argonavis.com.br

SVG 8

eventos
javascript
svg dom

Helder da Rocha (helder@argonavis.com.br)

Eventos

- SVG pode responder a eventos iniciados pelo usuário através de recursos da linguagem
  - Movimento do mouse ou cliques podem disparar animações, scripts ou realização de hiperlinks, zoom e rolagem, mudanças do cursor, etc.
- Aspectos de SVG afetados por eventos
  - Ouvintes de eventos do DOM (**DOM listeners**) registrados por scripts, e que chamam outro script quando executados
  - **Atributos de eventos** em elementos, que indicam script a executar quando o evento acontecer
  - **Elementos de animação** podem começar ou parar uma animação com base em eventos
- Atributos permitem ligar, desligar ou controlar a forma como eventos são tratados por elementos
  - **pointer-events** – controla aspectos de cliques, movimentos
  - **zoomAndPan** – controla interatividade de zoom
  - **cursor** – controla a aparência do cursor (também existe um elemento <cursor> que permite definir um cursor específico)

Alguns atributos de evento

xxx *=> evento*
*on**xxx** => atributo*

- on**focusin** e on**focusout**
  - Chamados quando elemento recebe ou perde foco (quando um texto é selecionado, por exemplo)
- on**activate**
  - Chamado quando um elemento é ativado através de um clique ou tecla
- on**click**, on**mouseup**, on**mousedown**
  - Chamados quando um mouse é clicado sobre o elemento
- on**mouseover**, on**mousemove**, on**mouseout**
  - Chamados de acordo com o movimento do mouse sobre um elemento
- on**load**, on**unload**, on**error**
  - Eventos relacionados à carga do objeto na memória.
- on**resize**, on**scroll**, on**zoom**
  - Chamados quando usuário redimensiona a tela, rola, usa zoom
- on**begin**, on**end**, on**repeat**
  - Eventos relacionados ao comportamento de animação

Scripting

- Para especificar a linguagem usada
  - Atributo **contentScriptType** de <svg> (default **application/ecmascript**) ou atributo **type** do elemento **<script>** (default **text/ecmascript**)
  - Se o cliente nao suportar a linguagem, nenhum bloco script é executado
- O elemento **<script>**
  - Aceita declaração local de código **<script>...</script>**
  - Aceita arquivo externo via **<script xlink:href="arquivo.js" />**

SVG Document Object Model

- Permite usar scripts para controlar componentes SVG
  - *http://www.w3.org/TR/SVG/svgdom.html*
- SVG DOM estende XML DOM
  - Interfaces do SVG DOM herdam do XML DOM
  - Pode-se fazer quase tudo conhecendo-se apenas o XML DOM
  - **SVGElement** herda de **Element** (que herda de **Node**)
  - **SVGDocument** herda de **Document** (que também é **Node**)
- Objetos especiais:
  - **`document`** (ponteiro para **/**)
  - **`evt`** (ponteiro para evento)
- Ponteiro para qualquer elemento pelo ID:
  - `document.getElementById("id");`
- Elemento que foi fonte de um evento:
  - `var elemento = evt.target;`

SVG DOM

- Possui interfaces específicas para cada objeto do SVG
  - SVGElement, SVGSVGElement, SVGRectElement, SVGImageElement, SVGPathElement, etc.
  - SVGStyleElement e propriedade **style** (da W3C spec DOM 2 CSS)
  - Propriedades **click**, **mouseover**, etc. (W3C spec DOM 2 Events)
  - SVGColor, SVGPaint, SVGTransform, etc.
- A maior parte apenas expõe propriedades dos elementos
  - Outros como SVGColor, SVGTransform e similares têm métodos e propriedades que facilitam a manipulação
- Exemplo: SVGStyleElement (igual a **style** do HTML DOM)
  - Para obter, use SVGElement.**getStyle()** ou **style**

```
document.getElementById("obj").style.fill="yellow"
```

Exemplos de uso do SVG/XML DOM

- document.**getElementById**(id)

```
var objeto = document.getElementById("seta");
```

- document.**documentElement** ou document.**rootElement**

```
var filhos = document.documentElement.childNodes.length;

funcion init(evt) {
     var svg = evt.target.ownerDocument.documentRoot;
}
```

- SVGEvent.target é um SVGElement:

```
function printParent(evt) {
     alert(evt.target.parentNode.nodeName);
}
```

- Para manipular atributos

```
evt.target.getAttribute("height");

evt.target.setAttribute("fill", "red");
```

