

Basicos Tecnologias Web

[Nivan Ferreira](mailto:nivan@cin.ufpe.br)
nivan@cin.ufpe.br

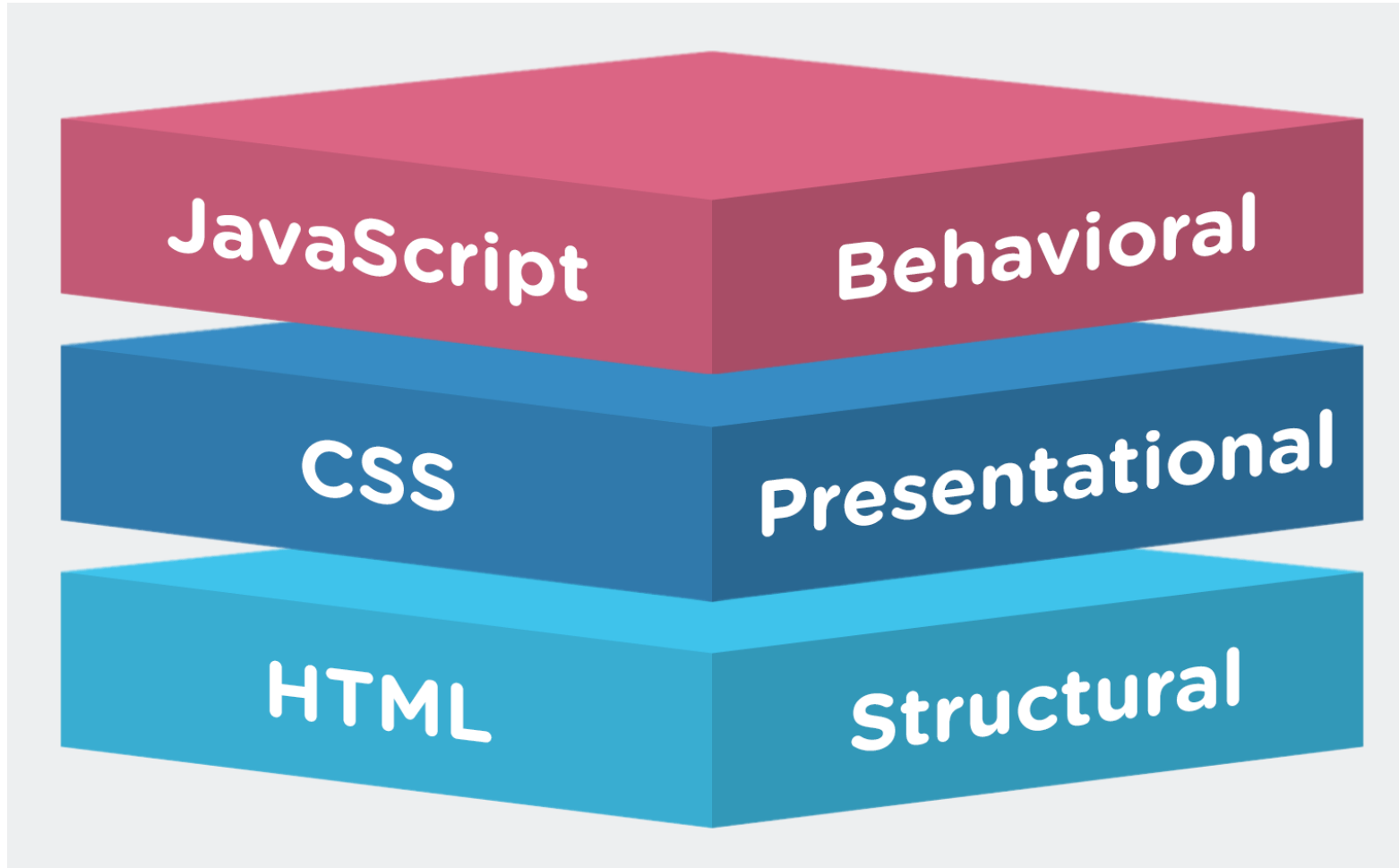


Revisão de Tecnologias Web

Tecnologias

- Como ferramentas para construir visualizações usaremos
 - HTML
 - CSS
 - JavaScript (D3)
- Fontes de Informação
 - [Mozilla Developer Network](#)

