method="post" action='enviar.php">
      ...
</form>
```

Para não deixar os campos do formulário vazios, podemos usar a propriedade **placeholder** em cada tag **input** para deixar uma mensagem da informação esperada de cada campo de entrada.

```
<input name="name" placeholder="Seu Nome">
```

Uma caixa de mensagem com mais linhas pode ser definida usando a tag **textarea**. O exemplo abaixo cria uma caixa de texto com 7 linhas. O número de colunas pode ser definido com a propriedade **cols**.

```
<textarea rows="7" name="mensagem" placeholder="Qual é sua dúvida?"></textarea>
```

Qualquer entrada que seja obrigatória deve ter a propriedade **required** dentro da tag **input**.

```
<input email="email" placeholder="Seu email" required>
```

Para enviar os dados do formulário usamos a propriedade submit.

```
<input id="submit" name="submit" type="submit" value="Enviar">
```

Para formatar os campos do formulário, basta criar as propriedades para cada item textarea ou input no CSS.

Exercício 8: Crie uma página de formulário de contato, formatando os elementos com CSS. Coloque os campos NOME, EMAIL, TELEFONE e MENSAGEM. A validação pode ser feita com um arquivo PHP com código a seguir.

```
<?php
      $name = $_POST['name'];
      $email = $_POST['email'];
      $message = $_POST['message'];
      $fone = $_POST['fone'];
      $from = 'De: Meu site';
      $to = 'paulohs@ufpr.br';
      $subject = 'Dúvida';
      $body = "De: $name\n E-Mail: $email\n Mensagem:\n $message";
      if ($_POST['submit'])
      { if ($name != '' && $email != '')
            { if (mail ($to, $subject, $body, $from))
                  { echo 'Sua mensagem foi enviada!';}
            else {echo 'Algo deu errado, volte e tente novamente!';}
            }
      else {echo 'Você precisa responder todas as questões!!';}
      }
?>
```

Agora vamos criar uma página com itens ajustáveis, criando-se tags aninhadas **div**. Crie um layout com uma tabela com uma linha, atribuindo as propriedades de 2 células: uma delas como barra lateral e a segunda com tags **div** aninhadas:

```
<td class="barra">
	Barra Lateral
</td>
<td class="conteudo">
	<div class="todos">
		<div class="item">
			ITEM 1
		</div>
		<div class="item">
			ITEM 2
		</div>
	</div>
</td>
```

A ideia é que o layout da célula de conteúd fique desta forma:

Para deixar a **div "todos**" alinhada na célula **td** usamos a seguinte propriedade css:

```
.conteudo {vertical-align:center;}
```

O display ajustável é o flex, o mesmo que usamos para criar os menus dos sites anteriores. O espaçamento **space-around** deixa os itens ajustados à medida da largura **width** da **div**. O comando **flex-wrap** (empacotado) permite que os itens filhos mudem de linha quando mudar a largura da tela de visualização do site, tornando-o responsivo (pelo menos nesta tag de conteúdo). Os demais atributos já foram usados nos sites anteriores.

```
.todos {width:100%; background:rgb(75,140,255); text-align:center;
	display:flex; justify-content:space-around; flex-direction:row; flex-wrap:wrap; color:white;}
```

As configurações dos itens podem incluir margens internas e externas (**padding** e **margin**), **flex-wrap**, bordas, e tamanho mínimo de largura **min-width**. O alinhamento pode ter as mesmas opções que usamos no menu: **flex-start**, **flex-end**, **center** ou **stretch**, que permite "esticar" o conteúdo.

Caso você queira que todos os itens tenham mesmo tamanho, para deixar o layout mais "certinho", basta usar a largura fixa, ao invés de tamanho mínimo **min-width**.

```
.item {padding:15px; background:rgba(0,0,100,0.2); margin:10px; min-width:150px; border:1px solid white;
	border-radius:7px;}
```

Com os itens ajustáveis, podemos criar uma classe de conteúdo para os produtos da Atividade 5, com os itens em **divs** separadas.

Atividade 5: Crie um site de vendas com layout em tabela similar ao exemplo abaixo (index.htm). Crie logo e título com imagens ou pictogramas, menu lateral, imagens de produtos e textos com promoções. Crie uma página com mesmo layout, mesclando as células de produtos em uma só, criando uma descrição da empresa, com uma imagem da empresa.

Use CSS para modificar os atributos dos elementos das tags de links do menu lateral e do horizontal (tag de link **a**, por exemplo). Crie **id** ou **class** para os produtos em promoção (mudar cor ou fundo).

<table>
<tr><td>LOGO</td><td colspan="5">BANNER</td></tr>
<tr><td rowspan="4">MENU LATERAL</td><td colspan="5">produtos | fale conosco | promoções | sobre nós</td></tr>
<tr><td>Produto 1
Preço:
Promoção:</td><td>Produto 2
Preço:
Promoção:</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>Produto 12
Preço:
Promoção:</td></tr>
<tr><td colspan="6">COPYRIGHT, ENDEREÇO,...</td></tr>
</table>

8. TABLELESS

Para criar sites com os elementos semelhantes aos que usamos na Atividade 5, podemos usar tags ** ** para criar cada grupo de elementos. Uma propriedade que podemos usar é a **float** (left, right ou none), que permite que as tags possam se encaixar à esquerda ou à direita de uma tag anteriormente colocada na página. Outras propriedades que já utilizamos são as chamadas **flex**, que permitem tamanhos variáveis dos elementos no site, de acordo com o tamanho da tela que o visitante abrir o site que criarmos.

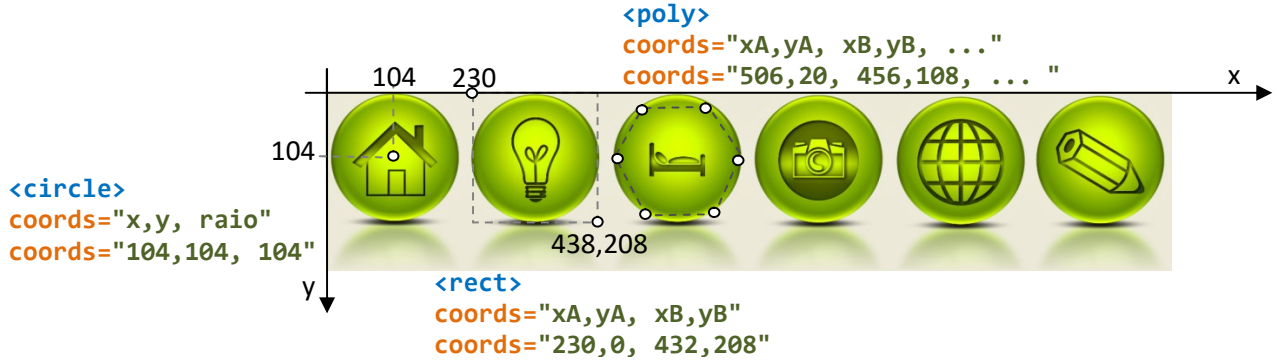
Vamos criar nossas próximas páginas sem usar tabelas, usando tags ** **: é o que chamamos de layout **tableless**.

8.1. MAPEAMENTO DE IMAGENS

O mapeamento de imagens é uma ferramenta de HTML que permite criar links ou informações sobre um determinado tema. Muitas vezes pode ser usada para criar os links de um menu, ou para criar sites com mapas temáticos. Ao criar o link de uma imagem, o visitante pode clicar em qualquer ponto da imagem para acessar o link. Com as tags de mapeamento, podemos dividir uma imagem em setores, com links diferentes.

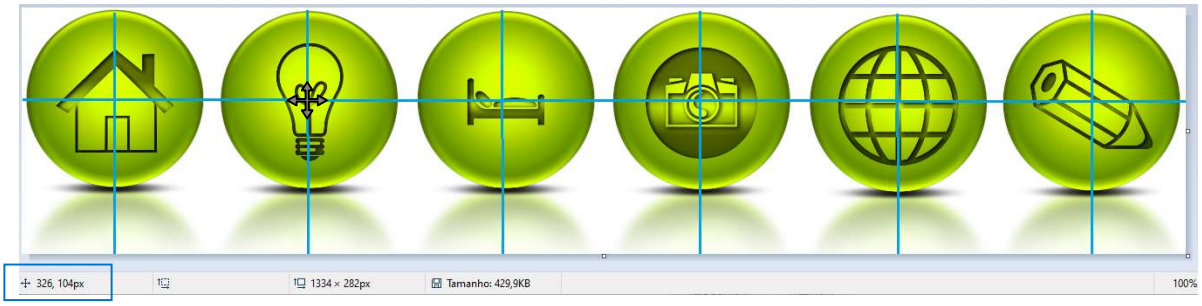
Considere o exemplo a seguir, onde uma imagem foi criada para um menu com os ícones **HOME**, **DICAS**, **HOSPEDAGEM**, **GALERIA DE IMAGENS**, **MAPAS** e **CONTATO**. Para criar o mapeamento da imagem do menu, precisamos calcular quais são os limites entre cada parte da imagem. Estes limites podem ser definidos usando círculos (**circle**), retângulos (**rect**) ou polígonos (**poly**).

Os elementos do círculo são: coordenadas **x** e **y** do centro e o **raio**. No retângulo, definimos as coordenadas do vértice à esquerda e no topo e as coordenadas do vértice à direita da base, como se estivéssemos percorrendo a diagonal do retângulo. Os polígonos são definidos pelas coordenadas dos vértices adjacentes.



O tamanho da imagem deste exemplo é de 1334px x 282px. Em um editor de imagens (Paint, Gimp, Corel) é possível verificar as coordenadas do centro e o tamanho do raio de um círculo que engloba o desenho do primeiro

item do menu. Da mesma forma, definimos as coordenadas do retângulo e do hexágono, para exemplificar o uso de cada figura.



Veja como fica a estrutura simplificada a seguir para montar o layout da página com mapeamento de imagens, com os exemplos de um círculo para o primeiro item, um retângulo para o segundo item e um hexágono para o terceiro item.

```
<!DOCTYPE html>
<html>
<head>
	<title>LAYOUT COM CSS</title>
	<meta name="author" content="AUTOR DA PÁGINA">
	<meta charset="UTF-8">
	<style type="text/css">
		body {background:rgba(100,200,220); width:80%; margin:auto; box-shadow:0px 0px 5px grey;}
	</style>
</head>
<body>
	<img src="imagens/menu_horizontal.png" width="1334" height="282" usemap="#Map">
	<map name="Map" id="Map">
		<area shape="circle" coords="104,104, 104" href="home.html" title="HOME" />
		<area shape="rect" coords="230,0, 432,208" href="dicas.html" title="DICAS" />
		<area shape="poly" coords="506,20, 456,108, 506,195, 605,195, 654,108, 605,20"
		   href="hospedagem.html" title="HOSPEDAGEM" />
	</map>
</body>
</html>
```

Exercício 9: Escolha uma das formas e complete o menu de mapeamento de imagens com os shapes HTML.

Ao redimensionar a tela, deixando-a com tamanho para um smartphone, a imagem não usa percentuais para mostrar os links. Por isso, essa construção de mapeamento pode ser feita quando não precisamos nos preocupar com o acesso em telas menores. Esta é uma desvantagem do mapeamento destas tags com os shapes, pois seu funcionamento só é permitido com tamanhos fixos em pixels dos shapes.

Uma alternativa para deixarmos uma página com mapeamento de imagens com formato responsivo é usar a tag **SVG**, que define gráficos vetoriais no formato de tags HTML. Mais adiante vamos usá-las para fazer os gráficos também. Neste ponto, vamos apenas usá-las para mapear imagens em nossas páginas.

Uma tag **<svg>** contém vários elementos gráficos, e pode ser inserida em qualquer parte de uma página HTML. Precisamos definir a janela de visualização **viewBox** para deixá-la ajustável com outros elementos da página. Basta definir as coordenadas do ponto mais à esquerda e ao topo da janela (geralmente é a origem do sistema de coordenadas) e o ponto mais à direita e na base da janela (representado pela largura e altura da imagem). No exemplo mostrado a seguir, a tela **SVG** tem altura de **150px** e largura de **430px**.

```
<svg viewBox="0,0, 430,150">
```

A tag **<circle>** desenha um círculo com centro de coordenadas **cx** e **cy** e raio **r**. Quando não informamos as coordenadas, o círculo é desenhado na origem. Podemos definir os seguintes atributos: cor da linha (**stroke**),

espessura da linha (**stroke-width**) e cor de preenchimento (**fill**). Estes atributos podem ser definidos dentro da tag ou com CSS.

```
<circle cx="40" cy="40" r="25" stroke="black" stroke-width="2" fill="red"/>
```

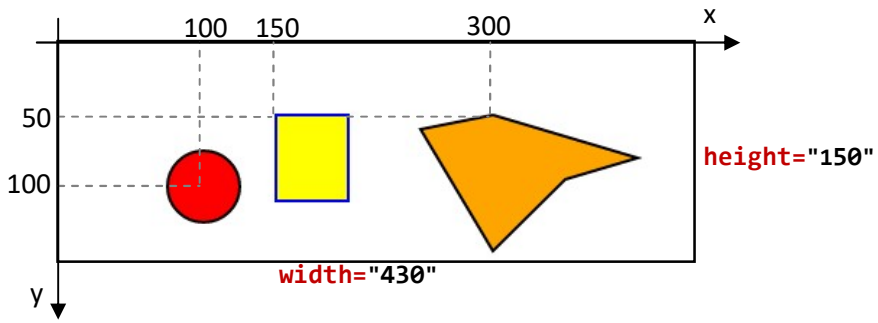
A tag **<rect>** desenha um retângulo com vértice do topo e esquerda de coordenadas **x** e **y**, com largura e altura que definimos.

```
<rect x="150" y="50" width="50" height="60" stroke="blue" stroke-width="2" fill="yellow"/>
```

A tag **<polyline>** desenha um polígono com vértices adjacentes com coordenadas **x** e **y**. Para fechar o polígono podemos repetir as coordenadas do primeiro vértice.

```
<polyline points="300,50, 400,80, 350,95, 300,145, 250,60  300,50" stroke="black" stroke-width="2"
fill="orange"/>
```

A janela criada da tag svg deste exemplo aparece da seguinte maneira:



Usando as medidas do menu horizontal do nosso exemplo anterior e a folha de estilos CSS para os atributos, o código HTML para os 3 shapes descritos de SVG fica da seguinte forma:

```
<svg viewBox="0,0, 1334,282">
	<image href="imagens/menu_horizontal.png" x="0" y="0" height="282" width="1334"/>
	<a xlink:href="home.html">
		<circle cx="104" cy="104" r="104"/>
	</a>
	<a xlink:href="dicas.html">
		<rect x="230" y="0" width="208" height="208">
	</a>
	<a xlink:href="dicas.html">
		<polyline points="506,20, 456,108, 506,195, 605,195, 654,108, 605,20">
	</a>
</svg>
```

Na folha de estilos CSS, podemos definir cores, linhas e efeitos para os links do nosso menu. O primeiro atributo é da largura da tag svg, que deve acompanhar a largura de sua tag "mãe".

```
<style type="text/css">
	svg {width:100%;}
</style>
```

Dentro da tag de estilos, vamos criar um efeito de que só apareçam os shapes quando o visitante passar o cursor do mouse por cima dos itens do menu. Logo, podemos colocar que o preenchimento dos shapes circle, rect e polyline será 0 (é o mesmo efeito que colocarmos que a cor de preenchimento destes elementos é o rgba(0,0,0,0)).

```
circle, rect, polyline {fill-opacity:0;}
```

Usando o atributo **hover** de cada shape, definimos cor de preenchimento **fill**, espessura de linha **stroke-width**, opacidade de preenchimento **fill-opacity** (que não será um valor próximo de 1, para não encobrir a imagem do menu) e a opacidade da linha **stroke-opacity**.

```
rect:hover, circle:hover, polyline:hover {fill:blue; stroke:green; stroke-width:5; fill-opacity:0.5;
	stroke-opacity:0.3;}
```

Exercício 10: Escolha uma das formas e complete o menu de mapeamento de imagens usando tags e shapes **svg**.

É comum que os mapeamentos de imagens mostrem efeitos e informações sobre os itens de um menu ou do conteúdo. Vamos aproveitar a estrutura usada com o SVG para mostrar os nomes dos itens do menu quando o visitante passar o cursor do mouse no menu.

O menu com cantos arredondados pode ser mapeado com os retângulos. Como todos os retângulos têm medidas iguais de largura e altura, podemos definir estas informações no CSS. Os cantos arredondados, com mesma função de border-radius que já usamos na Atividade anterior, têm o nome rx que está definido junto com altura e largura dos retângulos na tag abaixo.