- Criar elemento e adicioná-lo como filho de <svg>

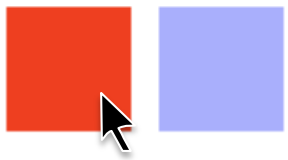
```
var rect = document.createElement("rect");
rect.setAttribute("width", 100);
document.rootElement.appendChild(rect);
```

Exemplo simples com JavaScript

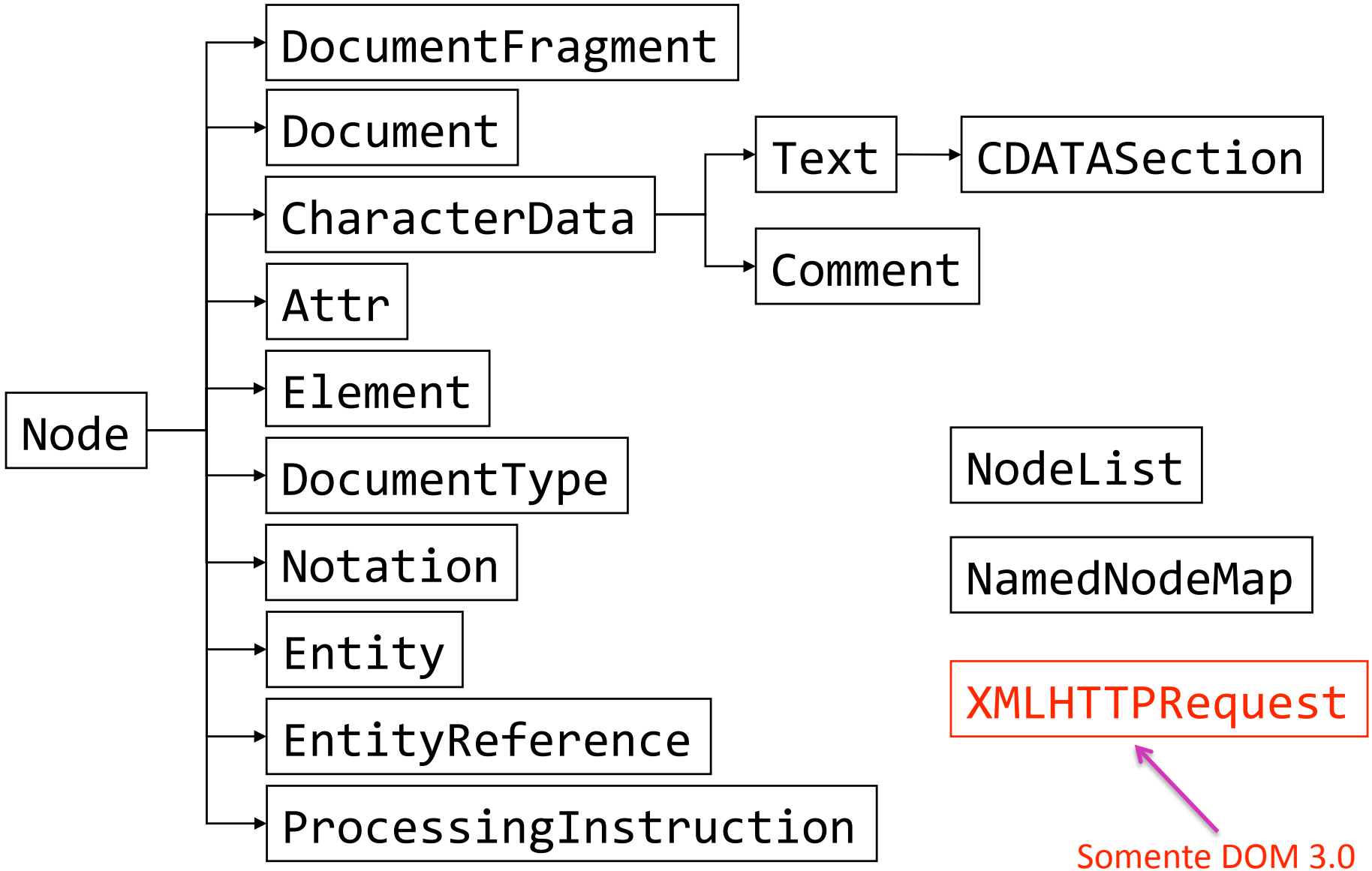
```
<svg width="100%" height="100%" xmlns="http://www.w3.org/2000/svg">

  <script type="text/ecmascript">
  <![CDATA[
    function acende(evt) {
       evt.target.setAttribute("opacity", "1.0");
    }
    function apaga(evt) {
       evt.target.setAttribute("opacity", "0.4");
    }
  ]]>
  </script>

    <g>
        <rect x="10"  y="100" width="45" height="45" fill="red" opacity="0.4"
              onmouseover="acende(evt);" onmouseout="apaga(evt);"/>
        <rect x="65" y="100" width="45" height="45" fill="blue" opacity="0.4"
              onmouseover="acende(evt);" onmouseout="apaga(evt);"/>
...
    </g>
</svg>
```