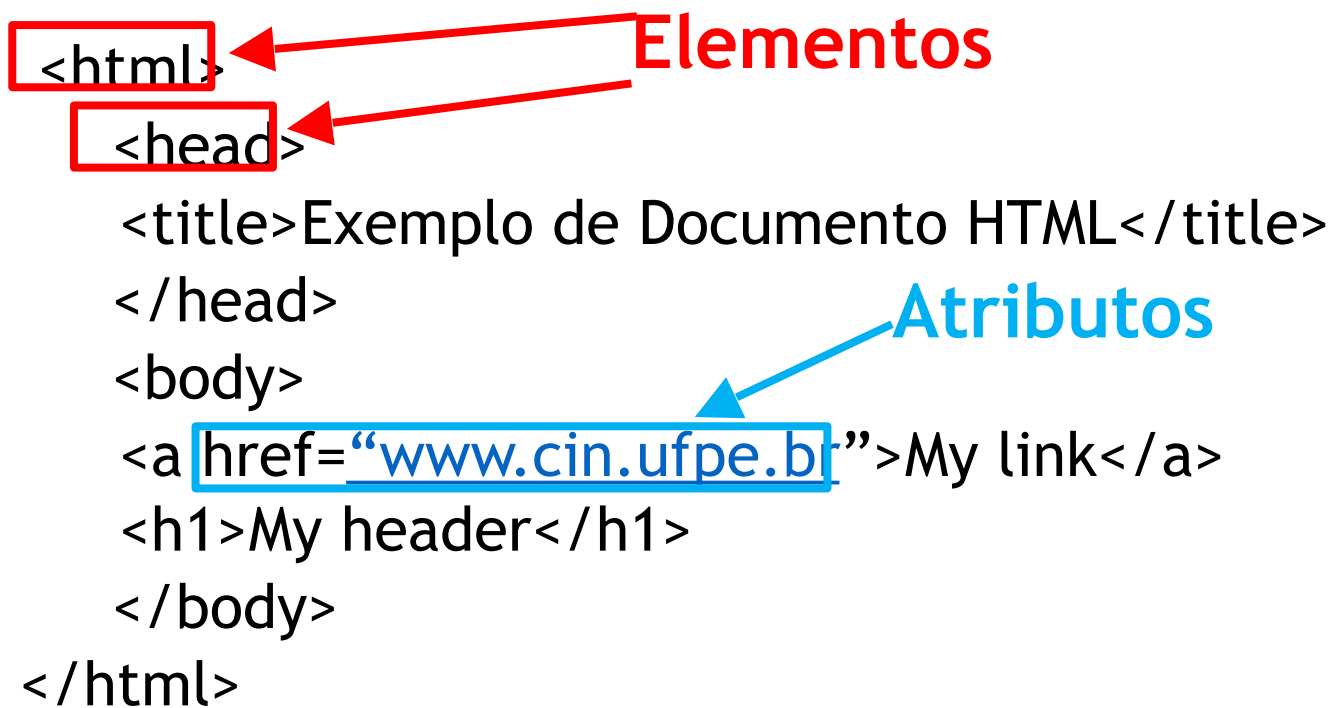
Tecnologias



HyperText Markup Language (HTML)

HTML

- Linguagem de Marcação



The diagram illustrates the structure of an HTML document. It shows a sequence of tags: `<html>`, `<head>`, `<title>`, `</head>`, `<body>`, ``, `My link`, `<h1>`, `My header</h1>`, `</body>`, and `</html>`. Two red arrows point from the word "Elementos" to the `<html>` and `<head>` tags, which are also enclosed in red boxes. A blue arrow points from the word "Atributos" to the `href="www.cin.ufpe.br"` attribute within the `<a>` tag, which is enclosed in a blue box.

Elementos

```
<html>  
  <head>  
    <title>Exemplo de Documento HTML</title>  
  </head>  
  <body>  
    <a href="www.cin.ufpe.br">My link</a>  
    <h1>My header</h1>  
  </body>  
</html>
```