```
rect {width:193px; height:190px; rx:40px;}
```

Podemos ligar os textos com seus respectivos retângulos colocando as tags HTML “aninhadas”, ou seja, dentro da tag de link, colocamos textos e retângulos da seguinte maneira:

```
<svg viewBox="0,0, 1287,220">
	<image href="imagens/menu_horizontal1.png" x="0" y="0" height="203px" width="1287px"/>
	<a xlink:href="home.html">
		<text x="8" y="215">HOME</text>
		<rect x="8" y="0"/>
	</a>
	<a xlink:href="dicas.html">
		<text x="223" y="215">DICAS</text>
		<rect x="223" y="0"/>
	</a>
</svg>
```

Textos e retângulos têm mesmas coordenadas x, para ficarem alinhados à esquerda. Para que textos e retângulos fiquem escondidos, podemos usar **fill-opacity** ou **opacity** no CSS para criar este efeito:

```
rect, text {opacity:0;}
```

O efeito para que apareça o texto respectivo de cada retângulo é obtido quando usamos o atributo **hover** condicional dos textos:

```
a:hover > text, a:hover > rect {opacity:1; transition:0.5s;}
```

Logo, forma-se um retângulo de cantos arredondados em torno do item do menu, com respectivo texto abaixo.

Exercício 11: Escolha cores e efeitos para fazer o mapeamento de um menu, com respectivas informações de texto dos ícones, usando tags e shapes **svg**.

Outra maneira de fazer mapeamento responsivo, ou seja, que funcione para tamanhos variáveis de telas (principalmente telas de smartphones e tablets), podemos criar classes e sub-classes em CSS com uso de percentuais do tamanho da imagem. Estes elementos serão definidos com uso das tags de blocos chamadas **div**. Esta maneira é mais prática quando temos mais informações para colocar, pois com o SVG fica um pouco mais trabalhoso fazer quebras de linhas.

O primeiro bloco que vamos criar será o Mapa principal, com a **div** chamada **Map** para englobar as divs filhas, que contém os itens do menu. Funciona como a criação de listas, que vimos nos primeiros sites.

Vamos criar as classes **i1**, **i2**,.... para cada item do menu, todos sem a propriedade de foco. Estas classes servem para criarmos retângulos em forma de **divs** para criar links adaptáveis ao tamanho da tela.

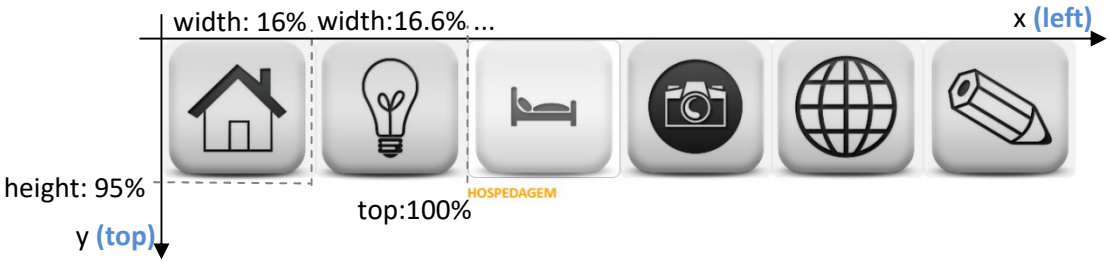
```
<div class="Map">
	<img src="imagens/menu_horizontal1.png">
	<div class="items">
		<div title="HOME"><a href="home.html" class="i1"></a><p>HOME</p></div>
		<div title="DICAS"><a href="dicas.html" class="i2"></a><p>DICAS</p></div>
		<div title="HOSPEDAGEM"><a href="hospedagem.html" class="i3"></a><p>HOSPEDAGEM</p></div>
	</div>
</div>
```

Na parte do CSS, vamos criar os códigos para esconder os elementos quando não estiverem em foco: basta usar a propriedade **visibility**. Quando o visitante passar o cursor do mouse sobre o elemento do menu (**hover**), **visibility** terá o atributo **visible**, e na visualização normal será **hidden**. O mesmo ocorre no texto, onde colocamos o atributo **visibility** para os itens de parágrafos dentre das tags de itens.

O mesmo ocorre em cada item com link **a**, dentro das tags **items**. Neste caso, colocamos um fundo com opacidade 0.3 para destacar que os itens estão em foco.

```
.Map {width:100%; position:relative;}
.Map img {width:100%;}
.items {width:100%; height:100%; position:absolute; left:0; top:0; visibility:hidden;}
.items div:hover {visibility:visible;}
.items div:hover > p {visibility:visible;}
.items div:hover > a {background:#fff; z-index:2; opacity:0.3; border:1px solid #000;}
```

Para mapear a imagem, podemos usar um editor de imagens com a régua para determinar os tamanhos dos retângulos (**width** e **height**), e suas posições relativas (**left** e **top**). Usando a unidade do editor de imagens, calculamos todas as medidas com percentuais, pois o site deverá ser responsivo.



A propriedade **z-index** pode ser usada para colocar a div em evidência quando estiver em foco. Valores maiores de z-index deixam o conteúdo mais evidente. O valor de **z-index** negativo deixa a imagem atrás de outras.

Cada item do menu tem sua classe **i1**, **i2**,.... e pode ser mapeado na tela com as coordenadas relativas (left, top, width e height).

Podemos usar a ferramenta de calculadora para facilitar as posições dos elementos da página. A largura de cada item será de **100%/6**, logo cada **width** pode ser calculado usando **calc(100%/6)**. As alturas dos mapeamentos podem ser de 95% nesta imagem de menu horizontal. Todos os retângulos de mapeamento ficarão no topo da imagem, ou seja, top: 0%;

```
.Map a {width:calc(100%/6); height:95%; top:0%;}
```

O espaçamento entre o item **i1** e **i2** será de **100%/6**; entre **i2** e **i3** será de **2*100%/6**, e assim sucessivamente. Esta medida também será usada para mostrar os textos dos itens do menu (**a + p,** ou seja, link + parágrafo). Logo, a programação do CSS fica valendo para itens e textos:

```
a.i1, a.i1 + p {left:0%;}
a.i2, a.i2 + p {left:calc(100%/6);}
a.i3, a.i3 + p {left:calc(2*100%/6);}
a.i4, a.i4 + p {left:calc(3*100%/6);}
a.i5, a.i5 + p {left:calc(4*100%/6);}
a.i6, a.i6 + p {left:calc(5*100%/6);}
```

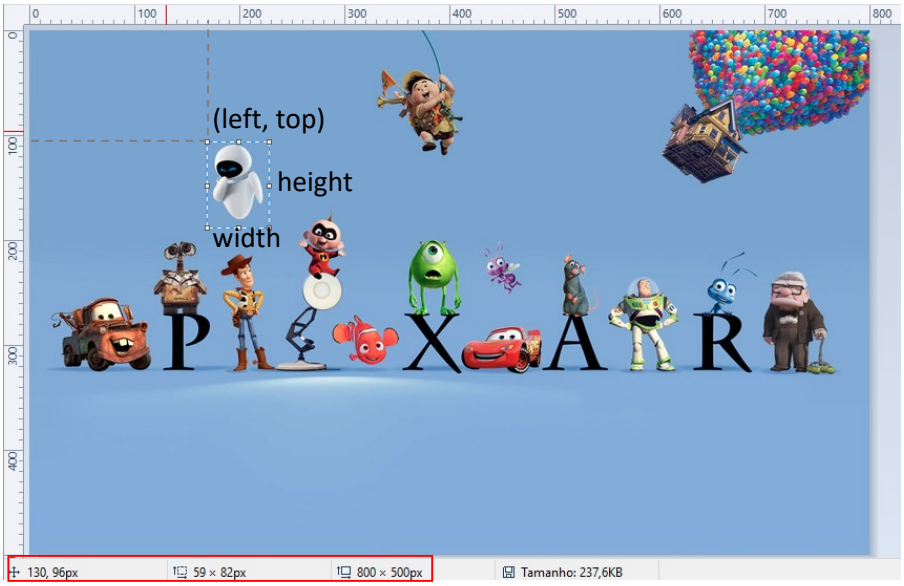
Os itens de texto podem ser mapeados usando a propriedade **a + p** (link + parágrafo), que deve ter visibility escondido (**hidden**) e só fica ativo quando o visitante focalizar o item (**hover**).

```
.Map a + p {position:absolute; top:75%; width:calc(100%/6); color:orange; visibility:hidden;}
```

Para mapear uma imagem, podemos anotar as medidas das coordenadas de um retângulo de seleção (**top** e **left**), as medidas de largura e altura (**width** e **height**) e usar a ferramenta **calc(medida/total*100%)**.

Exercício 12: Crie uma página com um menu lateral feito em um editor de imagens. Crie os links e retângulos de seleção com mapeamento de imagens. Use a ferramenta **calc** para ajustar os links.

Atividade 6: Crie uma página com mapeamento de uma imagem. Em um editor de imagens, selecione 3 itens (personagens, por exemplo) para inserir descrição de cada item. Use o mesmo tipo de estrutura do menu que fizemos para criar as caixas de descrições.

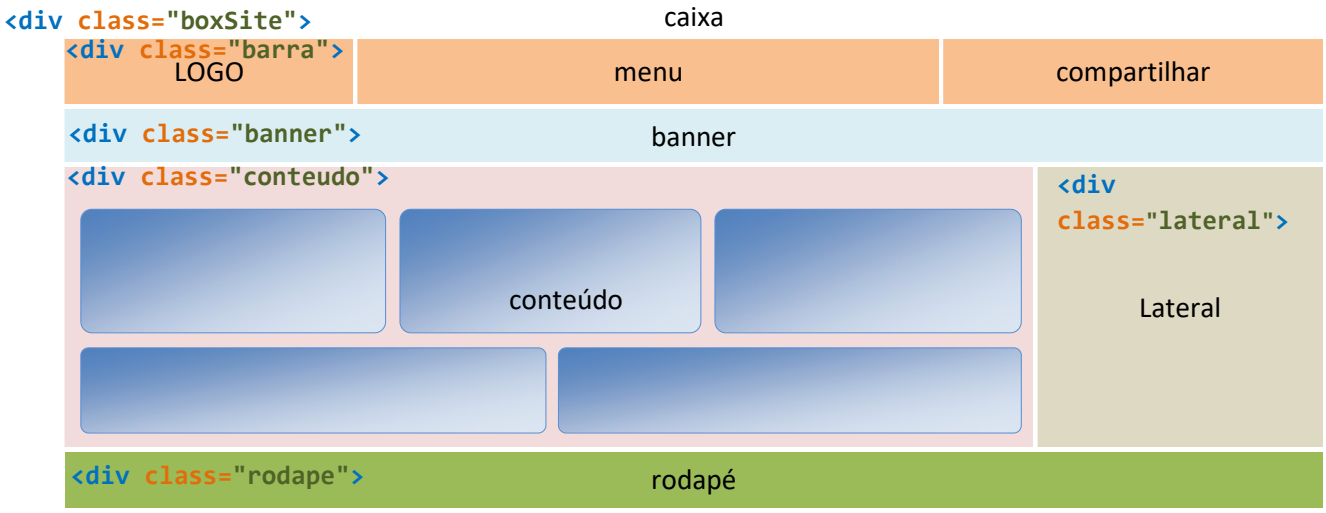


Vamos usar este exemplo para ajudar a compreensão do posicionamento e dimensionamento de cada item. O retângulo tem coordenada dp canto esquerdo **left = 130** e **top = 96**, mostrados na barra do editor de imagens. Nesta barra também aparecem a largura **width = 59** e a altura **height = 82** do retângulo. Logo, podemos usar uma regra de três relativa às dimensões da imagem: largura de **800** e altura de **500**.

Assim, a posição do retângulo será calculada por meio da expressão **`left:calc(130*100%/800).`**
A posição da altura será obtida com o cálculo **`top:calc(96*100%/500).`**
A largura do retângulo destacado na imagem será **`width:calc(59*100%/800).`**
A altura deste retângulo será **`height:calc(82*100%/500).`**

8.2. LAYOUT FLEX

Vamos criar um site com o seguinte layout:



Como teremos muitos elementos para formatar com CSS, podemos colocar todas as suas propriedades em um arquivo chamado **estilo.css**. No cabeçalho da nossa página index.htm vamos criar um link para o CSS:

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

Vamos utilizar as propriedades **flex** para ajustar os elementos da página. A **div** mãe dos elementos deve ter a propriedade de **display flex** e o ajuste da proporção das divs filhas é feita através das propriedades do **flexbox**.

Na tag da classe **barra**, teremos as divs filhas **logo**, **menu** e **compartilhar**, com a estrutura a seguir:

```
<!DOCTYPE html>
<html>
<head>
        <title>LAYOUT COM CSS</title>
        <meta name="author" content="AUTOR DA PÁGINA">
        <meta charset="UTF-8">
        <link href="estilo.css" rel="stylesheet" type="text/css">
</head>
<body>
  <div class="boxSite">
    <div class="barra">
       <div class="logo">
              Logo
        </div>
        <nav class="menu">
              <ul>
                <li><a href="index.htm">Home</a></li>
                <li><a href="pagina1.htm">Página 1</a>
                   <ul class="subitens">
                      <li><a href="#">Sub página 1</a></li>
                      <li><a href="#">Sub página 2</a></li>
                      <li><a href="#">Sub página 3</a></li>
                      <li><a href="#">Sub página 4</a></li>
                   </ul>
                </li>
                <li><a href="pagina2.htm">Página 2</a></li>
                <li><a href="pagina3.htm">Página 3</a></li>
                <li><a href="pagina4.htm">Página 4</a></li>
            </ul>
        </nav>
        <div class="compartilhar">
            <img src="imagens/facebook.png">
            <img src="imagens/email.png">
            <img src="imagens/twitter.png">
```

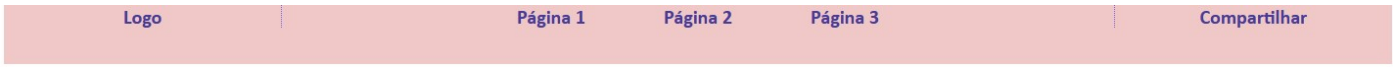
```
        </div>
    </div>
  </div>
</body>
</html>
```

No arquivo **estilo.css** podemos colocar configurações parecidas com as que usamos no menu das Atividades 4 e 5. Vamos começar com os atributos **flex**.

A tag "mãe" chamada **barra** engloba as tags "filhas" **logo**, **menu** e **compartilhar**. Ela possui a propriedade de **display flex** para facilitar a distribuição de proporções. Podemos definir uma altura mínima para esta barra e o alinhamento dos itens.

```
.barra {background:#81BEF7; min-height:120px; display:flex; align-items:center;}
```

O atributo align-items tem mesma função do vertical-align que já usamos. Veja como ficam os valores deste atributo para as tags filhas.



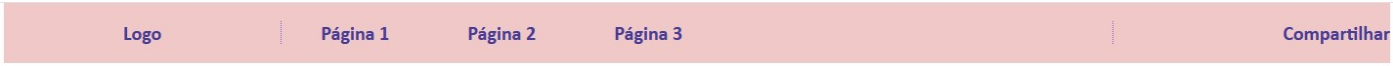

```
.barra {... align-items:flex-start;}
```


```
.barra {... align-items:flex-end;}
```


```
.barra {... align-items:center;}
```

O atributo de alinhamento horizontal já foi usado anteriormente: **justify-content**. Cada div deve ter este alinhamento definido separadamente no CSS. Veja os 3 exemplos deste alinhamento na nossa barra:

```
.menu ul {justify-content:flex-start;}
```




```
.logo {justify-content:center;}
.compartilhar {justify-content:flex-end;}
```

A propriedade **flex** distribui as proporções dos elementos de uma **div** mãe. Por exemplo, se colocarmos **flex:1** para todos os elementos filhos de uma **div**, as larguras serão distribuídas igualmente. Desta forma, todas as **div** "tentarão" ficar com mesma largura, ocupando a largura de 100% da **div mãe**. Veja alguns exemplos da aplicação deste atributo na nossa barra superior da página. Primeiro, todas com mesmo valor para **flex:1**.

```
<div class="barra">
  <div class="logo">          <nav class="menu">          <div class="compartilhar">
```


```
.logo {flex:1;}          .menu {flex:1;}          .compartilhar {flex:1;}
```

Neste caso, todas ficam com aproximadamente 33% de largura. Caso alguma **div** tenha mais conteúdo, a proporção sofre pequenos ajustes para mostrar os conteúdos. Agora, colocando valores **flex:2** para duas divs e **flex:1** para outra div.

```
<div class="barra">
  <div class="logo">          <nav class="menu">          <div class="compartilhar">
```


```
.logo {flex:1;}          .menu {flex:2;}          .compartilhar {flex:2;}
```

No nosso exemplo, foi colocado **flex:3** para a div **compartilhar**, e **flex: 1** para **menu** e **logo**. Portanto, a div compartilhar terá a proporção de **3/(3+1+1)=3/5=60%**, e as outras divs ficarão com **1/(3+1+1)=1/5=20%**. Estas proporções sofrem ajustes nos casos de conteúdos com tamanhos discrepantes (por exemplo, imagens muito pequenas misturadas com imagens muito grandes; nestes casos, temos que ajustar os tamanhos via CSS para não criar layouts estranhos).

Agora vamos adicionar mais atributos para os itens do **menu**. A propriedade **transition** faz o efeito hover suavizado, com possibilidade de configurá-la em tempo e tipo de transição (**ease**: lenta no começo e fim; **linear**: mesma velocidade; **ease-in**: lenta somente no começo; **ease-out**: lenta somente no fim).

```
.menu {flex:1;}
.menu ul {list-style-type:none; margin:0; padding:10px; display:flex; justify-content:center; flex-
    direction:row; flex-wrap:wrap;}
.menu li {margin:0; padding:0; border-right:1px grey solid; min-width:100px;}
.menu li a {color:#666; text-decoration:none; transition:1s linear;}
.menu li a:hover {color:blue; font-size:200px; text-shadow:1px 1px 6px black; transition:1s linear;}
.menu li:last-child {border-right:none;}
```

Outos atributos de alinhamentos e tamanhos de imagens podem ser definidos nas **div logo** e **compartilhar**. As imagens destas **div** podem ser alinhadas com a propriedade **justify-content** (**flex-start**, **flex-end**, **center**, **space-around** ou **space-between**). O alinhamento vertical dos conteúdos destas **div** também está definido como centralizado.

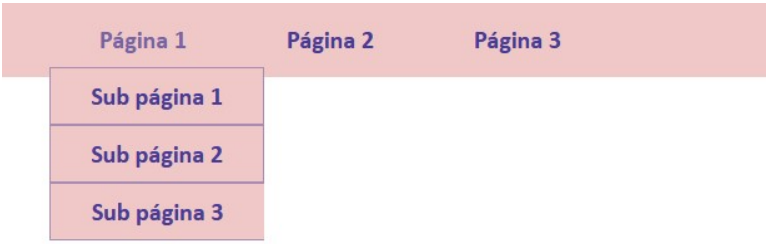
```
.logo {flex:1; display:flex; align-items:center; justify-content:center;}
.compartilhar {flex:3; display:flex; align-items:center; justify-content:center;}
.compartilhar img {width:10%; margin:15px; min-width:45px;}
```

Para criar os itens do menu como botões podemos usar as bordas arredondadas e um background para cada item **li** da lista **menu**. Ao usar apenas uma medida, as quatro bordas ficam com mesmo raio. Ao usar porcentagem, as bordas podem deixar as caixas com formato de elipse, que é o caso de usar **border-radius: 50%**.

```
border-radius:medidaTopoEsquerda medidaTopoDireita medidaBaseDireita medidaBaseEsquerda;
```

Os subitens do menu formam o chamado **menu dropdown**: quando o visitante passa o cursor do mouse sobre um item que tem subitens, e estes subitens aparecem.

O estado normal dos subitens (**li ul**) será com opacidade 0, (**opacity:0**), os quais podem ter um tamanho mínimo definido (para evitar que cada item apareça com um tamanho diferente), com **z-index:1** para que eles apareçam em evidência quando a função **hover** estiver ativa. Podemos usar também a propriedade de transição para que os subitens apareçam de forma mais suave na tela.