XML DOM 2: Principais elementos



Alguns métodos da interface Node

- Node **appendChild**(Node)
- Node **cloneNode**(boolean)
- NamedNodeMap **getAttributes**() — **attributes**
- NodeList **getChildNodes**() — **childNodes**
- boolean **hasAttributes**()
- boolean **hasChildNodes**()
- Node **insertBefore**(Node, Node)
- Node **removeChild**(Node)
- Node **replaceChild**(Node, Node)
- Node **getFirstChild**() — **firstChild**
- Node **getLastChild**() — **lastChild**
- Node **getNextSibling**() — **nextSibling**
- Node **getPreviousSibling**() — **previousSibling**
- String **getNodeName**() — **nodeName**
- short **getNodeType**() — **nodeType**
- String **getNodeValue**() — **nodeValue**
- Document **getOwnerDocument**() — **ownerDocument**
- Node **getParentNode**() — **parentNode**

DOM: tipos de nó

- DOM usa constantes para identificar tipos de nó (nodeType)

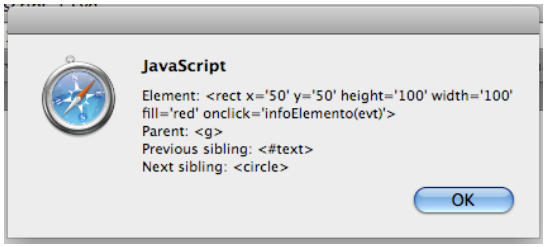
Constante (opcional)	Tipo	valor
ELEMENT_NODE	Element	1
ATTRIBUTE_NODE	Attr	2
TEXT_NODE	Text	3
CDATA_SECTION_NODE	CDATASection	4
ENTITY_REFERENCE_NODE	EntityReference	5
ENTITY_NODE	Entity	6
PROCESSING_INSTRUCTION_NODE	ProcessingInstruction	7
COMMENT_NODE	Comment	8
DOCUMENT_NODE	Document	9
DOCUMENT_TYPE_NODE	DocumentType	10
DOCUMENT_FRAGMENT_NODE	DocumentFragment	11
NOTATION_NODE	Notation	12

Métodos para listas e mapas

- NamedNodeMap
  - Node **item**(int)
  - Node **getNamedItem**(String)
  - Node **nextNode**()
  - void **reset**()
  - int **getLength**() **length**
- NodeList
  - Node **item**(int)
  - Node **nextNode**()
  - void **reset**()
  - int **getLength**() **length**