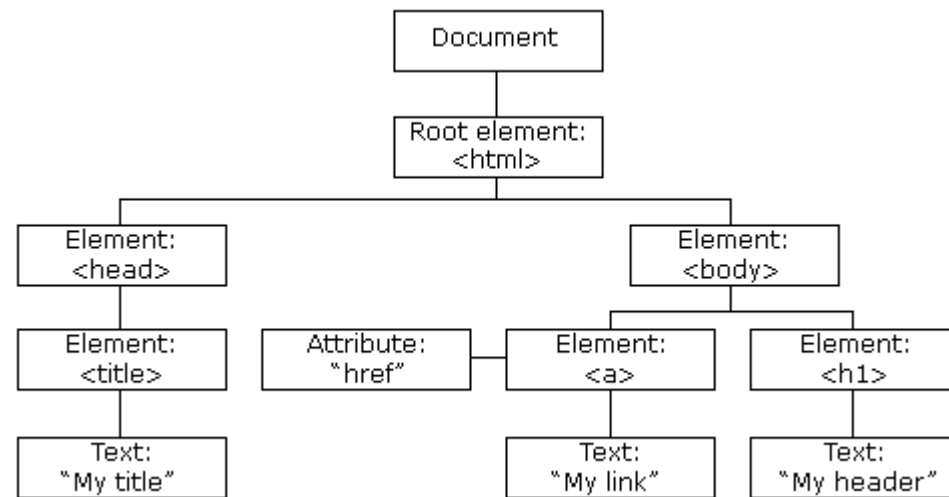
Atributos

HTML - Common Attributes

```
<html>
  <head>
    <title>Exemplo de Documento HTML</title>
  </head>
  <body>
    <a id="cinLink" href="www.cin.ufpe.br" >My link</a>
    <h1 class="myHeader">Faculty</h1>
    <h1 class="myHeader">Research</h1>
  </body>
</html>
```

DOM

- Interface de programação que permite representar documentos HTML como uma árvore
- Provê métodos para o acesso, navegação e edição desta árvore



HTML Boilerplate

```
<html>  
  <head>  
    <title>Title</title>  
  </head>  
  <body>  
  </body>  
</html>
```

Cascading Style Sheets (CSS)

CSS



- Linguagem usada para descrever a apresentação visual documentos HTML
- Funciona através de regras que selecionam elementos e associam a eles características de aparência

Seletor → `p` {
 `color: red;` ← **propriedade**
}

CSS - Declaração Externa

index.html

```
<html>
  <head>
    <title>My CSS experiment</title>
    <link rel="stylesheet"
href="style.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

style.css

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```


CSS - Declaração Interna

index.html

```
<html>
  <head>
    <title>My CSS experiment</title>
    <style>
      *****
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```



CSS - Seletores

- Seletor de elemento: Elemento {...}
- Exemplo, queremos ter parágrafos vermelhos

```
p {  
    color: red;  
}
```

CSS - Seletores

- Seletor de classe: `.classe {...}`
 - Exemplo:

```
.paragrafo {  
    color: red;  
}
```
- Seletor de id: `#meuid {...}`
 - Exemplo:

```
#paragrafo {  
    color: red;  
}
```

CSS - Seletores de Relação

- Seletor de Filho:

- elementoPai > elementoFilho {

- ...

};

- Seletor Descendente

- elemento elementoDescendente {

- ...

};

Scalable Vector Graphics (SVG)

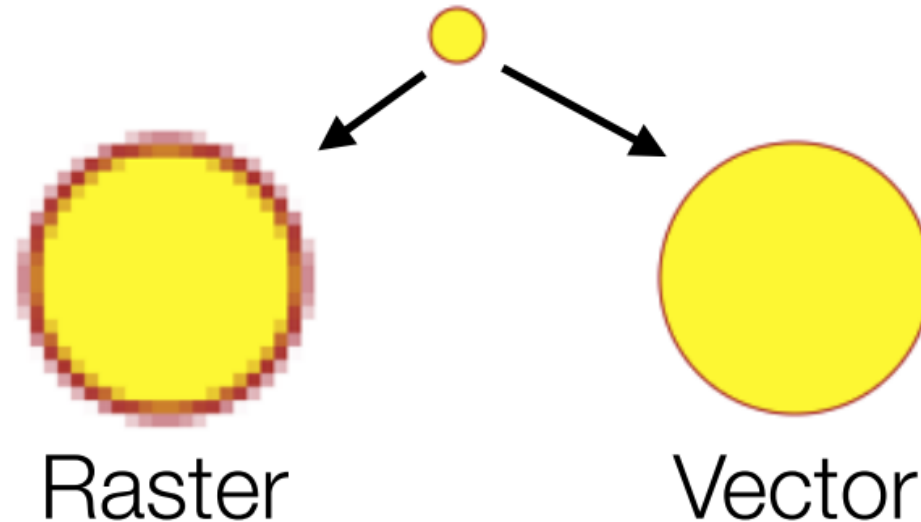
Scalable Vector Graphics (SVG)

- Uma linguagem de marcação:
 - Descreve formas, pontos, cores, espessura...
- Elementos SVG podem ser incluídos em documentos HTML5!