```
.subitens {position:absolute; z-index:1; border:1px grey solid; background:#81BEF7; opacity:0;
    transition:.3s linear;}
```

Os subitens podem ter uma borda inferior (**bottom**) para separar os textos. A propriedade **flex-direction** pode ser usada para deixar os itens dispostos horizontalmente (**row**) ou verticalmente (**column**).

```
.menu .subitens {margin-top:10px; flex-direction:row;}
.subitens li {border-bottom:1px solid grey;}
.menu li:hover ul {opacity:1;}
```

Para deixar nossas páginas **responsivas**, isto é, automaticamente adaptadas a telas menores, como tablets e smartphones, devemos fazer alguns testes para verificar quais serão as **divs** que sofrerão ajustes de largura para não afetar a visualização das suas páginas. Vamos começar modificando a **div barra**.

A técnica para mudanças de layout para telas menores é muito simples: basta diminuir o tamanho da tela do navegador e verificar quais elementos necessitam mudar de largura. Para ajustar a um **tablet** (modo paisagem com 1024px e retrato 768px), podemos deixar os itens da barra um abaixo do outro, ou seja, a propriedade **flex-direction** será **column**. O alinhamento dos itens deve ser modificado para **stretch** (esticado).

Como o modo paisagem do site não muda muito em relação ao modo retrato de um tablet, vamos nos preocupar somente com dispositivos com as larguras de telas inferiores a 768px (valor que pode ser arredondado para 800px). O atributo CSS para modificações de telas é chamado **@media screen**.

```
@media screen and (max-width:800px) {
    .barra {flex-direction:column; align-items:stretch;}
}
```

Podemos aproveitar para deixar o **menu** com a configuração **toggle** (alternância)**,** símbolo de pictograma usado em telas menores para indicar onde tem uma estrutura de menu do site.

Além de usar este toggle, vamos usar um truque para não programar o menu com javascript. Um checkbox será inserido, pois tem a função booleana (tipo “sim” ou “não”). Um label (rótulo) será usado como uma div mãe do menu, inserindo o **toggle** no menu. A estrutura HTML fica assim:

```
<input type="checkbox" id="checkbox1">
<label for="checkbox1">
        <nav class="menu">
                ....
        </nav>
<span class="toggle">&#9776;</span>
</label>
```

Já no CSS, precisamos esconder o **toggle** e o **checkbox** quando a largura da tela for maior do que 800px, ou seja:

```
#checkbox1, .toggle {display:none;}
```

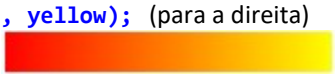
Mas quando a largura da tela for menor do que 800px, podemos inserir as propriedades de visibilidade do menu. Os subitens ficarão visíveis somente quando ocorrer o **hover** dos itens do menu. Estas propriedades são bem simples, e alteram altura **height** do menu de 0 para 100%, de acordo com o que acontece no **checkbox1**. Outra propriedade importante usada é a **flex-direction**, que deve ser **column** (vertical).

```
@media screen and (max-width:800px) {
    .barra {flex-direction:column; align-items:stretch;}
    .toggle {display:block; width:100%; text-align:center; font-size:30px; cursor:pointer;
       color:#595959; background:#dbdbdb;}
    #checkbox1:checked + label .menu li {visibility:visible;}
    #checkbox1:checked + label .menu {height:100%;}
    .menu ul {display:flex; flex-direction:column;}
    .menu {width:100%; height:0;}
    .menu li {visibility:hidden; border-right:none; padding:10px;}
    .menu li:hover .subitens {position:relative; display:flex;}
}
```

Antes de inserir os demais elementos da página, vamos modificar os backgrounds das barras (menu, logo e compartilhar) com preenchimentos gradientes.

Para criar o efeito de gradiente linear, podemos usar ângulos ou escrever a direção do efeito:

```
background:linear-gradient(to right, red, yellow);
```
(para a direita)



```
background:linear-gradient(to bottom right, red, yellow);
```
(diagonal)

```
background:linear-gradient(-40deg, red, yellow);
```
(com 40°)

```
background:linear-gradient(red, yellow, green);
```
(sem indicar o sentido, fica de cima para baixo)

```
background:linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet);
```

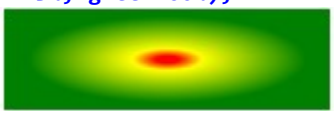
Podemos usar o padrão rgb ou rgba (com a última coordenada indicando opacidade).

```
background:linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
background:repeating-linear-gradient(red, yellow 50%, green 70%); (com repetição)
```

```
background:linear-gradient(45deg, rgb(0,0,0) 0%, rgb(0,39,21) 40%, rgb(21,222,333) 100%);
```

Os atributos para criar efeitos de gradiente radial são parecidos com os lineares:

```
background:radial-gradient(red, yellow, green); (cria elipses de acordo com o tamanho da div)
background:radial-gradient(red 5%, yellow 15%, green 60%);
```



```
background:radial-gradient(circle, red, yellow, green); (com circunferência)
background:repeating-radial-gradient(red, yellow 10%, green 15%); (com repetição)
```

Para definir o centro da circunferência, usamos o atributo **farthest-side**:

```
background:radial-gradient(farthest-side at 75% 75%, red, yellow, black);
```

Podemos criar um efeito gradiente nos itens do menu, e ao passar o cursor do mouse sobre os botões o efeito gradiente pode ser invertido. Coloque estas propriedades no CSS para criar a margem interna em botões:

```
.menu li a {background:linear-gradient(-40deg, red, white); border-radius:7px;...}
.menu li a:hover {background:linear-gradient(-40deg, white, red); ...}
```

Para criar animações com os elementos das divs do site, podemos usar a função **@keyframes**. Por exemplo, vamos colocar a propriedade de animação **AnimaLogo** na classe **logo** para mudar o tamanho e a cor da fonte.

```
.logo {animation:AnimaLogo 5s linear infinite alternate; ...}

@keyframes AnimaLogo {
  0% {font-size:0; color:white;}
  100% {font-size:45px; color:blue;}
}
```

Configurações do início da animação: 0%

Configurações do fim da animação: 100%

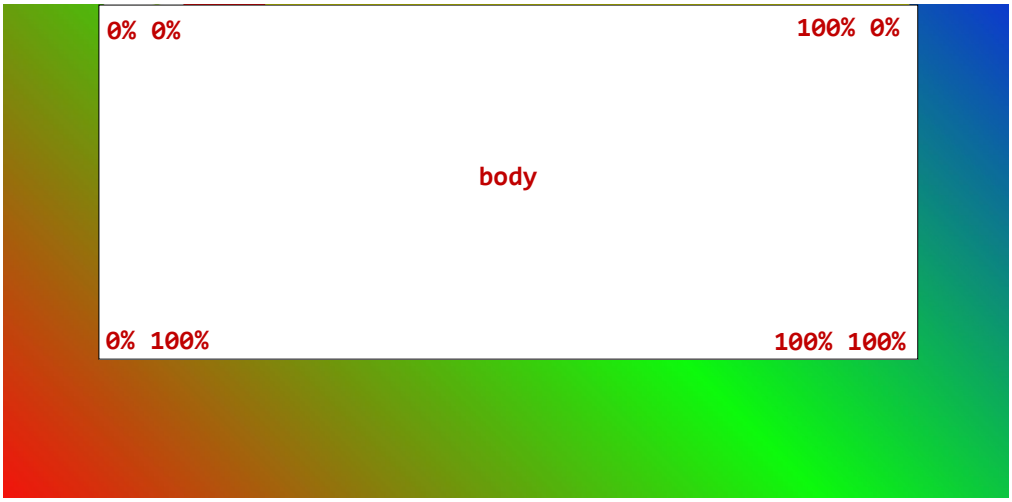
Este exemplo mostra a animação em 2 estágios: inicial e final. Os estágios intermediários podem ser inseridos quando você quiser alguma transição especial. Por exemplo: da cor branca, para a amarela, depois vermelha e no final a azul. Neste caso, precisaríamos de 2 estágios intermediários para as cordes amarela a vermelha. Vamos ver adiante mais exemplos com animações.

Seja qual for o elemento, as funções de animação serão similares, todas inseridas no arquivo CSS. Vamos criar um efeito de animação no fundo de um elemento do site. Por exemplo, vamos fazer no fundo do site, ou seja, com a tag **body** em CSS. O efeito será de mudança da posição do fundo que tem efeito gradiente.

Para funcionar, teremos que aumentar o tamanho do background do body, para mudarmos o fundo usando a propriedade **background-position** para animação (AnimaFundo, com 10 segundos de duração, transição **ease**, alternado e repetição infinita). O CSS do body fica da seguinte forma:

```
html, body {height:100%;}
body {width:90%; margin:auto; background:linear-gradient(45deg, rgba(250,0,0,0.95), rgba(0,250,0,0.95),
   rgba(0,0,250,0.95)); background-size:200% 200%; animation:AnimaFundo 10s ease infinite alternate;}
```

Vamos usar a animação para fazer o fundo do site “mexer” variando entre os cantos do site (body).



Usamos o atributo **keyframes** para inserir os efeitos da animação.

```
@keyframes AnimaFundo {
   0% {background-position:0% 0%; color:white;}
   25% {background-position:100% 0%;}
   50% {background-position:0% 50%;}
   75% {background-position:100% 0%;}
   100% {background-position:0% 0%; color:blue;}
}
```

0% o fundo tem cor de fonte **branca** e posição **0% 0%.**
25% o fundo tem transição de cor entre branca (75%) e azul (25%) e “anda” **100%** na horizontal.
50% o fundo tem transição de cor entre branca (50%) e azul (50%) e “anda” **50%** na vertical.
75% o fundo tem transição de cor entre branca (25%) e azul (75%) e “anda” **100%** na horizontal
100% o fundo tem cor de fonte **azul** e volta na posição **0% 0%.**

Para animar uma imagem de fundo de uma **div**, podemos usar a propriedade **background-position** e fazê-la mudar de 0% até 100% da posição horizontal x ou vertical y. Dada a imagem de nuvens abaixo (clouds.png) que tem 2247x190px, podemos colocar os efeitos de animação da seguinte forma:

```
.barra2 {animation:AnimaBarra 40s linear infinite; background:url(imagens/clouds.png);}

 @keyframes AnimaBarra {
   0% {background-position:0% 0%;}
   50% {background-position:50% 0%;}
   100% {background-position:100% 0%;}
}
```

Para fazer a imagem ficar com efeito de vai e vem, podemos colocar a propriedade **alternate** no CSS da classe menu. Para fazer a animação andar somente no sentido contrário, basta colocar a propriedade **reverse** ao invés de **alternate**.

Para suavizar a imagem desta barra2, podemos usar a propriedade opacity:

```
.barra2 {animation:AnimaBarra 40s linear infinite; background:url(imagens/clouds.png); opacity:0.6;}
```

Agora vamos inserir uma imagem de fundo no banner e usar a propriedade **display flex** para criar uma logo sobre o banner. O banner pode ter uma altura mínima definida.

Para que a imagem ocupe todo o espaço do banner, vamos usar a propriedade **background-size** de 100% (cobre a largura **width**) ou **cover** (cobre a largura **width** e a altura **height**), sem repetição.

```
.banner {display:flex; min-height:400px; background:url(imagens/fundo.png) no-repeat;
   background-size:100%; align-items:center; justify-content:center;}
```

Para ajustar a posição da imagem do fundo do banner, podemos usar a propriedade **background-position: left, top**. Por exemplo, para deixar a imagem deslocada 100px e 200px em relação à esquerda (x) e ao topo (y), respectivamente, utilizamos **background-position: 100px, 200px**. Estas medidas podem ser usadas com percentuais.

Para inserir uma div de descrição do site, podemos colocá-la por cima do banner com a propriedade **position:relative**, indicando quais são as distâncias do topo (y) e à esquerda (x).

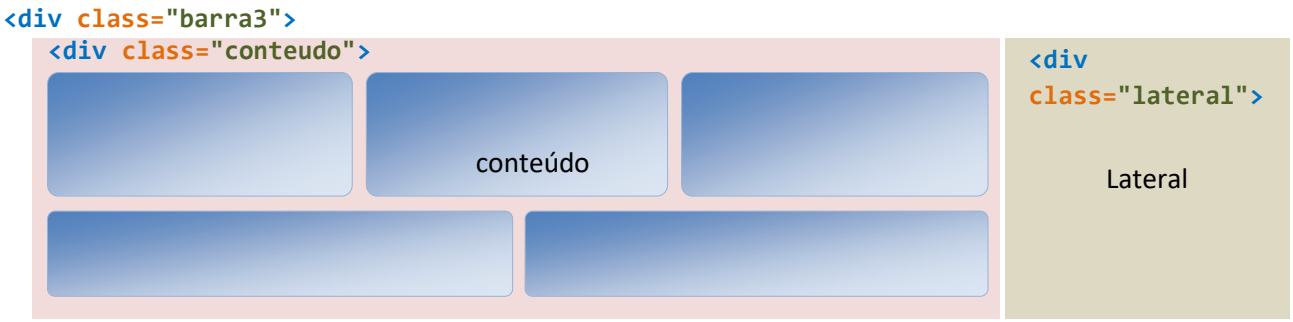
O tamanho desta div pode ser definido usando o atributo de largura **width,** ou deixando que seja ajustada automaticamente pelo seu conteúdo. Para colocar um fundo preto com opacidade, basta usar uma cor RGBA. Para mudar a cor de fundo desta div ao passar o cursor do mouse por cima, basta colocar a propriedade hover no CSS.

```
.descricao {padding:10px; position:relative; background:rgba(0,0,0,0.4);}
.descricao:hover {background:rgba(0,0,0,0.7);}
```

A div mãe do site é a **boxSite**. Ela serve para agrupar todo o site, e pode ter uma borda para delimitar o fundo e o conteúdo.

```
.boxSite {box-shadow:4px 4px 15px black;}
```

Agora vamos começar a estruturar o conteúdo da página. Para que o layout continue responsivo, vamos criar uma classe mãe para as classes **conteúdo** e **lateral: barra3**.



Podemos criar novas **divs** para inserir conteúdos em caixas dimensionáveis automaticamente. Ao usar a propriedade **flex:1 300px**, todas as caixas estarão distribuídas com espaçamentos iguais em largura e altura, com largura mínima de 300px.

Para que as caixas não fiquem grudadas, colocamos margens internas (padding) e externas (margin). A propriedade **flex-direction:column** é necessária para encaixar os elementos de cada caixa um abaixo do outro.

```
.barra3 {display:flex;}
.conteudo {display:flex; flex:3; background:khaki; flex-wrap:wrap;}
.box {border-radius:10px; flex:1 300px; flex-direction:column; color:#171e42; padding:10px; margin:10px;
   background:linear-gradient(rgba(0,0,255,0.5), rgba(0,0,255,0.1)); }
.lateral {flex:1; background:gold; align-items:center; justify-content:center;}
```

Vamos usar a propriedade **box-shadow** para criar uma sombra em cada caixa. O padrão para **box-shadow** é de uma sombra para fora do elemento, mas se você quiser uma sombra interna basta colocar a palavra **inset** na propriedade **box-shadow**. O exemplo a seguir é de uma sombra vermelha com 3px à direita e 3px abaixo da caixa, com raio de sombra 7px.

```
.box {box-shadow:3px 3px 7px red; ...}
```

O exemplo a seguir é de uma sombra externa verde e outra interna azul.

```
.box {box-shadow:4px 4px 10px grey, 6px 6px 10px blue inset; ...}
```

Para criar títulos que apareçam acima dos conteúdos das caixas podemos usar margens para separá-los dos demais conteúdos.

```
.titulo {margin-bottom:10px; background:rgba(0,153,255,0.5); color:white; padding:5px;}
```

Insira imagens e videos nas caixas do nosso site. Podemos inserir uma tabela na lateral da página, com cores diferentes nas linhas pares e ímpares. Coloque um título para a barra lateral e logo a seguir insira uma tabela. A propriedade **nth-child** é usada para diferenciar os pares (**even**) dos ímpares (**odd**).

```
tr:nth-child(even) {background:linear-gradient(0deg, rgba(255,0,0,0.4),rgba(255,0,0,0));}
tr:nth-child(odd) {background:linear-gradient(180deg, rgba(255,0,0,0.4),rgba(255,0,0,0));}
```

Podemos colocar um efeito **hover** para mudar as cores das linhas desta tabela.

```
tr:nth-child(even):hover {background:linear-gradient(180deg, rgba(0,0,255,0.4),rgba(0,0,255,0));}
tr:nth-child(odd):hover {background:linear-gradient(0deg, rgba(0,0,255,0.4),rgba(0,0,255,0));}
```

Você pode usar esta propriedade nas caixas de imagens e vídeos, da mesma forma:

```
.box:nth-child(even) {background:linear-gradient(0deg, blue, green);}
.box:nth-child(odd) {background:linear-gradient(180deg, blue, green);}
```

Agora vamos formatar o rodapé. Pode ser usada uma cor ou imagem de fundo e as informações finais da página. Usaremos **display:flex** e alinhamentos para centralizar o texto na **div** de rodapé.