Exemplo com Node

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%" onload="numFilhos()">
   <script><![CDATA[
     function numFilhos() {
        var data = "Elementos contidos em <svg>: "+document.documentElement.getChildNodes().length;
        alert(data);
     }
     function infoElemento(evt) {
          var atributos = evt.target.attributes;
          var data = "Element: <" + evt.target.nodeName;
          for (i = 0; i < atributos.length; i++) {
             data += " " + atributos.item(i).name + "='" + atributos.item(i).value + "'";
          }
          data += ">\nParent: <" + evt.target.parentNode.nodeName + ">";
          data += "\nPrevious sibling: <" + evt.target.previousSibling.nodeName + ">";
          data += "\nNext sibling: <" + evt.target.nextSibling.nodeName + ">";
          alert(data);
     }
   ]]></script>
    <g>
        <rect x="50" y="50" height="100" width="100" fill="red" onclick="infoElemento(evt)" />
<circle cx="225" cy="100" r="50" fill="blue" onclick="infoElemento(evt)"/><g font-size="10pt">
           <text x="300" y="50">Ubi dubium ibi libertas</text>
        </g>
    </g>
</svg>
```



Interface Element

- String **getAttribute**(String)
- String **getAttributeNS**(String, String)
- Attr **getAttributeNode**(String)
- Attr **getAttributeNodeNS**(String, String)
- NodeList **getElementsByTagName**(String)
- NodeList **getElementsByTagNameNS**(String, String)
- String **getTagName**() **tagName**
- boolean **hasAttribute**(String)
- boolean **hasAttributeNS**(String, String)
- void **removeAttribute**(String)
- void **removeAttributeNS**(String, String)
- void **setAttribute**(String, String)
- void **setAttributeNS**(String, String, String)

Interfaces Attr e Text

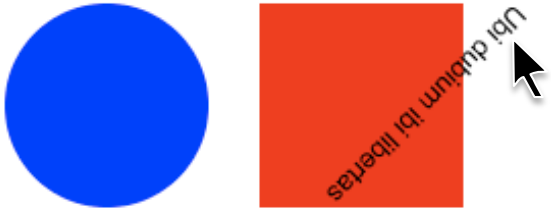
- Attr
  - String **getName**() **name**
  - Element **getOwnerElement**() **ownerElement**
  - String **getValue**() **value**
  - void **setValue**(String)
- Text e CharacterData
  - void **appendData**(String)
  - String **getData**() **data**
  - int **getLength**() **length**
  - void **insertData**(int, String)
  - void **replaceData**(int, int, String)
  - void **setData**(String)

Exemplo: elementos e atributos

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
   <script><![CDATA[
       function trocaLugar() {
            var rect = document.getElementsByTagName("rect").item(0);
            var circle = document.getElementsByTagName("circle").item(0);
            if (rect.getAttribute("x") == 50) {
                rect.setAttribute("x", "175");
                circle.setAttribute("cx", "100");
            } else {
                rect.setAttribute("x", "50");
                circle.setAttribute("cx", "225");
            }
       }

       var angulo = 0;
       function gira(evt) {
           if (angulo <= 360) { angulo += 45;
           } else { angulo = 0; }
           evt.target.setAttribute("transform", "rotate("+angulo+", 300, 50)");
       }
   ]]></script>

    <g>
        <rect x="50" y="50" height="100" width="100" fill="red" onclick="trocaLugar()" />
        <circle cx="225" cy="100" r="50" fill="blue"/>
        <g font-size="10pt" onclick="gira(evt)">
            <text x="300" y="50">Ubi dubium ibi libertas</text>
        </g>
    </g>
</svg>
```



Separando scripts do SVG

- Boa prática: faça isto sempre que possível
  - Permite reuso das funções por outras aplicações (ex: SVG, HTML5, XHTML)
  - Facilita depuração do JS, simplifica geração de SVG, aumenta a legibilidade

script2.js

```
/* funcoes usadas em exemplo2.svg */

var angulo = 0;
function gira(evt) {
   if (angulo <= 360) {
      angulo += 45;
   } else {
      angulo = 0;
   }
   evt.target.setAttribute("transform", "rotate("+angulo+", 300, 50)");
}

function trocaLugar() {
    var rect = document.getElementsByTagName("rect").item(0);
    var circle = document.getElementsByTagName("circle").item(0);
    if (rect.getAttribute("x") == 50) {
        rect.setAttribute("x", "175");
        circle.setAttribute("cx", "100");
    } else {
        rect.setAttribute("x", "50");
        circle.setAttribute("cx", "225");
    }
}
```

```
<svg xmlns="http://www.w3.org/2000/svg">

  <script type="text/ecmascript" xlink:href="script2.js" />

  <g>
    <rect x="50" y="50" height="100" width="100" fill="red"
onclick="trocaLugar()" /> ...
    <g font-size="10pt" onclick="gira(evt)">
      <text x="300" y="50">Ubi dubium ibi libertas</text></g>
  </g>
</svg>
```

exemplo2.svg

DOM 2.0 com namespaces

- Use métodos que levam em conta o **namespace**
  - É necessário para acessar elementos e atributos que usam namespaces (ex: xlink)
  - É necessário quando se usa SVG com namespaces (ex: quando usado junto com XHTML, XSL-FO, etc.)
- Em vez de getAttribute, getElement, etc.
  - Use getAttributeNS, getElementNS, etc.

```
var svgNS = "http://www.w3.org/2000/svg";
var xlinkNS = "http://www.w3.org/1999/xlink";

var circle = document.createElementNS(svgNS,"circle");
circle.setAttributeNS(null,"cx",500);
circle.setAttributeNS(null,"cy",500);
circle.setAttributeNS(xlinkNS, "href", "http://www.a/com");
```

Interface Document

- Attr **createAttribute**(String)
- Attr **createAttributeNS**(String, String)
- Element **createElement**(String)
- Element **createElementNS**(String, String)
- Text **createTextNode**(String)
- DocumentType **getDocType**() **docType**
- Element **getDocumentElement**() **documentElement**
- Element **getDocumentById**(String)
- NodeList **getElementsByTagName**(String)
- NodeList **getElementsByTagNameNS**(String, String)