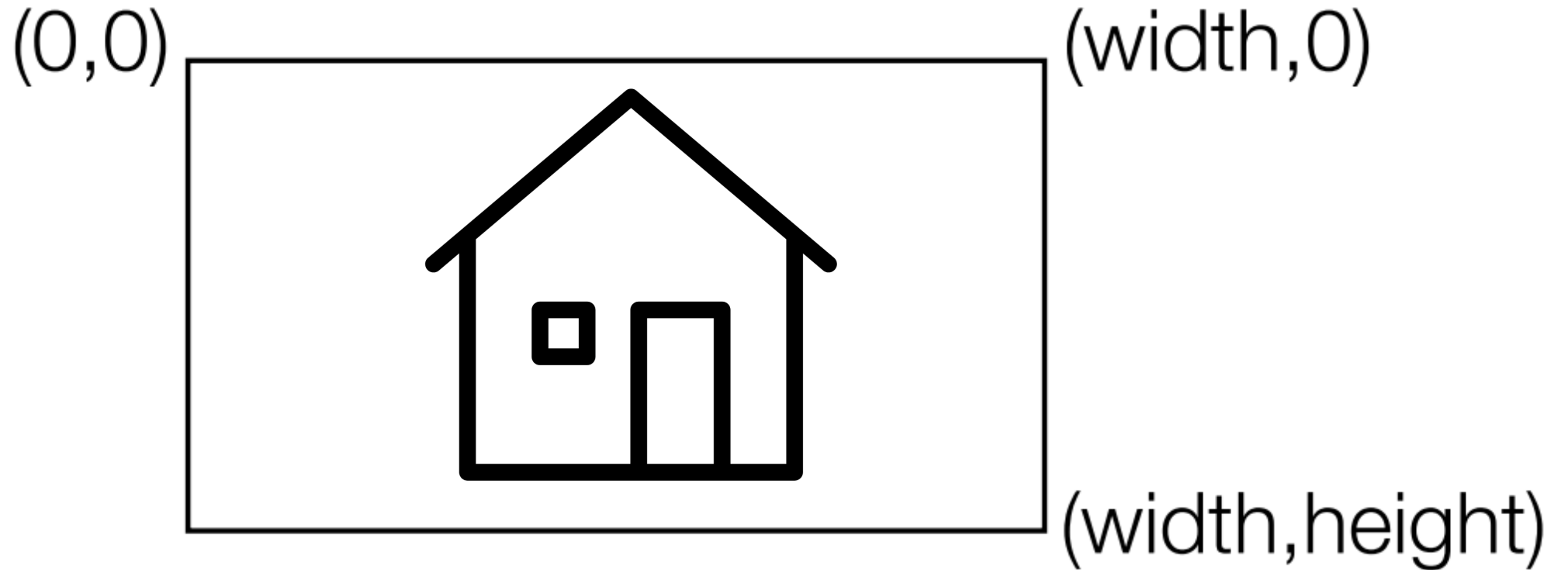
```
<svg width="400" height="400"> </svg>
```

Scalable Vector Graphics (SVG)

- Gráficos de Rasterização vs. Gráficos Vetoriais
- Grid de pixels vs. Comandos de desenho
- Por que usar gráficos vetoriais?

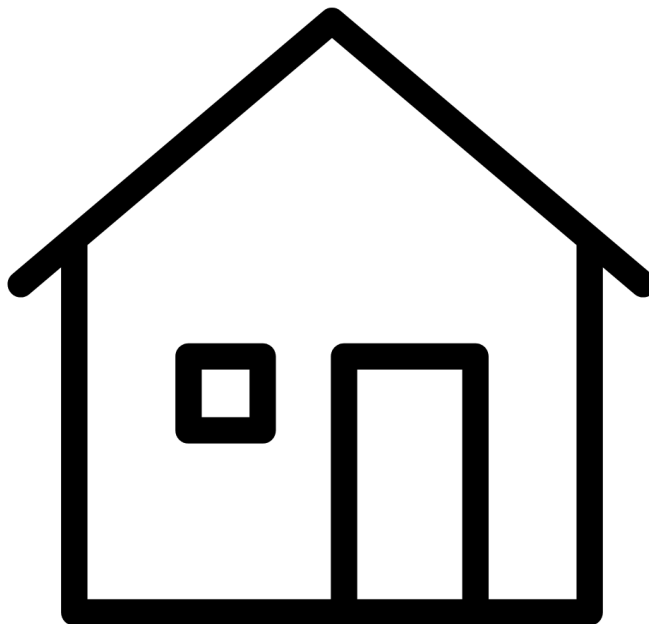


SVG Sistema de Coordenadas



Scalable Vector Graphics (SVG)