```
.rodape {background:bisque; display:flex; justify-content:flex-start; padding:20px; color:black;}
```

Vamos criar mais efeitos de animação no site. Os elementos de páginas que podemos animar são **width**, **height, padding, margin**, **text-shadow, background**, **font-size** e **color.** Para colocar títulos do site e mudar a cor da fonte e a sombra do texto com um novo efeito de animação, basta criar o seguinte código em CSS:

```
@keyframes AnimaTitulo {
   0% {text-shadow:2px 2px 2px #CCC; color:rgba(255,255,255,0);}
   25% {text-shadow:2px -2px 2px #CCC; color:rgba(255,255,255,0.25);}
   50% {text-shadow:-2px -2px 2px #CCC; color:rgba(255,255,255,0.5);}
   75% {text-shadow:-2px 2px 2px #CCC; color:rgba(255,255,255,0.75);}
   100% {text-shadow:2px 2px 2px #CCC; color:rgba(255,255,255,1);}
}
```

Os efeitos de animação do CSS dos títulos ficam assim:

```
.titulo {animation:AnimaTitulo 7s ease infinite alternate; ...}
```

Outro efeito que fica interessante para o título principal de um banner de site é de mudar o tamanho da fonte (font-size) junto com a mudança da cor (color) usando opacidade.

Quando formatamos o tipo de fonte do nosso site, corremos o risco de que o computador do visitante do site não tenha a fonte instalada. Para contornar este problema, podemos utilizar fontes online para formatar nosso site. O site mais usado para inserir fontes diferentes das usuais é o **Google Fonts**.

https://fonts.google.com

Ao escolher as fontes, colocamos a tag **link** criada com as fontes escolhidas dentro da tag **<head>** do site:

```
<link href='https://fonts.googleapis.com/css?family=Open+Sans|Raleway|PT+Sans' rel='stylesheet'
   type='text/css'>
```

Para indicar no CSS a fonte de todos os itens do site, basta usá-la da seguinte forma:

```
* {font-family:'Open Sans';}
```

Agora utilizaremos as principais transformações geométricas do CSS para criar um efeito simples de galeria de imagens. As transformações mostradas a seguir podem ser usadas em diversos elementos do HTML (por exemplo, **div**, **p** e **img**).

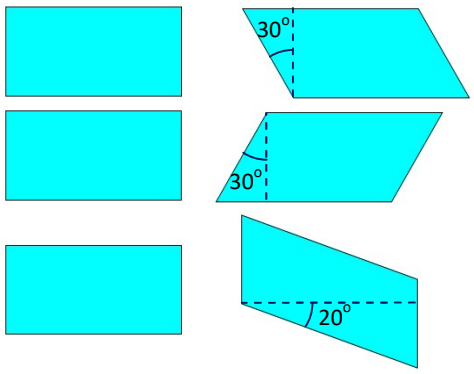
As principais transformações geométricas são: rotação (**rotate**), translação (**translate**), escala (**scale**) e cisalhamento (**skew**).

Vamos testar estas transformações nas imagens inseridas na **div box**. Vamos começar pelo cisalhamento de uma imagem, transformação que será ativada quando o visitante passar o cursor do mouse por cima de cada imagem. A primeira coordenada é o ângulo de deformação em relação ao eixo y (mantendo-se os lados paralelos ao eixo x sem deformação) e a segunda coordenada é o ângulo de deformação em relação ao eixo x (mantendo-se os lados paralelos ao eixo y sem deformação):

```
transform:skew(30deg,0deg);
```

```
transform:skew(-30deg,0deg);
```

```
transform:skew(0deg,20deg);
```



Para que a transformação tenha um efeito suave, podemos colocar a propriedade **transition** no CSS:

```
.box img:hover {transform:skew(45deg,0deg); transition:transform 0.5s linear; ...}
```

Para que o efeito na volta da imagem para a galeria não seja brusco, podemos usar a propriedade transition para as imagens originais do box1:

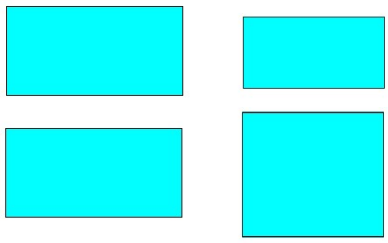
```
.box img {transition:transform 0.5s linear; ...}
```

Mude o ângulo de cisalhamento em x para 180°. Faça um teste com ângulo 0° para x e 180° para y.

Outra transformação geométrica que vamos usar é a mudança de escala (homotetia). Assim como o **skew**, o **scale** pode ser usado para mudar a escala em x, y ou em ambos os eixos.

```
transform:scale(0.8);
```

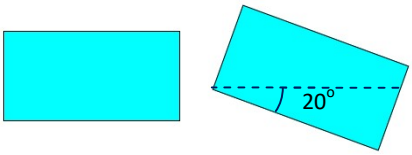
```
transform:scale(0.8,1.4);
```



Vamos usar o scale junto com o skew para criar um efeito de galeria de imagem com a CSS a seguir:

```
.box img:hover {transform:scale(3.5) skew(180deg,0deg);}
```

Outra transformação importante é a rotação, feita com o centro de rotação no centro do elemento e o ângulo relativo ao eixo x. O elemento rotacionado mantém todas as suas medidas, sem deformações e apenas muda de posição em relação ao eixo x.




```
transform:rotate(20deg);
```

A translação é feita a partir do ponto que a figura se encontra até o ponto com deslocamentos dx e dy nos eixos x e y. A notação é **translate(dx,dy)**.

Podemos usar a translação para arrumar o efeito da nossa galeria com relação às imagens que ficam mais próximas dos cantos da página. Neste caso, podemos colocar uma **translação** com percentual, o **skew** e **scale**.

```
.box img:hover {transform:scale(3.5) skew(180deg,0deg) translate(25%,0);}
```

Quando usarmos **scale** com valores (-1,1) teremos o efeito de espelhamento da imagem da figura em torno do eixo y; com valores (1,-1) teremos o espelhamento da imagem em torno do eixo x. A imagem que colocamos na descrição do site pode ser usada para testarmos estes efeitos de espelhamento junto com rotações com uma nova animação no site.

```
.descricao img {animation:AnimaDescr 7s linear infinite alternate;}
```

Vamos rotacionar a imagem no começo da animação, além de usarmos opacidade gradativamente e mudança de escalas. No final da animação, usaremos o efeito de espelhamento.

```
@keyframes AnimaDescr {
   0% {opacity:0; transform:rotate(-30deg);}
   25% {opacity:0.25; transform:scale(0.92);}
   50% {opacity:0.5; transform:scale(0.95);}
   75% {opacity:0.75; transform:scale(5);}
   100% {opacity:1; transform:scale(-1,1);}
}
```

Para finalizar animações do site, crie efeito no logo do site, com rotação e escala. Use efeito de escala em elementos iframe, assim como fizemos com a galeria de imagens.

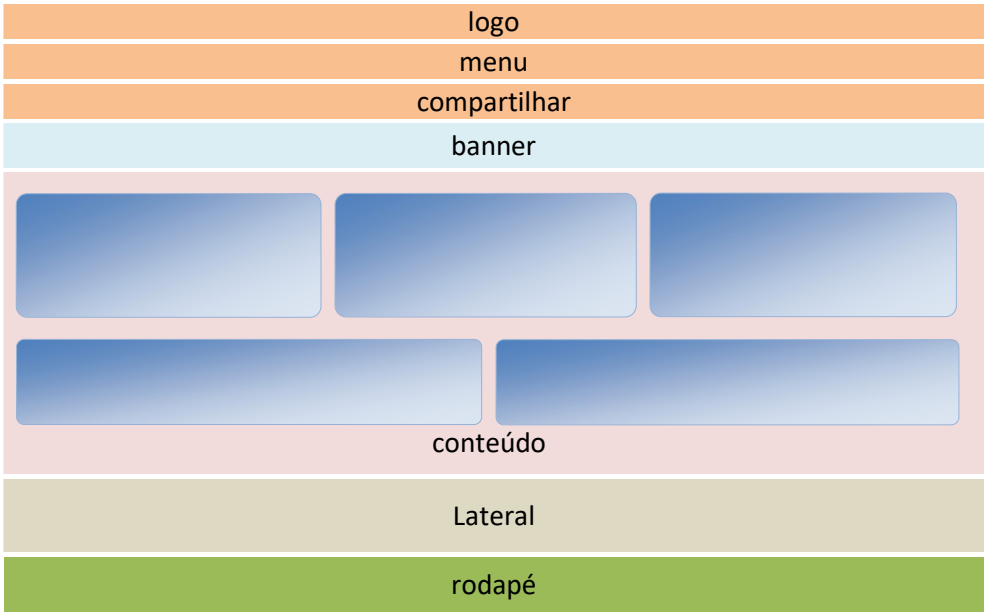
Para deixar o nosso site **responsivo**, devemos fazer alguns testes para verificar quais serão as **divs** que sofrerão ajustes de largura para não afetar a visualização do seu site. Assim como fizemos na **barra**, a **barra3** também deve ter a direção flex modificada para **column** e alinhamento **stretch**. Verifique se outros elementos do site não precisam de modificações para telas menores.

```
@media screen and (max-width:800px) {
    .barra, .barra3 {flex-direction:column; align-items:stretch;}
}
```

Vale lembrar que com as configurações CSS feitas para telas menores, nosso **menu** que originalmente é horizontal passa a ser um menu vertical. Logo, para fazer um site com menu vertical, podemos usar a mesma estrutura, com direção flex **column**.

Ajuste os outros elementos da página, tais como imagens, as caixas Box, itens do menu. Quando abrimos esta página em telas menores, alguns elementos ficam mal posicionados ou com tamanhos desproporcionais.

Nosso site ficará com um layout parecido com o apresentado a seguir. Dependendo do tamanho dos Box, eles podem aparecer um abaixo do outro, por causa do display flex.



Atividade 7: Utilize nosso site tableless formatado para inserir conteúdos com um tema, criando 3 páginas com imagens, vídeos e animações. Edite as cores de animações e backgrounds para os sites usando CSS.

Agora vamos aprender como utilizar um vídeo como imagem de fundo de uma página. Utilizaremos os códigos estruturais HTML, classes CSS e funções de JavaScript. Vamos começar com a estrutura HTML, que basicamente terá a inserção de um vídeo com a tag **<video>** que já utilizamos nas primeiras atividades, além de uma tag ** ** de conteúdo.

Insira o caminho do vídeo e crie um identificador chamado **VideoFundo** para ser formatado com CSS e utilizado como entrada da função programada em JavaScript. Utilizaremos também um botão de controle com a tag **<button>** para que o visitante possa pausar o vídeo quando achar necessário. Nesta tag, podemos colocar que ao clicar (**onclick**), a função chamada **minhaFuncao()** que estará programada com JavaScript será ativada. Vamos formatar o botão com CSS criando o identificador **botao**.

```
<video autoplay muted loop id="VideoFundo" src="videos/marine.mp4"></video>
<button id="botao" onclick="minhaFuncao()">Pause</button>
<div class="conteudo">
	<h2>Minha página com vídeo no fundo</h2>
	<p>Podemos clicar no botão para pausar o vídeo!</p>
</div>
```

Vamos posicionar o conteúdo a partir da metade da altura da tela (50vh), colocando um fundo com opacidade de 50% e alinhamento de texto centralizado. O vídeo ficará fixo no fundo, com as coordenadas **top:0** e **left:0**, ou seja, a partir do canto superior esquerdo da tela. É importante utilizar o atributo **z-index: -1** para que o vídeo fique atrás dos demais elementos da página.

```
.conteudo {position: fixed; top: 50vh; background: rgba(200,200,200,0.5); text-align: center; width: 90%}
#VideoFundo {position: fixed; top: 0; left: 0; min-width: 100%;  min-height: 100%; z-index: -1;}
```

As propriedades do botão podem ser definidas com cores, tamanho de fonte, largura, altura, margens e sua posição na tela.

```
#botao {position: fixed; right: 0; bottom: 0; width: 200px; font-size: 18px; padding: 10px; border: none;
	background: red; color: #fff; cursor: pointer;}
#botao:hover {background: cyan; color: black;}
```

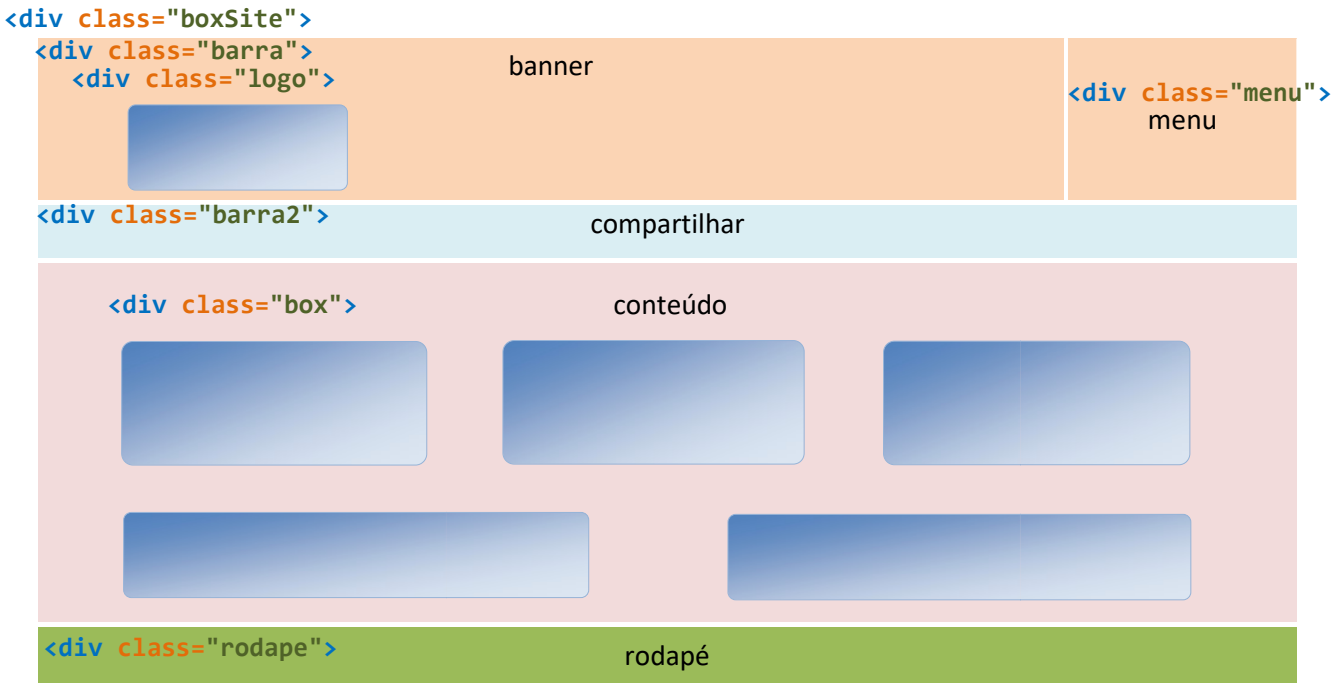
Dentro da tag **<body>** da página, podemos inserir a função programada em JavaScript para pausar o vídeo.

```
<script>
        var video=document.getElementById("VideoFundo");
        var btn=document.getElementById("botao");
        function minhaFuncao() {
          if (video.paused) {
                video.play();
                btn.innerHTML="Pause";
          } else {
                video.pause();
                btn.innerHTML="Play";
          }
        }
</script>
```

O layout desta página fica da seguinte forma:

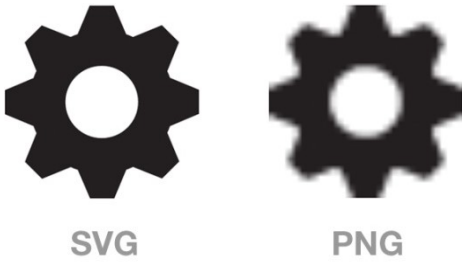
Atividade 8: Crie um site com layout **tableless** (div) similar ao exemplo a seguir, com os seguintes elementos:

- logo e menu que "flutuem" sobre o banner;
- use CSS para criar efeitos de gradiente e transparência em elementos do site;
- use animação para alguns elementos do site;
- no conteúdo, crie uma galeria de imagens e/ou vídeos com efeitos de animação **hover**;
- insira um vídeo como fundo do site;
- use **border-radius**, **text-shadow** e **box-shadow** para formatar elementos do layout;
- insira um menu com 3 links de páginas neste layout.



9. SCALABLE VECTOR GRAPHICS - SVG

Podemos criar logotipos, banners e gráficos em páginas da internet com o uso de desenhos com formato **SVG** (**Scalable Vector Graphics**), que possuem melhor resolução do que imagens em jpg, png ou bmp.



O formato **SVG** é basicamente um conjunto de descrições matemáticas em um arquivo **XML** (Extensible Markup Language) que define uma forma geométrica. Desta forma, o desenho é um conjunto de vetores criados com instruções numéricas.

Quando trabalhamos com imagens png, jpg ou bmp, as formas estão colocadas em uma grade de pixels, que perde a resolução com mais facilidade. Muitos softwares permitem a criação de imagens em svg. A partir do código fonte do arquivo SVG criado, podemos extrair somente as tags importantes para carregá-las em nossos sites.

As tags SVG mais simples são recomendadas, pois são carregadas mais rapidamente nos navegadores. A seguir, veremos como as tags nativas SVG são estruturadas através de seus comandos.

Já vimos algumas tags nas atividades de mapeamentos de imagens. Agora podemos usar mais algumas tags para desenhar formas simples em nossas páginas. Formas complexas podem ser desenhadas em editores de SVG, tais como o **SVG Editor**, **DrawSVG**, **Inkscape** ou **Illustrator**. Depois, basta copiar o código HTML do desenho e colar no código da sua página.

Vamos começar criando uma página com SVGs dentro da tag **logo**. Podemos repetir a tag de barra para testarmos desenhos de vários SVG para nossa próxima Atividade. Na tag de **banner**, vamos inserir apenas um título.

```
<div class="boxSite">
  <div class="barra">                    caixa
    <div class="logo">   <div class="banner">      <nav class="menu">
          logo                 banner                   menu
```

Os desenhos de linhas podem ser feitos com a tag **line**. Para desenharmos retângulos, círculos, elipses e polígonos, usamos as tags **<rect>**, **<circle>**, **<ellipse>** e **<polygon>**, respectivamente. Para incluir texto, usamos a tag **<text>**. Veja alguns exemplos destas tags:

```
<line x1="10" y1="10" x2="30" y2="50" stroke="grey" stroke-width="1"/>
<rect x="10" y="50" width="10" height="100" fill="orange" stroke="grey" stroke-width="1"/>
<circle r="50" cx="50" cy="70" fill="rgb(100%,90%,0%)" stroke="black" stroke-width="1"/>
<ellipse cx="100" cy="75" rx="60" ry="30" fill="orange" stroke="blue" stroke-width="1"/>
<polygon points="200,10 250,190 160,210" fill="rgb(203,200,12)" stroke="green" stroke-width="2"/>
<text x="50" y="120" fill="aqua" stroke="blue" stroke-width="1">Seu texto aparece aqui</text>
```

Quando vários elementos compartilham as mesmas propriedades, podemos agrupá-los com a tag **<g>**, informando estas propriedades no arquivo CSS ou dentro da própria tag. Assim, o código da página fica otimizado, carregando mais rapidamente nos navegadores. No exemplo mostrado a seguir, todos os elementos têm mesma espessura (**stroke-width**) e cor (**stroke**) de linha:

```
<g stroke="grey" stroke-width="1">
    <line x1="10" y1="10" x2="30" y2="50"/>
    <rect x="10" y="50" width="10" height="100" fill="orange"/>
    <circle r="50" cx="50" cy="70" fill="rgb(100%, 90%, 0%)"/>
</g>
```

Esta tag de grupo é usada para criarmos as animações e efeitos gradientes. Ao criar um grupo com a tag **<g>** podemos usar uma transformação geométrica nos elementos que estão nesta tag para modificá-los simultaneamen-

te. As transformações são: rotação (**rotate**), translação (**translate**), escala (**scale**) e cisalhamento (**skewX** e **skewY**), similares às que usamos com CSS nas galerias de imagens e vídeos das Atividades anteriores.

Vamos testar estas transformações para criar um desenho de logo da nossa página. Podemos fazê-las em tags de grupos **<g>**, ou diretamente em elementos de linhas, círculos ou retângulos. A notação para a **rotação** de um elemento segundo ângulo θ, usando como centro o ponto de coordenadas (x,y) é a seguinte:

```
transform="rotate(θ,x,y)"
```

Quando as coordenadas x e y não são informadas, a rotação é feita a partir da origem das coordenadas (0,0), e a tag fica desta forma:

```
transform="rotate(θ)"
```

A translação é feita a partir do ponto que a figura se encontra até o ponto com deslocamentos dx e dy nos eixos x e y. A notação é a seguinte:

```
transform="translate(x,y)"
```

A mudança de escala (redimensionamento ou homotetia) é feita a partir do ponto da figura mais próximo da origem (0,0). Se for colocada apenas com um valor, é feita a mudança de escala igual para x e y. Para mudança diferente de escala, basta colocar primeiro o valor de mudança de escala em x e depois em y.