Exemplo: criando elementos

script_3.js

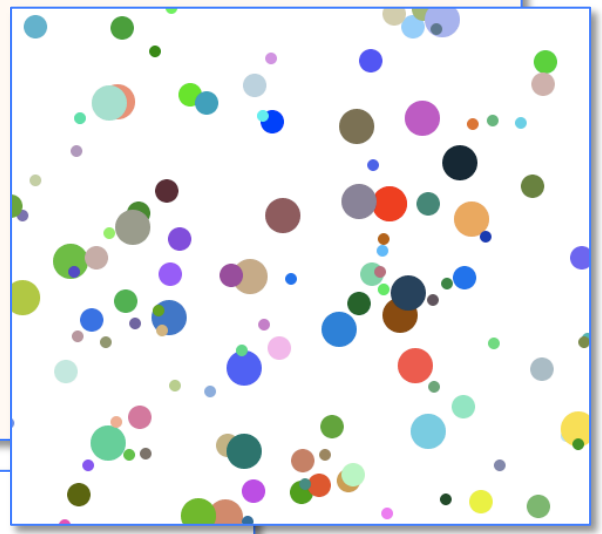
```
var svgNS = "http://www.w3.org/2000/svg";

function criarCirculo(evt) {
   var randomX =
       Math.floor( Math.random() * document.rootElement.getAttributeNS(null, "width")   );
   var randomY =
       Math.floor( Math.random() * document.rootElement.getAttributeNS(null, "height")  );
   var color = "rgb(" + Math.floor(Math.random() * 256) +", "+
                        Math.floor(Math.random() * 256) +", "+
                        Math.floor(Math.random() * 256) +")";

   var circulo = document.createElementNS(svgNS, "circle");
   circulo.setAttributeNS(null, "cx", randomX);
   circulo.setAttributeNS(null, "cy", randomY);
   circulo.setAttributeNS(null, "fill", color);
   circulo.setAttributeNS(null, "r",
                          evt.target.getAttributeNS(null, "r"));

   circulo.addEventListener("click", criarCirculo, false);

   evt.target.parentNode.appendChild(circulo);
}
```



```
<svg xmlns="http://www.w3.org/2000/svg" width="500" height="500">
   <script type="text/ecmascript" xlink:href="script_3.js" />
   <circle cx="125" cy="40" r="5"   fill="green" onclick="criarCirculo(evt)"/>
   <circle cx="225" cy="100" r="10" fill="blue" onclick="criarCirculo(evt)"/>
   <circle cx="325" cy="170" r="15" fill="red"  onclick="criarCirculo(evt)"/>
</svg>
```

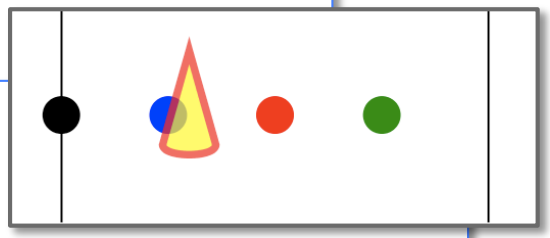
Animação usando SVG DOM

```
<svg width="500px" height="500px"  viewBox="0 0 500 500"
     xmlns="http://www.w3.org/2000/svg" version="1.1"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  <script type="text/ecmascript" xlink:href="script_5.js"/>

  <line x1="50"  y1="0" x2="50"  y2="200" stroke-width="2" stroke="black" />
  <line x1="450" y1="0" x2="450" y2="200" stroke-width="2" stroke="black" />
  <circle cx="50"  cy="100" r="17.64" fill="black"></circle>
  <circle cx="150" cy="100" r="17.64" fill="blue" ></circle>
  <circle cx="250" cy="100" r="17.64" fill="red"  ></circle>
  <circle cx="350" cy="100" r="17.64" fill="green"></circle>

  <g id="barquinho" transform="translate(50,140)" onclick="iniciarMovimento(evt)">
      <path d="M-25,-12.5 C-25,0 25,0 25,-12.5 L 0,-100.5 z"
            fill="yellow" stroke="red" stroke-width="7.06" >
      </path>
  </g>
</svg>
```



Use funções recursivas com **setTimeout()** para animações simples

```
var x   = 0.0;
var fim = 400.0;
var barquinho;

function iniciarMovimento(evt) { barquinho = evt.target;   mover(); }
function mover() {
    if (x >= fim) { return; }
    x = x + 1;
    barquinho.setAttributeNS(null, 'transform', "translate(" + x + ",0)");
    if (x > 50) {
        barquinho.setAttributeNS(null, 'opacity', (fim - x) / fim);
    }
    setTimeout("mover()", 30);
}
```

script_5.js