- Primitivas de Desenho:
 - Linhas, círculos, retângulos, elípses, textos, linhas poligonais, caminhos
- O princípio é descrever uma imagem através de informações sobre primitivas de desenho



Círculo

```
<svg width="400" height="200">
```

```
  <circle
```

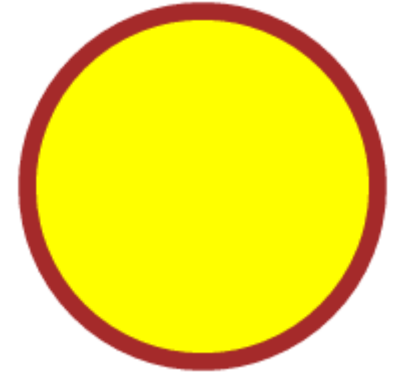
```
    cx="100"
```

```
    cy="100"
```

```
    r="50"
```

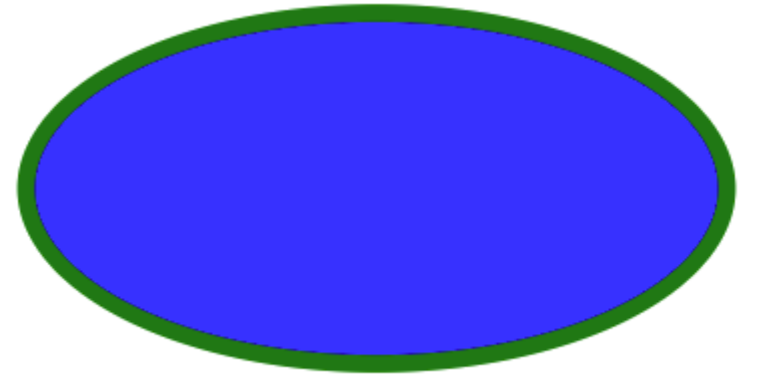
```
    style="fill:yellow; stroke:brown; stroke-width:5px"/>
```

```
</svg>
```



Ellipse

```
<svg width="400" height="200">  
  <ellipse cx="200"  
    cy="100"  
    rx="100"  
    ry="50"  
    style="fill:blue; stroke:green; stroke-width:5px"/>  
</svg>
```



Retângulo

```
<svg width="400" height="200">
```

```
  <rect
```

```
    x="160"
```

```
    y="50"
```

```
    width="200"
```

```
    height="100"
```

```
    style="fill:red; stroke:black; stroke-width:3px"/>
```

```
</svg>
```



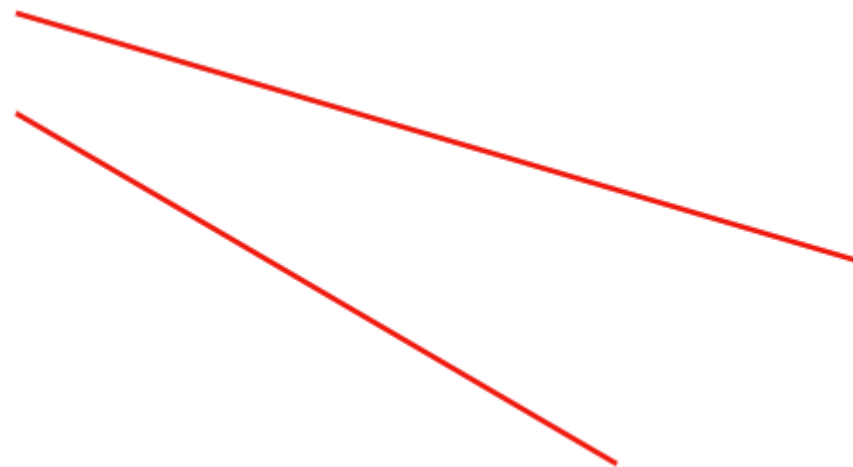
Linhas

```
<svg width="400" height="200">
```

```
  <line x1="30" y1="30" x2="200" y2="80" style="stroke:red"/>
```

```
  <line x1="30" y1="50" x2="150" y2="120" style="stroke:red"/>
```

```
</svg>
```