```
transform="scale(d)"  ou  transform="scale(dx,dy)"
```

Quando usarmos a mudança de escala com valores (-1,1) temos o efeito de espelhamento da imagem da figura em torno do eixo y; com valores (1,-1) teremos o espelhamento da imagem em torno do eixo x:

```
transform="scale(-1,1)"  ou  transform="scale(1,-1)"
```

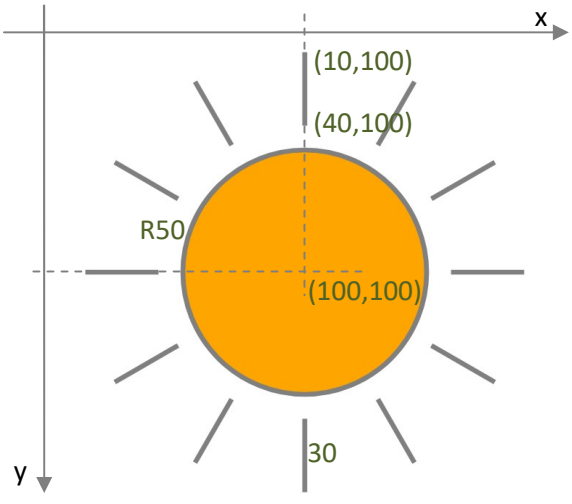
A função de cisalhamento cria o efeito de inclinação de ângulo θ em x ou y. A notação é a seguinte:

```
transform="skewX(θ)"  ou  transform="skewY(θ)"
```

Podemos criar também uma sequência de transformações para uma figura ou grupo. Basta colocar na tag a sequência das transformações separadas por espaço, como fizemos em CSS. Por exemplo, uma translação de 20px em x e 40px em y seguida por uma rotação de 45° em torno do ponto de coordenadas (70,80) ficaria assim:

```
transform="translate(20,40) rotate(45,70,80)"
```

Vamos utilizar as transformações para criar um desenho de sol na tag **logo** do nosso primeiro banner. Podemos criar uma janela **SVG** de 200px x 200px dentro da tag de logo.




```
<div class="logo">
   <svg viewBox="0 0 200 200">
      <g stroke="grey" stroke-width="1">
         <circle r="50" cx="100" cy="100" fill="orange"/>
         <line x1="100" y1="10" x2="100" y2="40"/>
      </g>
    </svg>
</div>
```

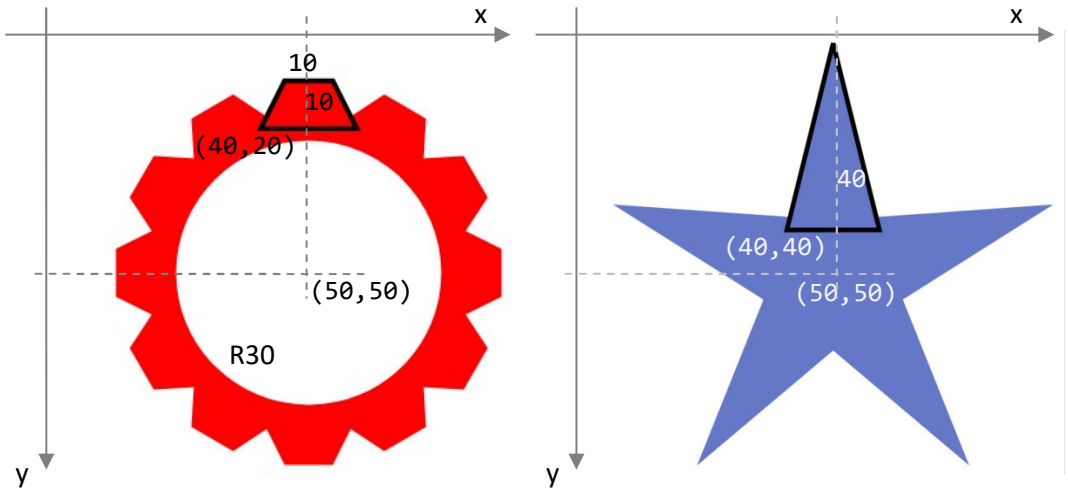
Usando os mesmos atributos de espessura e cor de linha no círculo e na linha, colocamos estes valores na tag de grupo **<g>**. Nesta tag podemos criar os outros raios, pois eles são facilmente encontrados por meio de rotações em torno do centro do círculo. Se quisermos 12 raios, como mostra o desenho, podemos rotacionar a primeira linha com 30°, depois 60° e assim sucessivamente.

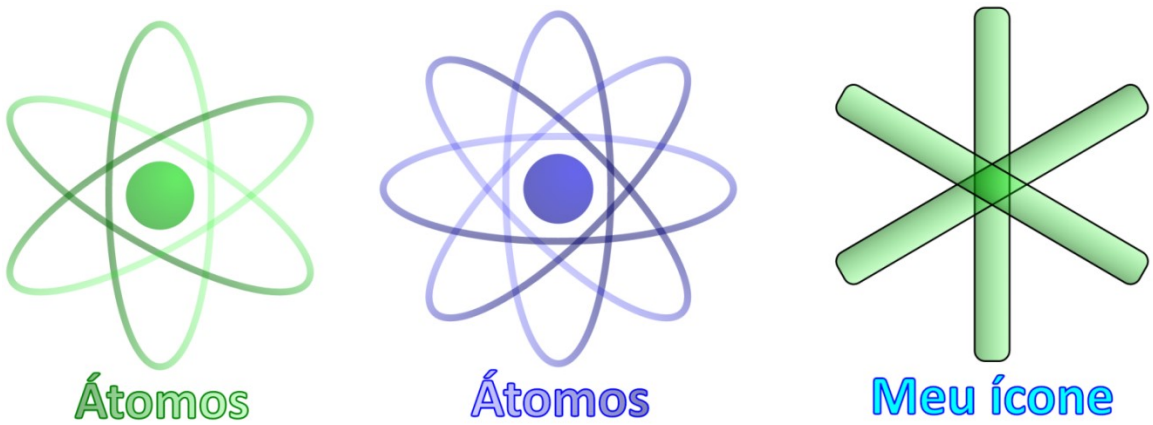
Logo, a estrutura para desenhar o sol fica assim:

```
<div class="logo">
  <svg viewBox="0 0 200 200">
     <g stroke="grey" stroke-width="1" id="sol">
         <circle r="50" cx="100" cy="100" fill="orange"/>
         <line x1="100" y1="10" x2="100" y2="40"/>
         <line x1="100" y1="10" x2="100" y2="40" transform="rotate(30,100,100)"/>
         <line x1="100" y1="10" x2="100" y2="40" transform="rotate(60,100,100)"/>
         ...
         <line x1="100" y1="10" x2="100" y2="40" transform="rotate(330,100,100)"/>
     </g>
   </svg>
</div>
```

Desenhos de engrenagens e estrelas também podem ser feitos com os atributos que usamos no desenho do sol.

Exercício 13: Crie mais duas **class=”barra”** na página svg para inserir um logo em cada linha. Escolha entre as sugestões abaixo ou crie outros desenhos.





Para atribuir transparência no preenchimento das figuras svg, usamos **fill-opacity**. Este valor varia entre 0 e 1. Para atribuir transparência somente nas linhas do desenho, usamos **stroke-opacity**. Porém, se você quiser atribuir a opacidade nas linhas e no preenchimento, o atributo tem o mesmo nome que já usamos no CSS: **opacity**.

Os três últimos exemplos do exercício anterior usam transparência: no último somente no preenchimento, e nos dois anteriores (dos átomos), a transparência é usada nas linhas.

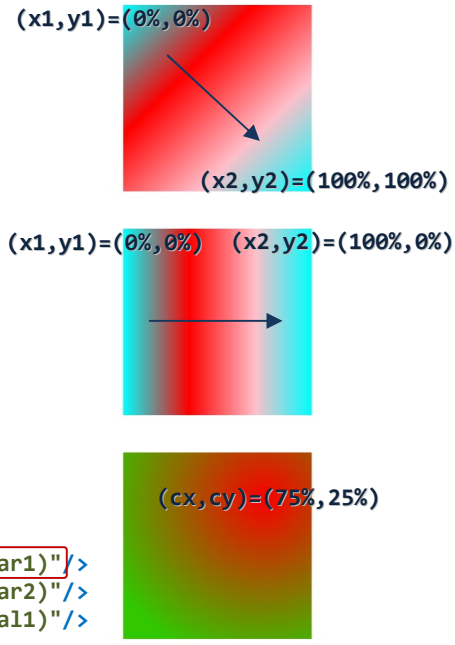
A propriedade **stroke-dasharray** serve para criar linhas tracejadas, e pode ser usada como um vetor com os valores das distâncias de linhas contínuas e tracejadas, como mostrados nos exemplos a seguir.

```
<line x1="0" y1="10" x2="100" y2="10" stroke="blue" stroke-width="1" stroke-dasharray="10 5 15 8"/>
```

```
<line x1="0" y1="10" x2="100" y2="10" stroke="red" stroke-width="1" stroke-dasharray="10 5"/>
```

As propriedades de preenchimento gradiente podem ser usadas criando tags **<defs>** com **id**. No exemplo a seguir, criamos as definições dos efeitos e a propriedade **fill** é usada com os efeitos. Os gradientes também podem ser aplicados nos contornos dos desenhos. As coordenadas **x1** e **y1** definem o ponto de início da direção do gradiente linear; e as coordenadas **x2** e **y2** definiem o ponto do final da direção do gradiente linear. As coordenadas **cx** e **cy** definem o centro do gradiente radial; o valor **r** define o raio de cobertura do efeito de gradiente radial.

```
<svg viewBox="0 0 35 80">
    <defs>
       <linearGradient id="linear1" x1="0%" y1="0%" x2="100%" y2="100%">
           <stop offset="0%" stop-color="cyan"/>
           <stop offset="35%" stop-color="red"/>
           <stop offset="70%" stop-color="pink"/>
           <stop offset="100%" stop-color="cyan"/>
       </linearGradient>
       <linearGradient id="linear2" x1="0%" y1="0%" x2="100%" y2="0%">
           <stop offset="0%" stop-color="cyan"/>
           <stop offset="35%" stop-color="red"/>
           <stop offset="70%" stop-color="pink"/>
           <stop offset="100%" stop-color="cyan"/>
       </linearGradient>
       <radialGradient id="radial1" cx="75%" cy="25%" r="90%">
           <stop offset="0%" stop-color="rgb(250,0,0)"/>
           <stop offset="50%" stop-color="rgb(150,100,0)"/>
           <stop offset="100%" stop-color="rgb(50,200,0)"/>
       </radialGradient>
    </defs>
    <rect x="10" y="10" width="20" height="20" stroke="none" fill="url(#linear1)"/>
    <rect x="10" y="32" width="20" height="20" stroke="none" fill="url(#linear2)"/>
    <rect x="10" y="54" width="20" height="20" stroke="none" fill="url(#radial1)"/>
</svg>
```



Exercício 14: Crie tags **defs** para criar efeitos gradientes nas figuras que você desenhou no Exercício 13.

Podemos criar filtros para mais efeitos nos nossos desenhos. O primeiro é o de desfocar (esfuminho), chamado de filtro Gaussiano, que deve ser colocado dentro da tag **<defs>**. O exemplo a seguir pode ser colocado em um dos desenhos do Exercício 13.

```
<defs>
    <filter id="filtro1" height="200%" width="200%">
       <feGaussianBlur in="SourceGraphic" stdDeviation="1"></feGaussianBlur>
    </filter>
</defs>
```

O atributo **in=”SourceGraphic”** quer dizer que o filtro será aplicado ao desenho original. Indicaremos o filtro dentro da respectiva **div** do desenho através do link do **filter**. Os números negativos em x e y indicam que o efeito pode ultrapassar os limites do desenho, evitando falhas nos contornos.

```
<g id="sol" filter="url(#filtro1)">
```

O atributo **in=”SourceAlpha”** aplica um filtro de sombra da figura, como uma máscara do desenho original. Quando precisamos mesclar vários filtros, usamos a tag **<feMerge>** com as referências dos resultados dos filtros. O exemplo a seguir mistura efeito de sombra com a imagem desfocada da figura.

```
<filter id="filtro2" x="-20" y="-20" height="50" width="50">
    <feGaussianBlur in="SourceAlpha" stdDeviation="2" result="sombra"></feGaussianBlur>
    <feGaussianBlur in="SourceGraphic" stdDeviation="0.5" result="desfocado"></feGaussianBlur>
    <feMerge>
        <feMergeNode in="sombra"/>
        <feMergeNode in="desfocado"/>
    </feMerge>
</filter>
```

Para deixar a sombra da figura deslocada usamos a propriedade **<feOffset>**.

```
<filter id="filtro3" x="-20" y="-20" height="50" width="50">
    <feGaussianBlur in="SourceAlpha" stdDeviation="2" result="desfocado"></feGaussianBlur>
    <feOffset in="desfocado" dx="5" dy="5" result="sombra"></feOffset>
    <feMerge>
        <feMergeNode in="sombra"/>
        <feMergeNode in="desfocado"/>
        <feMergeNode in="SourceGraphic"/>
    </feMerge>
</filter>
```

Exercício 15: Utilize tags **defs** para criar filtros nas figuras que você desenhou do Exercício 13.

Quando conhecemos as coordenadas dos pontos que formam um desenho, podemos usar os caminhos (**path**) do SVG. Estes caminhos podem ser formados por segmentos e curvas: basta informar a sequência das coordenadas. Quando usamos os comandos com a letra **maiúscula**, devemos informar as **coordenadas absolutas** de cada ponto do caminho. Porém, torna-se muito mais simples trabalhar com as coordenadas relativas. Neste caso, usando-se as letras **minúsculas**, informamos cada **coordenada relativa** ao ponto anterior do caminho.

Vamos ver alguns comandos para desenhar caminhos com o svg.

M (move to) define a posição do início de um desenho.
L ou **l** define uma linha a partir de um ponto.
H ou **h** define uma linha horizontal.
V ou **v** define uma linha vertical.

Para desenhar a linha vermelha que começa em **M(70,100)** e termina em (200,200) podemos usar **path** (caminho) e definir o comando com letra L maiúscula para usar coordenadas absolutas.

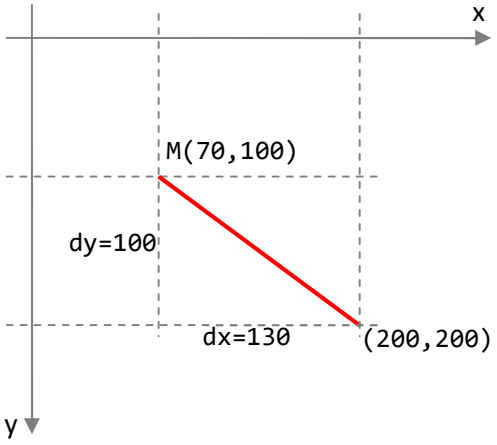
```
<path d="M70,100 L200,200" stroke="red" stroke-width="3"/>
```

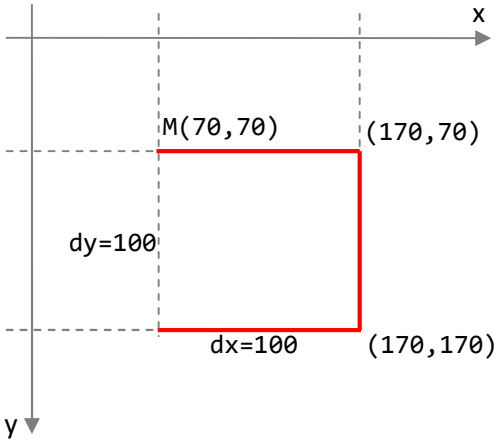
Para usar coordenadas relativas, usamos as diferenças das coordenadas absolutas dx e dy e a letra **l** minúscula:

```
<path d="M70,100 l130,100" stroke="red" stroke-width="3"/>
```

O comando **H** define uma linha horizontal desde o ponto **M** até a distância informada. Se a distância for positiva, a linha vai para a direita; se for negativa, a linha vai para a esquerda. O comando **V** desenha linhas verticais, com o mesmo raciocínio do comando **H**.

Para desenhar a linha vermelha que começa em (70,70) e termina em (70,170) podemos usar **path** (caminho) e definir o comando com letras **H** e **V** maiúsculas para usar coordenadas absolutas:





```
<path d="M70,70 H170 V170 H70" stroke="red" stroke-width="3" fill="none"/>
```

Para usar coordenadas relativas, usamos as diferenças das coordenadas absolutas **dx** e **dy** e as letras **h** e **v** minúsculas:

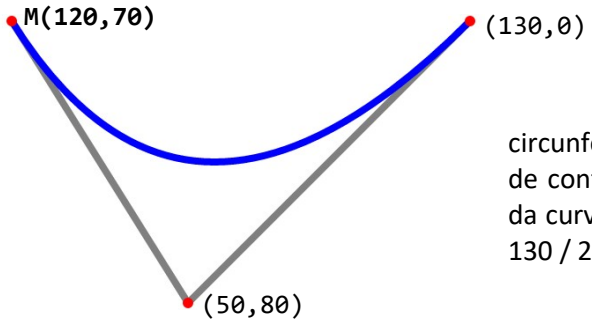
```
<path d="M70,70 h100 v100 h-100" stroke="red" stroke-width="3" fill="none"/>
```

O comando **Z** desenha uma linha que fecha uma poligonal. No exemplo anterior, se usarmos o comando **Z** no final do caminho **path**, o resultado é o desenho do quadrado:

```
<path d="M70,70 h100 v100 h-100Z" stroke="red" stroke-width="3" fill="none"/>
```

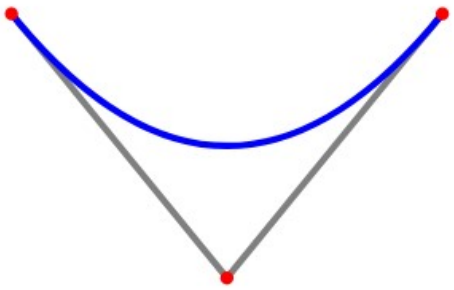
A tag de curva Bezièr quadrática pode ser usada para desenharmos caminhos com cantos arredondados. O comando é **Q** para coordenadas absolutas e **q** para coordenadas relativas. As primeiras coordenadas são do ponto de controle, e as próximas coordenadas são do ponto final da curva. Os segmentos de controle não aparecem na tela quando usamos este comando **svg**.