Texto

```
<svg width="400" height="100">  
  <text x="30" y="30">Some text</text>  
  <text x="30" y="50">Some more text</text>  
</svg>
```

Some text
Some more text

Caminhos

```
<svg width="200" height="60" style="stroke:blue; fill:none">  
  <path d="M 10 10 L 50 10 L 50 50 L 100 50 L 100 10 C 150 50  
150 50 150 10"/>  
</svg>
```



Ordenação

- O que acontece quando dois elementos se interceptam?

```
<svg width="400" height="200">
```

```
<ellipse cx="200" cy="100" rx="100" ry="50" style="fill:blue;  
stroke:green; stroke-width:5px"/>
```

```
<rect x="50" y="50" width="200" height="100" style="fill:red;  
stroke:black; stroke-width:3px"/>
```

```
</svg>
```



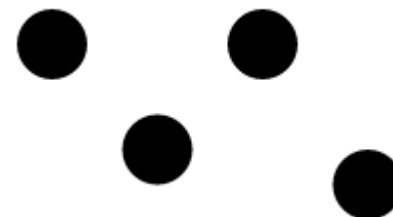
Grupos em SVG

- Mecanismo para organização de elementos svg
- Servem para aplicar operações em múltiplos elementos



Grupos em SVG

```
<svg width="200" height="200">  
  <g>  
    <circle cx="50" cy="50" r="10"/>  
    <circle cx="80" cy="80" r="10"/>  
    <circle cx="110" cy="50" r="10"/>  
    <circle cx="140" cy="90" r="10"/>  
  </g>  
</svg>
```



Transformações em SVG

Translation

- mover elemento pro ponto *translate(x,y)*

Rotation

- rotacionar um elemento em *rotate(graus)*

Scaling

- muda o tamanho de um elemento em *scale(x,y)*

Grupos em SVG

```
<svg width="200" height="200">
```

```
  <g transform="translate(0, 200) scale(1, -1)">
```

```
    <circle cx="50" cy="50" r="10"/>
```

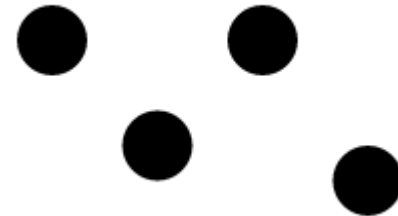
```
    <circle cx="80" cy="80" r="10"/>
```

```
    <circle cx="110" cy="50" r="10"/>
```

```
    <circle cx="140" cy="90" r="10"/>
```

```
  </g>
```

```
</svg>
```



SVG - Grupos

```
<svg width="200" height="200">
```

```
  <g transform="translate(200,0) scale(-1, 1)">
```

```
    <text x="10" y="50">Text</text>
```

```
    <text x="70" y="50">as they read</text>
```

```
    <text x="40" y="80">in</text>
```

```
    <text x="100" y="90">Australia</text>
```

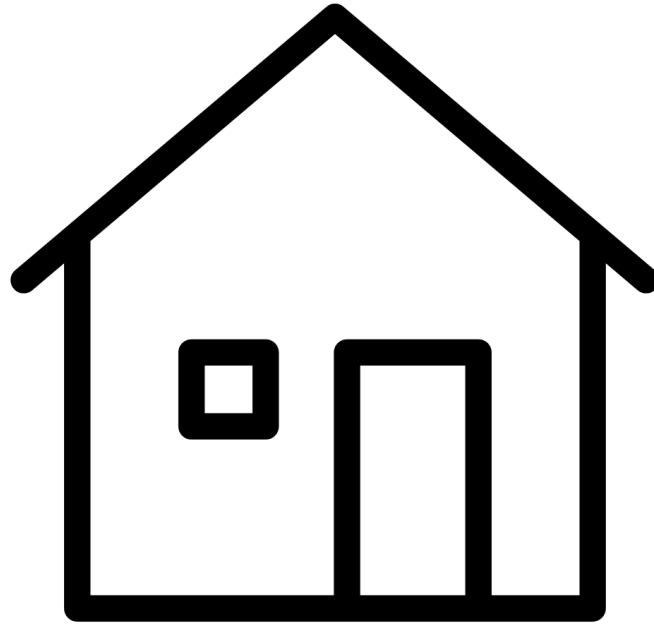
```
  </g>
```

```
</svg>
```

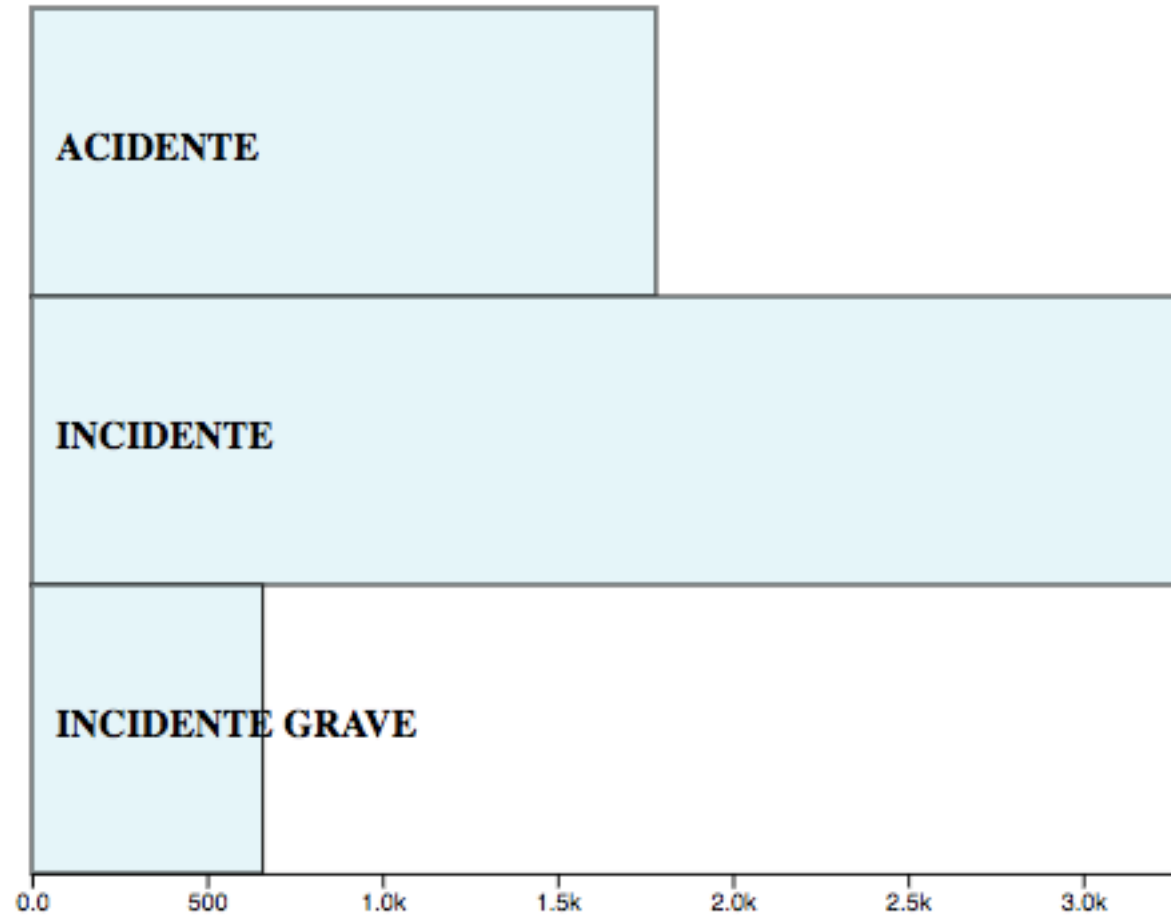
Text
as they read
in
Australia

Exercícios SVG

- Como você desenharia esta figura?



Como você desenharia esta figura?