```
<path d="M120,70 q50,80 130,0" stroke="blue" stroke-width="2" fill="none"/>
```



Neste caso, como fazer um arco similar ao de uma circunferência com este comando? Basta usar as coordenadas do ponto de controle na mediatriz do segmento que une os pontos inicial e final da curva. Neste exemplo, a coordenada x do ponto de controle seria x = 130 / 2 = 65.

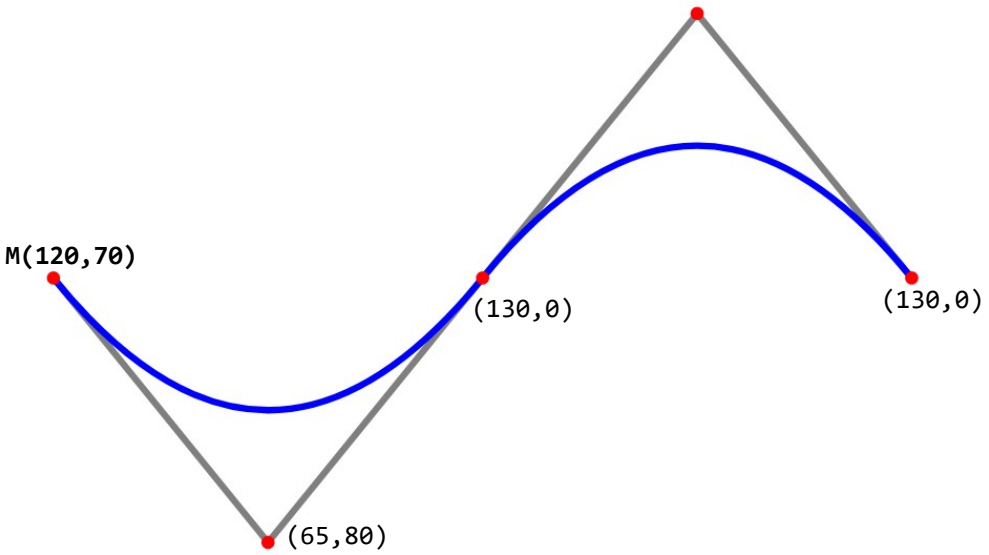
```
<path d="M120,70 q65,80 130,0" stroke="blue" stroke-width="2" fill="none"/>
```



O comando **T** ou **t** faz a continuação da curva de Bèzier, mas no semi-plano oposto. O ponto final da primeira parte fica sendo o ponto de tangência entre as duas curvas. Se usarmos a mesma variação usada para a primeira parte (ou seja, dx = 130), a curva fica simétrica em relação ao ponto de tangência.

A tag destas curvas simétricas fica assim:

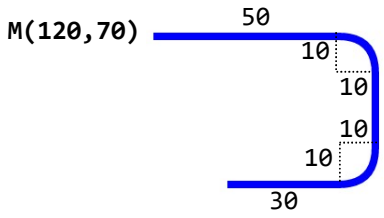
```
<path d="M120,70 q65,80 130,0 t130,0" stroke="blue" stroke-width="2" fill="none"/>
```



Com estas tags, podemos fazer caminhos para diversos desenhos e alguns para inserir texto com animações.

Veja como fica o uso da curva de Bèzier quadrática em um caminho com cantos arredondados:

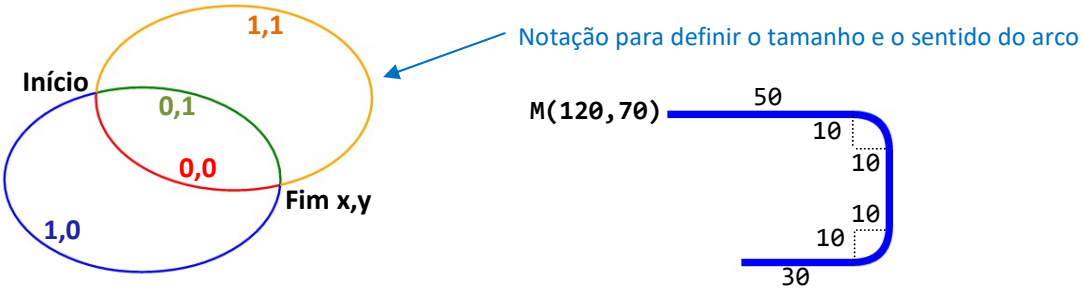
```
<path d="M120,70 h50 q10,0 10,10 v20 q0,10 -10,10 h-30" stroke="blue" stroke-width="2" fill="none"/>
```



Para desenhar os arcos de circunferência ou elipse usamos o comando **a** com os seguintes atributos:

a rx,ry θ tamanho,sentido x,y

onde:
rx e **ry** são os raios em relação aos eixos;
θ é o ângulo de rotação em relação ao eixo x;
tamanho é 0 quando você quer o arco menor, e 1 quando precisar do arco maior;
sentido é 0 quando for desenhado no sentido anti-horário e 1 quando for no sentido horário;
x e **y** são as coordenadas finais do arco.

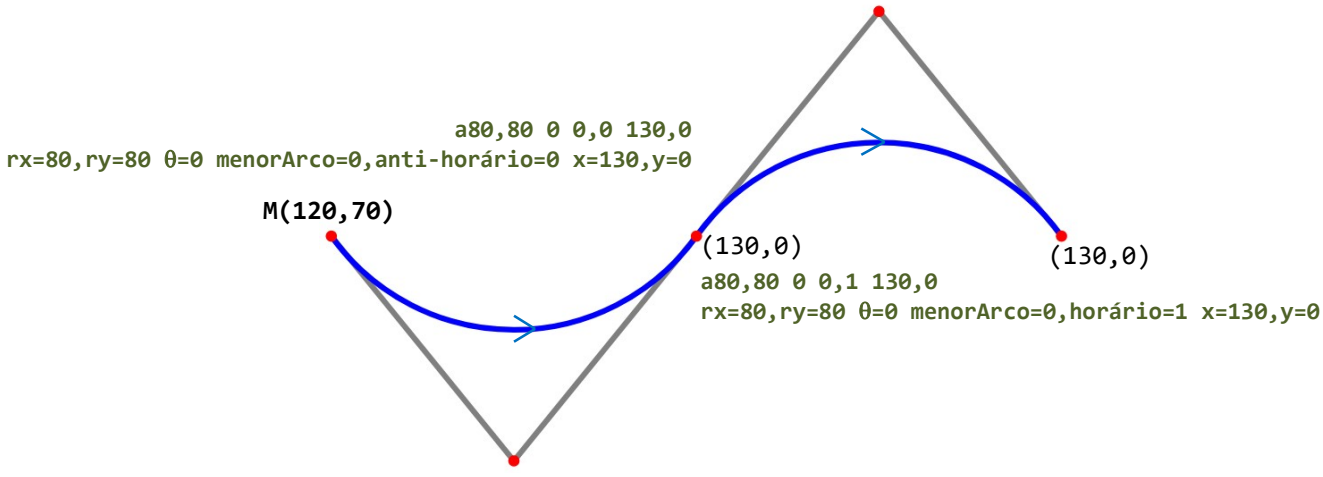


Voltando ao exemplo dos cantos arredondados, podemos desenhar a figura acima com os comandos:

```
<path d="M120,70 h50 a10,10 0 0,1 10,10 v20 a10,10 0 0,1 -10,10 h-30" stroke="blue" fill="none" stroke-
    width="2"/>
```

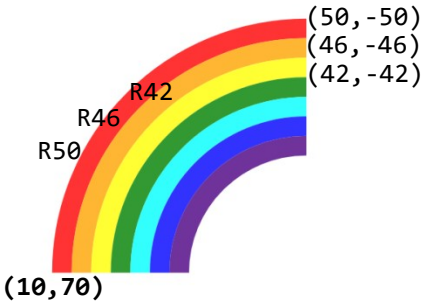
No exemplo da curva de Bèzier, os arcos de circunferências aproximados têm raios iguais a 80, e usamos os seguintes comandos para desenhá-los:

```
<path d="M120,70 a80,80 0 0,0 130,0 a80,80 0 0,1 130,0" stroke="blue" stroke-width="2" fill="none"/>
```



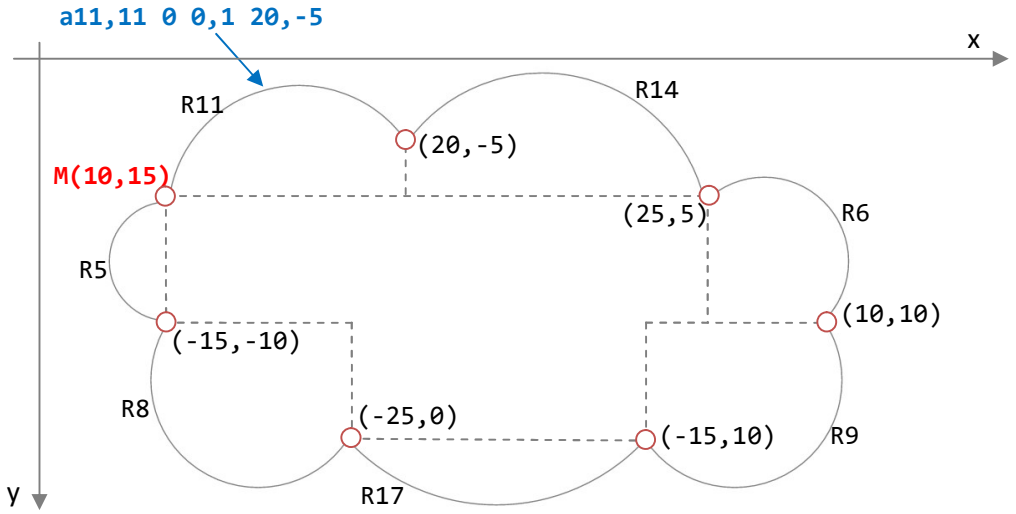
Se for possível usar a curva quadrática de Bèzier ao invés de arcos, os comandos ficam bem mais simples. Podemos sempre ajustar a curva para ficar semelhante ao arco que precisamos desenhar, conforme mostrado nos exemplos.

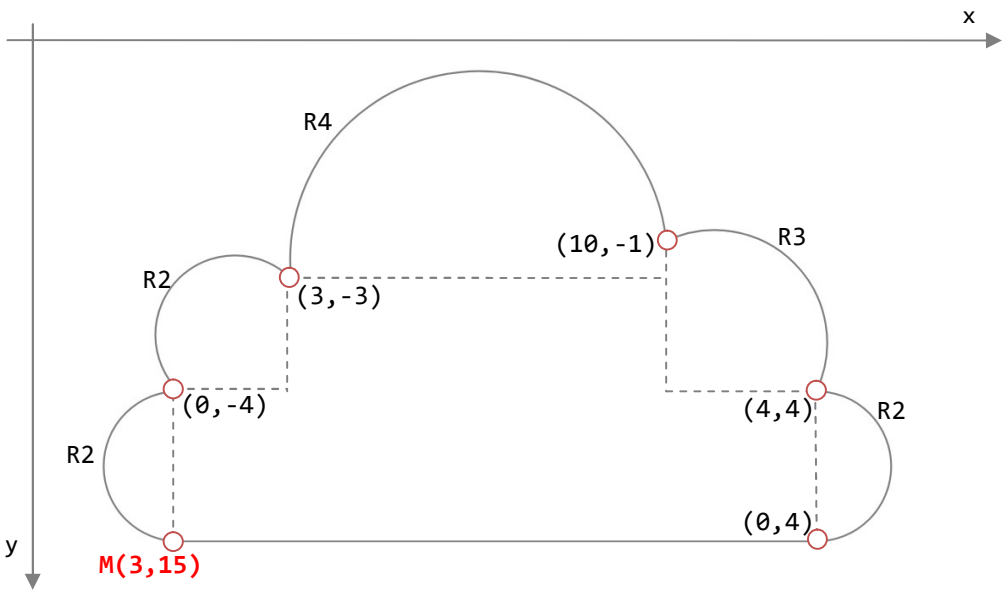
Veja mais um exemplo do uso do comando de arcos. Para desenharmos um arco-íris, podemos usar uma janela de visualização de 100x100, com um ponto inicial do arco vermelho em **(10,70)**. Se considerarmos o primeiro arco com raio de 50, podemos diminuir os próximos raios de acordo com a espessura usada em todos os arcos. Neste exemplo, utilizamos a espessura 4.



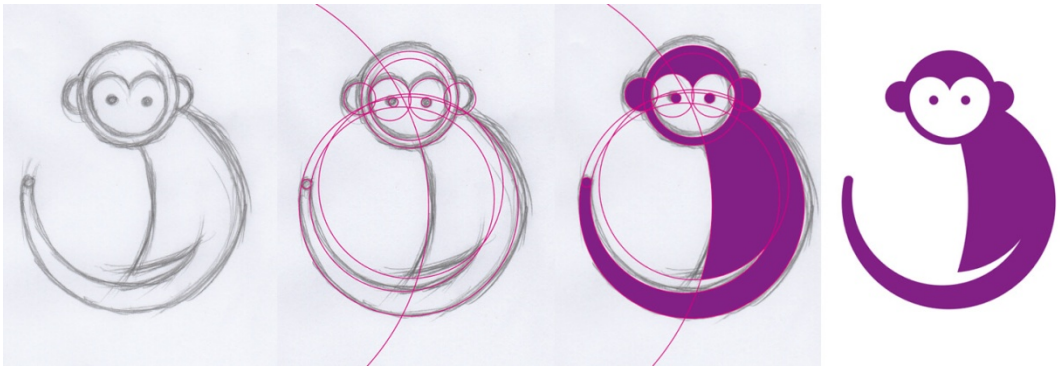

```
<g id="arcoiris" stroke-width="4" opacity="0.8">
   <path d="M10,70 a50,50 0 0,1 50,-50" fill="none" stroke="red"/>
   <path d="M14,70 a46,46 0 0,1 46,-46" fill="none" stroke="orange"/>
   <path d="M18,70 a42,42 0 0,1 42,-42" fill="none" stroke="yellow"/>
   <path d="M22,70 a38,38 0 0,1 38,-38" fill="none" stroke="green"/>
   <path d="M26,70 a34,34 0 0,1 34,-34" fill="none" stroke="aqua"/>
   <path d="M30,70 a30,30 0 0,1 30,-30" fill="none" stroke="blue"/>
   <path d="M34,70 a26,26 0 0,1 26,-26" fill="none" stroke="indigo"/>
</g>
```

Exercício 16: Utilize as tags de arcos para desenhar cada nuvem abaixo, usando para cada nuvem uma tag de caminho **path**. Faça cada nuvem em uma **viewBox** diferente, como logos para nossas barras.

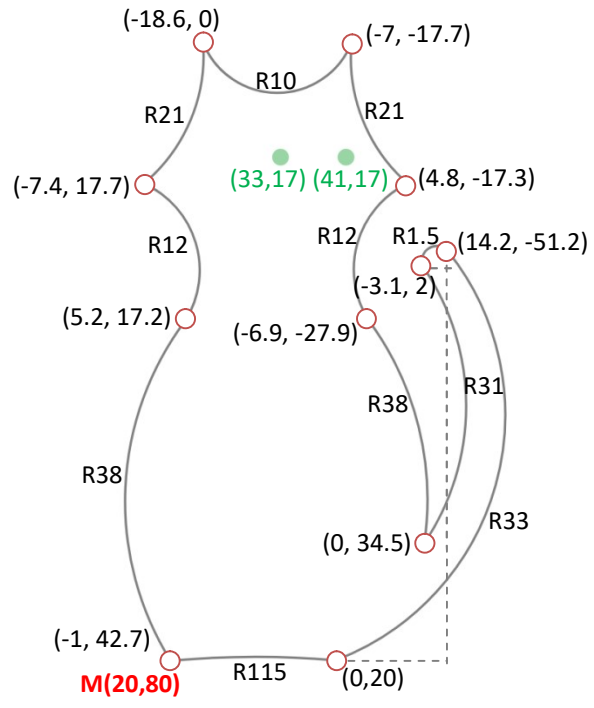




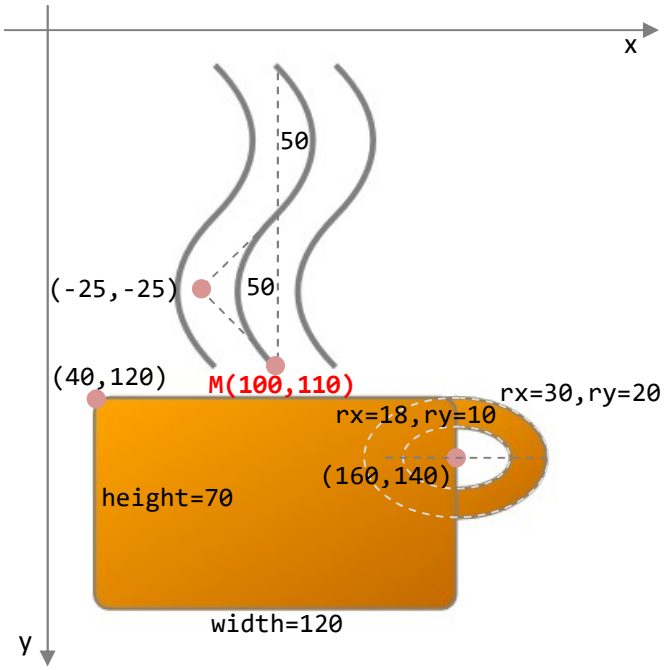
Os desenhos mostrados a seguir são adaptados do trabalho de Dorota Pankowska, que mostra 13 desenhos de animais com 13 arcos de circunferências. Tais desenhos podem ser feitos com tags SVG similares às que usamos para desenhar as nuvens.



Illustating animals with 13 circles, Dorota Pankowska (2016)



Exercício 17: Utilize as tags svg de elipses, curva Bèzier, path e retângulo para desenhar o pictograma de xícara. Use uma janela de visualização de 200px x 200px. Crie um grupo **<g>** para as curvas da fumaça e outro grupo **<g>** para a xícara.



A animação com tags SVG é bem simples e exige apenas atenção para inserirmos corretamente os efeitos. Uma informação importante é que os desenhos com muitas tags devem estar agrupados em tags **<g>** ou **<set>** para serem animados. Podemos animar praticamente todos os atributos de um desenho de SVG.

As animações em SVG podem ser feitas usando CSS (que deixa a animação um pouco lenta) ou diretamente nas tags SVG. A linguagem que a animação SVG usa é baseada na especificação SMIL (Synchronized Multimedia Integration Language).

As tags são as seguintes:
<g>, **<set>**, **<animate>**, **<animateColor>**, **<animateTransform>**, **<mpath>** **e** **<animateMotion>**.

Aplica transformações geométricas nos desenhos

Anima atributos do desenho

Muda a cor de atributos durante certo tempo

Insere desenhos em caminhos path

O objeto que será animado pode ser indicado usando **xlink:href="#id"** ou sendo o elemento principal de um bloco onde você define a animação com uma das tags acima.

A forma mais simples de animação utiliza a variação do valor de um atributo de um elemento **(<animate>)**, com certa duração. Os valores intermediários não informados são determinados através de interpolação. Podem ser definidos mais valores intermediários, para ter mais controle da animação com o uso do atributo **values**.

Vamos fazer algumas animações com atributos do desenho do arco-íris da página anterior. O primeiro deles é a **opacidade**. Para animar todas as partes do desenho com mesmo efeito, inserimos a tag de animação dentro da tag de grupo do arco-íris. Nesta tag, definimos a duração, o atributo, os valores inicial e final do atributo, o instante do início da animação e o estado final do preenchimento do desenho (se não for informado, o desenho some).