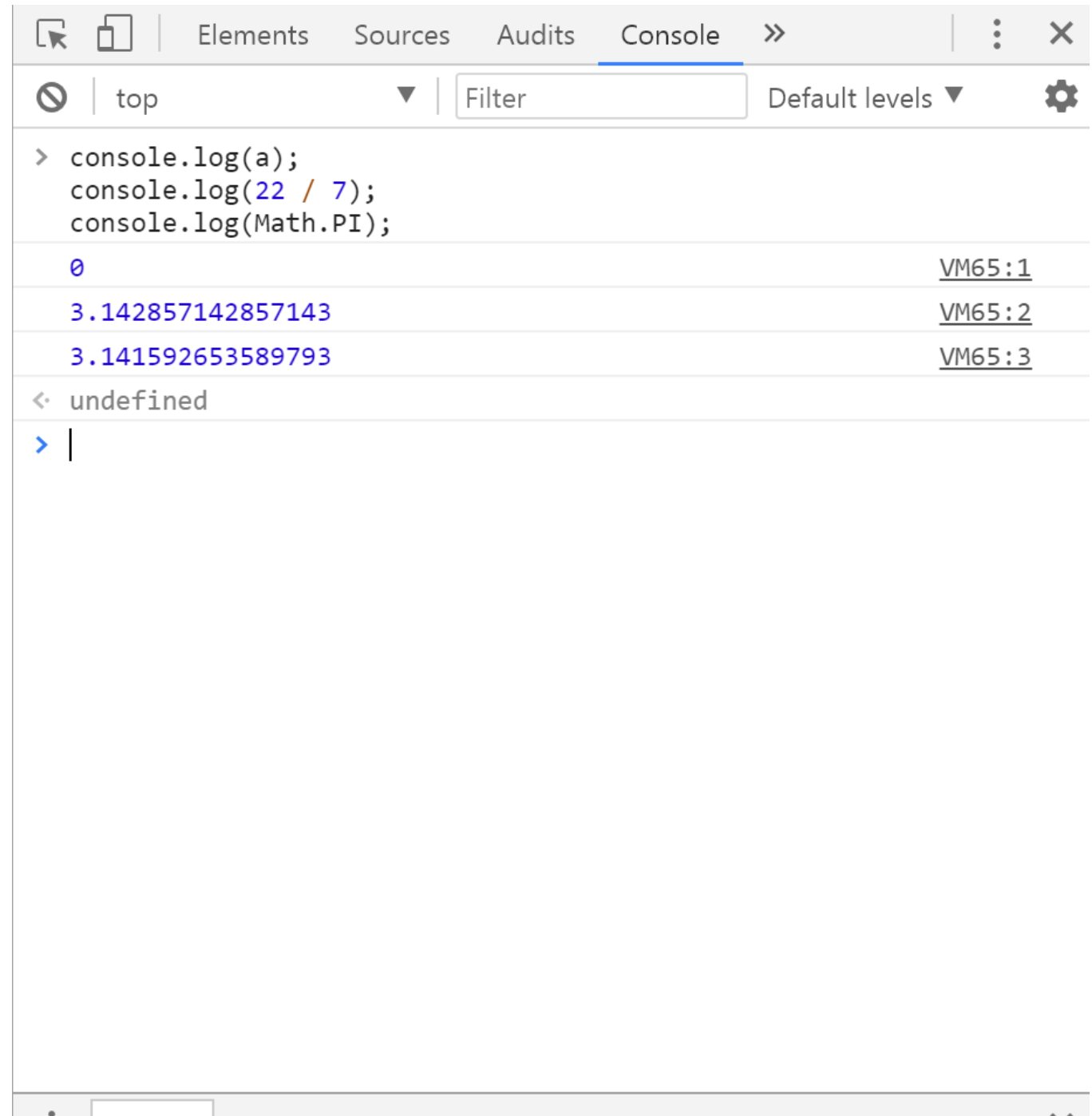
Javascript

Variáveis

- `a = 0;`
 - `b = "1";`
 - `c = [1, 2, "3", [4]];`
 - `f = false;`
 - `f = 34.56;`
 - `g=undefined`
-
- `var x = 0; //local variable`

Console

- `console.log(a);`
- `console.log(22 / 7);`
- `console.log(Math.PI);`



Tipos Compostos

- Arrays
 - `var c = [0,1,2];`
 - `var e = [];` // empty array declaration
 - `console.log(c[0]);`
- Objetos
- `obj = {
 key1: 3
};`
`console.log(obj["key1"]);`
`console.log(obj.key1);`
`obj.key3 = "new value";`

Funções

```
• function someFunction(v) {  
    if (v < 10) {  
        return v;  
    } else {  
        return v*v;  
    }  
}
```

```
console.log(someFunction(30));  
console.log(someFunction(-5));
```

```
console.log(someFunction("50"));  
console.log(someFunction("what?"));  
console.log(someFunction(30, "huh?"));  
console.log(someFunction(30, "huh?"));  
console.log(someFunction());
```


Funções Anônimas

- `someOtherFunction = function(v) {
 if (v > 10) {
 return "big";
 }
 else {
 return "small";
 }
};`

Classes

- ```
class Rectangle {
 constructor(height, width) {
 this.height = height;
 this.width = width;
 }
 // Method
 calcArea() { return this.height * this.width; }
}
var square = new Rectangle(10, 10);
console.log(square.calcArea()); // 100
```