```
<g id="arcoiris" stroke-width="4" opacity="0">
   <path d="M10,70 a50,50 0 0,1 50,-50" fill="none" stroke="red"/>
   <path d="M14,70 a46,46 0 0,1 46,-46" fill="none" stroke="orange"/>
   ...
   <path d="M34,70 a26,26 0 0,1 26,-26" fill="none" stroke="indigo"/>
   <animate attributeName="opacity" from="0" to="0.8" dur="7s" begin="0s" fill="freeze"
        repeatCount="indefinite"/>
</g>
```

Nesta animação, a opacidade do desenho mudará de 0 para 0.8 em 7 segundos, mantendo-se o preenchimento da figura (`fill="freeze"`) e começando logo que a página é atualizada (`begin="0s"`). Para que a animação comece quando o visitante da página clicar na figura, podemos usar `begin="click"`.

Esta animação tem o contador de repetições com o valor **indefinite**. Se este atributo não for informado, o svg fará a animação apenas uma vez. Se você quiser um determinado número de vezes, basta informar neste contador. Por exemplo, para animar 4 vezes, colocamos `repeatCount="4"`. Para restringir a animação por um determinado período de tempo (por exemplo, 1 minuto), usamos o atributo `repeatDur="1:00"`.

Para programar o efeito vai e vem, podemos usar valores intermediários com o atributo **values**, que substitui os atributos **from** e **to**. Ao colocar o atributo como `values="0; 0.8; 0"`, a opacidade começa com 0, vai até 0.8 em 3,5 segundos e retorna para 0 em 7 segundos. A tag de animação fica assim:

```
<animate attributeName="opacity" values="0; 0.8; 0" dur="7s" begin="0s" fill="freeze"
     repeatCount="indefinite"/>
```

Para criar o efeito de animação de desenhar o contorno de um desenho, usamos animação no atributo **stroke-dashoffset**, que é a distância entre a origem do desenho (ponto M) e o primeiro ponto onde pode ser definido um tracejado na linha (**dasharray**).

Desta forma, o maior arco do arco-íris tem aproximadamente 80px de comprimento (é o comprimento de ¼ da circunferência de raio 50px, ou seja, $2 \cdot \pi \cdot 50/4 = 78.5$px), devemos colocar o atributo **stroke-dasharray="80"** na tag do grupo do arco-íris:

```
<g id="arcoiris" stroke-width="4" stroke-dasharray="80">
```

Na tag de animação, colocamos os valores da origem do tracejado:

```
<animate attributeName="stroke-dashoffset" from="80" to="0" dur="7s" begin="0s" fill="freeze"
        repeatCount="indefinite"/>
```

Se você quiser o efeito vai e vem, basta usar o atributo values no lugar de **from** e **to**:

```
<animate attributeName="stroke-dashoffset" values="80; 0; 80" dur="7s" begin="0s" fill="freeze"
     repeatCount="indefinite"/>
```

Exercício 18: Utilize os efeitos de animação de tracejado sroke-dashoffset e opacidade para desenhar a xícara do exercício 16. Faça o mesmo efeito de animação com a fumaça e crie um texto abaixo do desenho da xícara. Aumente a altura da tela de visualização viewBox para caber o texto junto com o desenho. Faça a animação com o contorno do texto.

Os efeitos de animações em svg podem ser feitos em vários atributos: altura, largura, coordenada, raio, cor de linha, cor de preenchimento, cor de gradiente. Veja outros exemplos de animações que podemos criar:

Em uma tag de um círculo:

```
<animate attributeName="cx" to="150" dur="3s" begin="click" fill="freeze"/>
```

Em uma tag de um retângulo:

```
<animate attributeName="y" to="120" dur="3s" begin="3s" fill="remove"/>
```

Em uma tag de gradiente linear, dentro de **<stop offset>**:

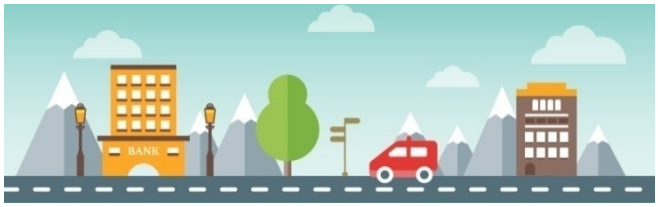
```
<animate attributeName="stop-color" from="rgb(0,0,120)" to="rgb(190,0,0)" dur="3s" begin="0s"
    fill="freeze"/>
```

Em uma tag de uma figura:

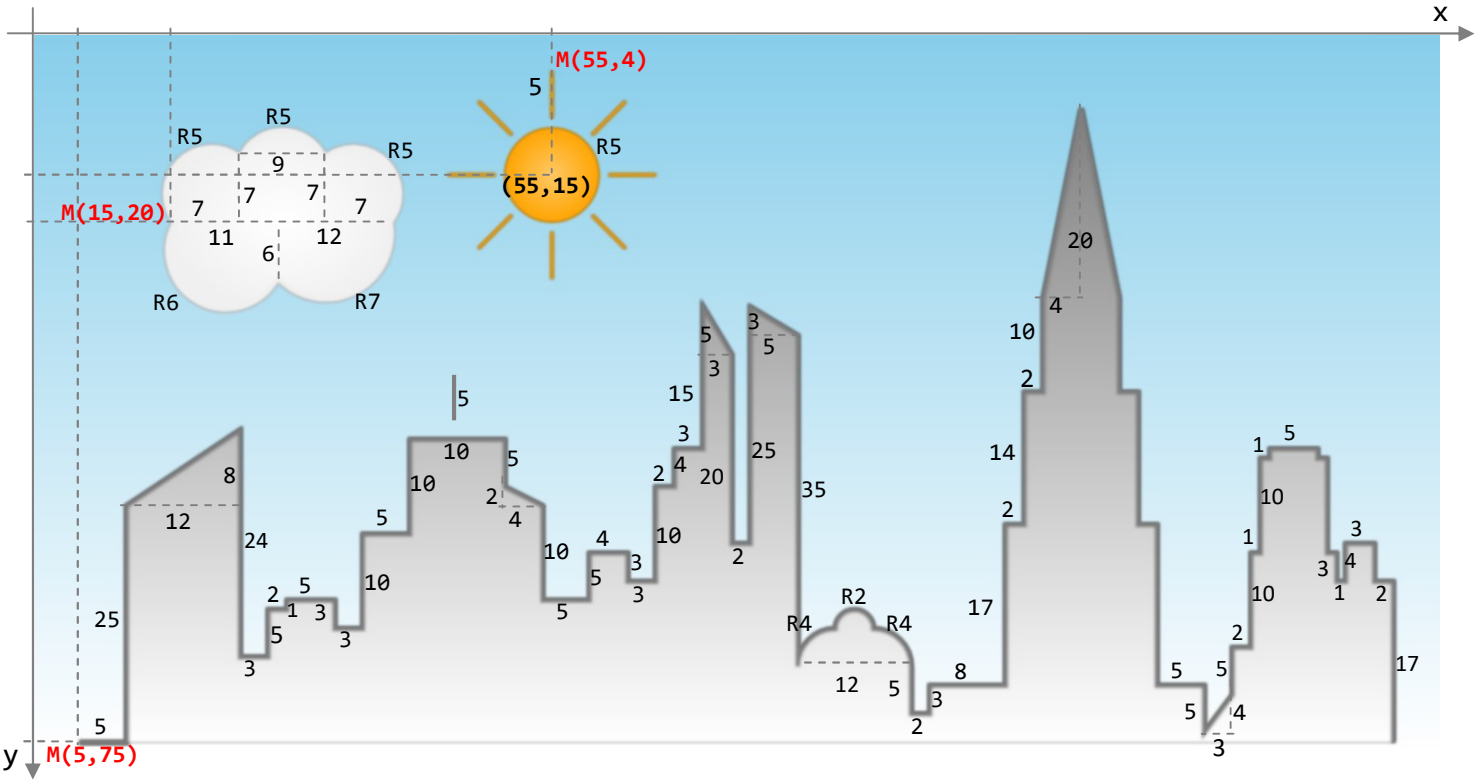
```
<animate attributeName="fill" from="rgb(0,0,250)" to="rgb(250,0,0)" dur="3s" begin="0s" fill="freeze"/>

<animate attributeName="stroke" from="rgb(0,0,250)" to="rgb(250,0,0)" dur="3s" begin="0s" fill="freeze"/>
```

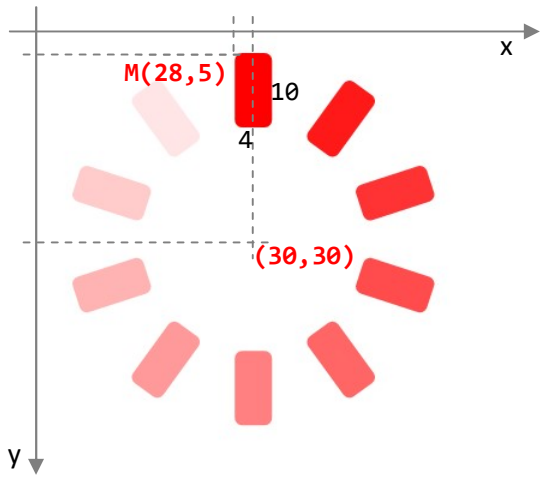
Agora vamos criar um banner com paisagem (landscape). A seguir, temos alguns exemplos interessantes para criação de banners de nossas páginas. http://www.freepik.com



Exercício 19: Crie um caminho **<path>** com os contornos dos edifícios e depois crie outro **<path>** para a nuvem. Crie um grupo para o desenho do sol. Utilize os efeitos de filtros e gradientes neste desenho. Coloque o efeito de animação para desenhar os contornos dos prédios, da nuvem e do sol. Coloque filtros nos desenhos. Insira os desenhos svg do **sol** e da **nuvem** no banner da página **index.htm**.



Agora vamos ver as animações de transformações geométricas. Vamos fazer o efeito de rotação de uma logomarca com formato redondo: será um efeito de rotação de 360° em torno do centro da logomarca. As tags de animações com transformações são **animateTransform**, onde definimos o tipo da transformação, os valores inicial e final, além do instante do início da animação. Podemos informar valores intermediários com o atributo **values**.



Usando o centro no ponto de coordenadas (30,30), podemos construir apenas um retângulo de cantos arredondados; os outros podem ser construídos com rotações, como foi feito no exercício 13.

Uma parte da tag do desenho da logo fica da seguinte forma:

```
<g name="logo" stroke-width="0" fill="red">
        <rect x="28" y="5" width="4" height="10" rx="1"/>
        <rect x="28" y="5" width="4" height="10" rx="1"
            transform="rotate(36,30,30)" opacity="0.9"/>
        <rect x="28" y="5" width="4" height="10" rx="1"
            transform="rotate(72,30,30)" opacity="0.8"/>
        ...
</g>
```

Dentro desta tag de grupo definimos a animação com rotação com a tag:

```
<animateTransform attributeType="xml" attributeName="transform" type="rotate" from="0,30,30"
      to="360,30,30" dur="10s" begin="2s" fill="freeze"/>
```

Este efeito é de apenas uma rotação de 360° em torno do centro (30,30) no sentido horário. Para fazer a rotação no sentido anti-horário basta inverter: `from="360,30,30"` e `to="0,30,30"`. Usando **repeatCount** com propriedade **indefinite**, a rotação é feita sem interrupção.

Para o efeito de vai e vem, substituímos os atributos **to** e **from** por **values**:

```
<animateTransform attributeType="xml" attributeName="transform" type="rotate"
      values="0,30,30;360,30,30;0,30,30" dur="10s" begin="2s" fill="freeze"/>
```

Neste efeito, sempre precisamos informar o ângulo e as coordenadas do centro de rotação. Se não informarmos, a rotação é feita em torno da origem do sistema de coordenadas.

Para inserir mais de um efeito de transformação, precisamos indicar o atributo `additive="sum"` nas animações novas. A primeira animação definida não precisa deste atributo. Por exemplo, para aplicar a animação de escala junto com a rotação, colocamos as tags da seguinte maneira:

```
<animateTransform attributeType="xml" attributeName="transform" type="rotate"
      values="0,30,30;360,30,30;0,30,30" dur="10s" begin="2s" fill="freeze"/>
<animateTransform attributeType="xml" attributeName="transform" type="scale" values="1;0.9;1.2" dur="10s"
      additive="sum" begin="2s" fill="freeze"/>
```

A animação de escala funciona com os valores que multiplicamos no desenho original, criando o novo desenho por homotetia. Neste exemplo, a animação começa com a escala 1, ou seja, o desenho original. Na metade da animação este desenho diminui para 90% do tamanho original e no final, fica com 120% do tamanho original. Todos estes efeitos são produzidos junto com as rotações definidas na primeira tag.

Outra transformação que pode ser usada é a translação. Para usá-la junto com a escala e a rotação no exemplo da logomarca, basta inseri-la dentro da tag do desenho.

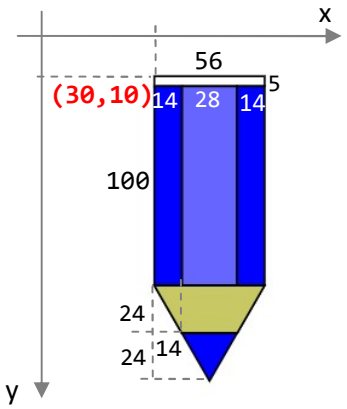
```
<animateTransform attributeType="xml" attributeName="transform" type="rotate"
      values="0,30,30;360,30,30;0,30,30" dur="10s" begin="2s" fill="freeze"/>
<animateTransform attributeType="xml" attributeName="transform" type="scale" values="1;0.9;1.2" dur="10s"
      additive="sum" begin="2s" fill="freeze"/>
<animateTransform attributeType="xml" attributeName="transform" type="translate" values="0,0;15,20;0,0"
      dur="10s" additive="sum" begin="2s" fill="freeze"/>
```

A translação começa na posição original do desenho, ou seja, nas coordenadas relativas (0,0); na metade da animação, o desenho vai para as coordenadas (15,20); no final, retorna para a posição inicial em (0,0).

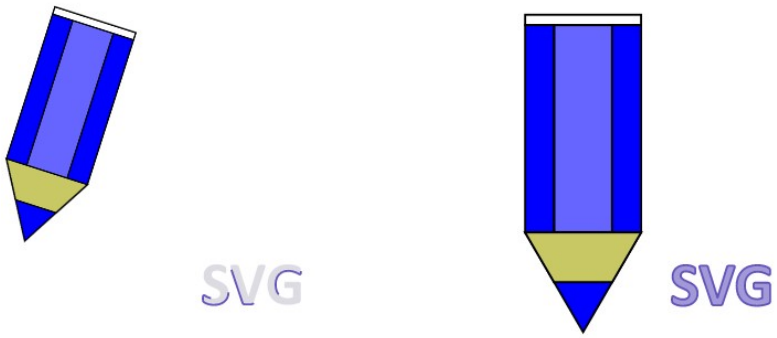
Quando estas animações são feitas com os atributos **from** e **to**, não temos o efeito de vai e vem e suas tags ficam assim:

```
<animateTransform attributeType="xml" attributeName="transform" type="scale" from="1" to="0.9" dur="10s"
      additive="sum" begin="2s" fill="freeze"/>
<animateTransform attributeType="xml" attributeName="transform" type="translate" from="0,0" to="15,20"
      dur="10s" additive="sum" begin="2s" fill="freeze"/>
```

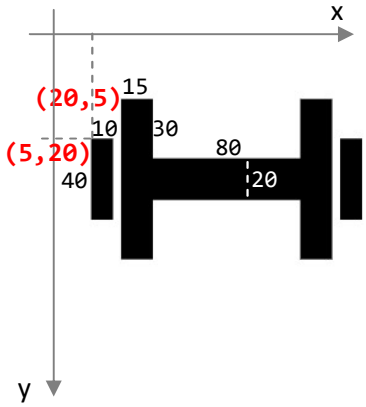
No exemplo mostrado a seguir, temos a combinação de 4 efeitos: opacidade e desenho dos contornos da parte escrita; rotação e escala do desenho do lápis.



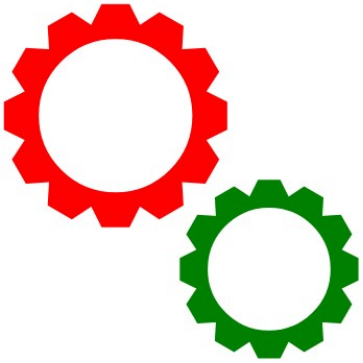

```
<g name="lapis" stroke-width="1" stroke="black">
        <rect x="30" y="10" width="56" height="5" fill="white"/>
        <rect x="30" y="15" width="14" height="100"
           fill="rgba(0,0,255,1)"/>
        <rect x="44" y="15" width="28" height="100"
           fill="rgba(0,0,255,0.6)"/>
        <rect x="72" y="15" width="14" height="100"
           fill="rgba(0,0,255,1)"/>
        <path d="M30,115 l14,24 h28 l14,-24z" fill="brown"/>
        <path d="M30,115 l14,24 h28 l14,-24z" fill="rgb(200,200,100)"/>
        <path d="M44,139 l14,24 l14,-24z" fill="blue"/>
        <animateTransform attributeName="transform" attributeType="xml"
           type="rotate" from="20,30,30" to="0,30,30" begin="0s" dur="5s"/>
        <animateTransform attributeName="transform" attributeType="xml"
           type="scale" from="0.7" to="1" begin="0s" dur="5s"
           additive="sum"/>
</g>
<text class="texto" x="100" y="150" fill="none" stroke="rgba(100,85,190,1)" stroke-width="1" stroke-
    dasharray="200">SVG
    <animate attributeName="stroke-dashoffset" from="200" to="0" dur="5s" begin="0s" fill="freeze" />
    <animate attributeName="fill" from="rgba(0,0,0,0.1)" to="rgba(100,85,190,0.6)" dur="5s" begin="0s"
    fill="freeze"/>
</text>
```



Exercício 20: Use os atributos de retângulos e caminhos para desenhar o símbolo de halteres. Crie efeitos de animação com transformações, opacidades e desenhos dos contornos.



Exercício 21: Crie efeito de transformação **rotate** na roda dentada do Exercício 13. Crie outra roda dentada com transformação **scale** para fazer animação com as duas rodas.



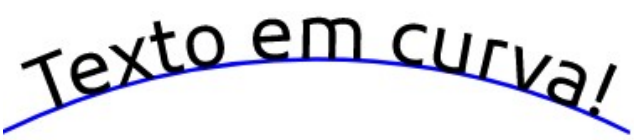
Exercício 22: Crie efeitos de transformações **scale**, **translate** e **rotate** no banner do Exercício 19.

Uma tag para criar o efeito de que o texto está seguindo uma curva pode ser feita da seguinte forma: crie uma curva com arcos com **id** chamada de curva, e faça a ligação usando a tag **textpath**, com referência à **#curva**. Uma animação pode ser feita alterando a propriedade **startOffset**.

```
<path id="curva" fill="none" stroke="blue" d="M0,30 a40,40 0 0,1 30,0 a40,40 0 0,0 30,0 a40,40 0 0,1 30,0
   a40,40 0 0,0 30,0 a40,40 0 0,1 30,0"/>
<text>
 <textpath xlink:href="#curva">Texto em curva!
       <animate attributeName="startOffset" from="-150" to="5" dur="5s" begin="0s" fill="freeze" />
  </textpath>
</text>
```

Se você quiser curvas mais simples, basta usar apenas um arco de circunferência ou de elipse:

```
<path id="curva" fill="none" stroke="blue" d="M0,30 a170,170 0 0,1 150,0"/>
```



Para esconder a curva, basta usar um dos atributos de linha: `stroke="blue"` ou `stroke-width="0"`.

Atividade 9: Escolha um tema, e crie a parte superior de uma página (logomarca e banner) com desenhos e textos com svg. Coloque animações, gradientes e filtros como efeitos de partes dos desenhos escolhidos. Coloque o banner criado, incluindo os 12 banners que criamos neste módulo de desenhos SVG, com as animações e efeitos gradientes e de filtros.

10. GALERIAS DE IMAGENS

Vamos criar uma galeria de imagens utilizando apenas o que aprendemos com **HTML, CSS** e **SVG**. Galerias de imagens com mais efeitos podem ser desenvolvidas junto com funções da linguagem **JavaScript**. Nossa próxima atividade começará de uma página estruturada, com espaço para inserir **banner**, **menu**, **galeria** e **rodapé**.

```
<div class="boxSite">                    caixa
  <div class="barra">                    banner / compartilhar
    <svg></svg>
  <div class="barra">                    menu
    <nav class="menu">
  <div class="galeria">
    <div class="mini">
```

```
  <div class="rodape">                   rodapé
```

Primeiro vamos estruturar o arquivo HTML com as tags de imagem e hiperlink. Vamos usar dois novos atributos: **id** (servirá como ligação entre a foto ampliada e a foto em miniatura) e **title** (servirá como legenda da imagem na miniatura). Podemos criar duas classes: **mini** (para as miniaturas das imagens) e **maior** (para as imagens ampliadas, que serão escondidas usando CSS). O **id** da imagem ampliada (**img1**) servirá como referência do hiperlink `href="#img1"` da imagem em miniatura.

Quando o visitante clicar na imagem miniatura com `href="#img1"`, o bloco desta imagem será ativado. Na classe chamada **bloco**, vamos inserir os botões com classes chamadas: **anterior**, **proxima** e **fechar**, colocados nesta ordem. Para a imagem ampliada aparecer com legenda, vamos colocá-la dentro da tag **figure**, incluindo a tag **figcaption** com sua respectiva legenda.

```
<div class="galeria">
    <!-- miniatura da foto -->
    <a href="#img1" class="mini" title="Meu projeto 1">
        <img src="imagens/projeto1.jpg">
    </a>
    <!-- foto ampliada, escondida pelo CSS -->
    <div class="maior" id="img1">
        <div class="bloco">
            <a href="#img12" class="botao anterior">&lt;</a>
            <figure>
                <img src="imagens/projeto1.jpg">
                <figcaption>Meu projeto1</figcaption>
            </figure>
            <a href="#img2" class="botao proxima">&gt;</a>
            <a href="#_" class="botao fechar">x</a>
        </div>
    </div>
    ...
</div>
```

Ligação entre a imagem ampliada e a imagem em miniatura.

Ligação para a imagem anterior e a próxima imagem.

A referência do hiperlink do botão **fechar** é a tag sem **id** correspondente: `href="#_"`. Ela serve justamente para que a imagem ampliada suma da tela quando o visitante clicar neste botão. Os símbolos **<** e **>** serão usados nos botões de próxima imagem e imagem anterior. Como são elementos de tags, não podem ser digitados desta

maneira. Por isso, usamos os códigos HTML `<` e `>` que significam menor (less than) e maior (greater than). No botão de fechar, podemos usar a letra X maiúscula ou minúscula.

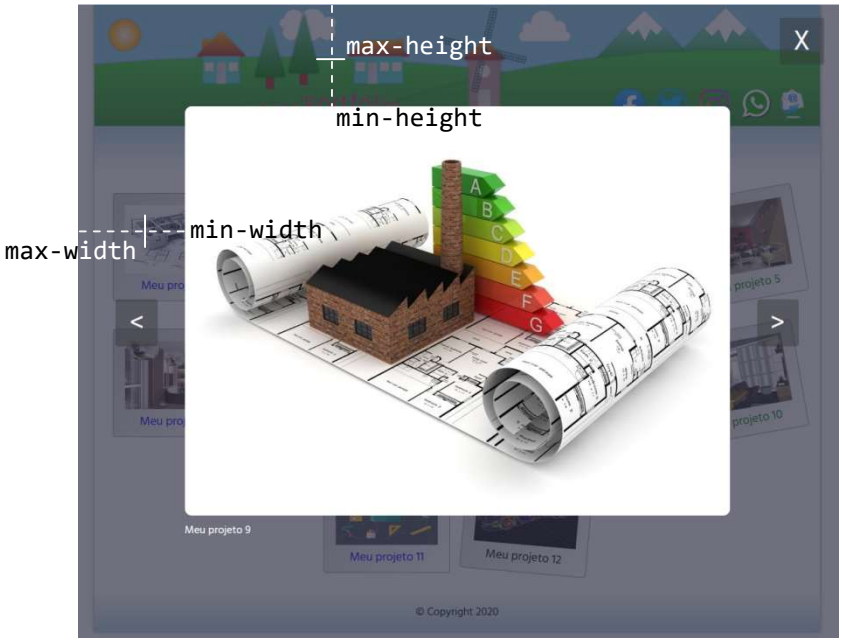
Vamos usar o atributo **flexbox** para distribuir as miniaturas das imagens na página. Com os atributos **justify-content** e **flex-wrap:wrap**, podemos alinhar as miniaturas e deixá-las sempre dentro da página.

```
.galeria {display:flex; flex-wrap:wrap; padding:10px; justify-content:center;}
```

Para esconder as imagens ampliadas, usamos **display:none** e **opacity:0** na classe **.maior**. Assim que o visitante ativar a imagem ampliada, ela deve aparecer por cima da página com miniaturas: logo, usamos **z-index:2** para os elementos desta classe. O efeito de aparecer o fundo com transparência junto com a imagem ampliada e os botões é feito usando o fundo com cor RGBA, largura de **100vw** (viewport width) e altura **100vh** (viewport height). Além disso, posicionamos este elemento em **top:0** e **left:0**.

```
.maior {display:none; opacity:0; z-index:2; width:100vw; height:100vh; text-align:center; color:white;
        text-decoration:none; position:fixed; top:0; left:0; background:rgba(50,50,75,0.7);}
```

Podemos usar as medidas de altura e de largura máxima e mínima que relacionam a imagem com o tamanho do navegador. Para mostrar sempre os botões e a legenda, as medidas de altura e largura máximas ficarão com valores menores do que 100.



```
.maior img {max-width:75vw; max-height:75vh; min-width:50vw; min-height:50vh; border-radius:10px;}
```

Assim que o visitante clicar na foto em miniatura da classe **mini**, o alvo (**target**) do link será a classe **maior**, que pode ser configurada com animação (como fizemos nas atividades anteriores). A classe **bloco** vai servir para alinharmos horizontal e verticalmente a imagem com os botões.

```
.maior:target {outline:none; display:block; opacity:1;}
.bloco {height:100vh; display:flex; align-items:center; justify-content:center;}
.maior img {animation:Efeito1 1.2s;}
.anterior {animation:Efeito2 1.2s;}
.proxima {animation:Efeito3 1.2s;}

@keyframes Efeito1 {
  0% {transform:scale(0.5); opacity:0;}
  100% {transform:scale(1); opacity:1;}
}
@keyframes Efeito2 {
  0% {transform:translate(500px,0); opacity:0;}
  100% {transform:translate(0,0); opacity:1;}
}
@keyframes Efeito3 {
  0% {transform:translate(-500px,0); opacity:0;}
  100% {transform:translate(0,0); opacity:1;}
}
```

Agora vamos configurar os botões. O botão **fechar** ficará fixo, no canto superior direito da tela. Os botões **anterior** e **proxima** ficarão alinhados com a imagem da galeria. Podemos definir cores de fonte e de fundo além das margens destes botões. Para que estes botões apareçam em cima da imagem, usamos **z-index:3**.

```
.botao {color:#fafafa; font-size:2.5em; background-color:rgba(50,50,50,0.4); padding:5px 20px 0 20px;
            border-radius:4px; text-decoration:none; cursor:pointer; z-index:3;}
.botao:hover {background-color:rgba(50,50,50,0.6);}
.fechar {position:absolute; right:2%; top:2%;}
.fechar:hover {background-color:rgba(50,50,50,0.6);}
```

Agora vamos configurar as miniaturas das imagens com classe **mini**. Para deixar um efeito de que as imagens estão sobrepostas, podemos criar margens laterais negativas e criar margens internas com **padding:7px**. Vamos usar o alinhamento centralizado para colocar o título (**title**) de cada imagem como legenda (como fazíamos com **figcaption** nas primeiras aulas).

```
.mini {margin:20px 10px; padding:7px; text-align:center; text-decoration:none; background:linear-
      gradient(-45deg, #FFF,#CCC); border-radius:10px; border:1px solid #999;}
```

Vamos configurar um mesmo tamanho para todas as imagens em miniatura, com **z-index:1** e efeito de transição para fazermos animação com **hover**.

```
.mini img {width:150px; height:100px; margin:10px 10px -5px 10px; transition:0.4s all ease-in-out; z-index:1;}
.mini img:hover {transform:scale(1.1);}
```

O recurso de pseudo-classe que usaremos para inserir os títulos é chamado de **before/after**, que pode ser usado para inserir elementos antes ou depois de algum elemento da página. Este elemento pode ser texto ou imagem. Este recurso é utilizado para inserir automaticamente imagens antes de cada item de menus.

Nos exemplos a seguir, antes de cada título **h1** da página uma imagem é inserida, e antes de cada parágrafo um hífen é inserido.

```
h1:before {content:url(imagens/smiley.gif);}
p:before {content:"- ";}
```

Em nossa galeria, vamos usar a pseudo-classe **after** para inserir os títulos de cada imagem em miniatura. Como cada imagem tem seu título, vamos usar a função **attr()** para buscar o título de cada imagem.

```
a.mini:after {content:attr(title); display:block; font-size:1.1em;}
```

Para criar um efeito para que as miniaturas das imagens fiquem com aspecto “desarrumado”, podemos usar a pseudo-classe **nth-child** (a mesma que usamos em atividades anteriores).

Podemos agrupar todas as minaturas no HTML, antes das classes de imagens ampliadas. Se você quiser apenas alternar 2 efeitos entre as miniaturas, basta usar números **2n+1;** para 3 efeitos, usamos **3n+1.** O exemplo que vamos usar tem 5 efeitos diferentes.

```
<!-- miniaturas das fotos -->
<a href="#img1" class="mini" title="Meu projeto 1">
    <img src="imagens/projeto1.jpg">
</a>
<a href="#img2" class="mini" title="Meu projeto 2">
    <img src="imagens/projeto2.png">
</a>
<a href="#img3" class="mini" title="Meu projeto 3">
    <img src="imagens/projeto3.jpg">
</a>
...
<!-- fotos ampliadas, escondidas pelo CSS -->
<div class="maior" id="img1">
   ...
</div>
<div class="maior" id="img2">
   ...
</div>
<div class="maior" id="img3">
   ...
</div>
...
```

	n=0	n=1	n=2	estilo
5n+1	1	6	11	azul, 0°
5n+2	2	7	12	preto, 5°
5n+3	3	8	13	amarelo, -5°
5n+4	4	9	14	vermelho, 5°
5n+5	5	10	15	verde, -10°

Logo, o estilo ficará assim:

```
a.mini:nth-child(5n+1) {color:blue; transform:rotate(0deg);}
a.mini:nth-child(5n+2) {color:black; transform:rotate(5deg);}
a.mini:nth-child(5n+3) {color:yellow; transform:rotate(-5deg);}
a.mini:nth-child(5n+4) {color:red; transform:rotate(5deg);}
a.mini:nth-child(5n+5) {color:green; transform:rotate(-10deg);}
```

Para criar os itens de compartilhar, podemos inserir imagens dentro do banner do site com link nas tags de SVG, indicando as coordenadas x e y, além das medidas de altura e largura de cada imagem. É a mesma ideia que usamos nos mapeamentos de imagens.

Para deixá-las alinhadas à direita, na base do banner, colocamos as coordenadas x próximas da medida da largura de 1200px e da altura de 140px.

```
<a href="http://www.facebook.com/pages/UFPR">
   <image xlink:href="imagens/facebook.png" x="860" y="140" height="50px" width="50px"/>
</a>
<a href="http://www.twitter.com/ufpr">
   <image xlink:href="imagens/twitter.png" x="930" y="140" height="50px" width="50px"/>
</a>
<a href="https://www.instagram.com/ufpr_oficial/">
   <image xlink:href="imagens/instagram.png" x="1000" y="140" height="50px" width="50px"/>
</a>
<a href="whatsapp://send?text=Meu portfolio - http://www.degraf.ufpr.br/docentes/paulo/webdesign/">
   <image xlink:href="imagens/whatsapp.png" x="1070" y="140" height="50px" width="50px"/>
</a>
<a href="mailto:seuemail@provedor.com.br">
   <image xlink:href="imagens/email.png" x="1130" y="140" height="50px" width="50px"/>
</a>
```

Podemos organizar a galeria em sub-galerias com a tag **details**. Basta inserir dentro de cada uma das tags as imagens em miniatura e ampliadas de cada sub-galeria. Se você quiser deixar a primeira já aberta, basta inserir o atributo **open** na primeira tag **details**. A tag **summary** contém o título de exibição da sub-galeria.

```
<details open><summary>Disciplina 1</summary>
    <div class="galeria">
       <!-- miniaturas das fotos -->
       <a href="#img1" class="mini" title="Meu projeto 1">
           <img src="imagens/projeto1.jpg">
       </a>
       <a href="#img2" class="mini" title="Meu projeto 2">
           <img src="imagens/projeto2.jpg">
       </a>
       ...
       <!-- fotos ampliadas, escondidas pelo CSS -->
       <div class="maior" id="img1">
           ...
       </div>
       <div class="maior" id="img2">
           ...
       </div>
       ...
     </div>
</details>
<details><summary>Disciplina 2</summary>
    <div class="galeria">
       ...
```

Podemos configurar as propriedades destas tags no CSS. Definimos atributos de fontes, margens, alinhamentos e bordas. O atributo cursor:pointer mostra ao visitante a possibilidade de clicar na imagem: é o mesmo efeito de quando passamos o cursor do mouse sobre um link.

```
summary {font-size:1.3em; color:#004953; font-weight:bold; cursor:pointer; margin-bottom:12px; text-
        shadow:1px 1px 1px skyblue; text-align:center;}
details {border:1px solid skyblue; border-bottom:none; padding:3px; border-radius:10px;}
```

Atividade 10: Entregar os arquivos do site estruturado de um portfólio com galeria de imagens, menu, desenhos de banner e/ou logos com animações em SVG. A galeria de imagens pode ser única ou separada com tags **details**.