Instituto Federal Goiano - Campus Ceres
Bacharelado em Sistemas de Informação
Prof. Me. Ronneesley Moura Teles

Gusttavo Nunes Gomes
Ianka Talita Bastos de Assis

HTML5 Canvas e Javascript

Outubro
2017

Sumário

HTML5 Canvas e Javascript

1 Principios Fundamentais

A sociedade obtêm extremo interesse em métodos que visam ampliar tanto visual, físico e intelectualmente as diversas ferramentas existentes no âmbito tecnológico. O surgimento da web fez-se necessário para que a partir deste, pudéssemos engrandecer as linguagens que são entendidas por vias de acessos variados. Com isso, o nobre *Tim Berners-Lee* desenvolveu incrivelmente o HTML, tendo em vista a otimização deste meio. A popularização do HTML por sua vez, tornou-se devastadoramente desconhecido até meados dos anos 90, onde tomou enorme precisão sendo utilizado por diversos desenvolvedores e alguns fabricantes de browsers que buscavam compartilhar convicções e conhecimentos. O HTML passou a ganhar força quando o browser desenvolvido por Marc Andreessen intitulado Mosaic expandiu-se nos mercados. A decisão de desenvolver o HTML5 partiu de algumas empresas que buscavam o aperfeiçoamento do HTML 4.01 e do XHTML 2.0, sendo elas: *Web Hypertext Application Technology Working Group* (WHATWG) e *World Wide Web Consortium* (W3C) onde trabalharam juntas no HTML5. O projeto HTML5 foi devidamente recebido pelos desenvolvedores Web tornando-se assunto deliberadamente argumentado nas mídias em 2010, quando a CEO da *Apple Inc.*, Steve Jobs declarou em carta pública o seguinte tema: "*Reflexões sobre o Adobe Flash*". Steve Jobs concluiu que o desenvolvimento do projeto HTML5 faria com que a utilização do *Adobe Flash* fosse banida, não sendo tão necessário assim em questões de vídeos ou até mesmo em exibições de quaisquer conteúdos relacionados a web. Muitos que utilizam o HTML5 batem na tecla da proporcionalização de melhores funcionalidades e a variedade do browser em relação a demonstração de páginas diferentes fazendo com que não houvesse um padrão a se seguir. Logo, em novembro de 2011 a Adobe veio anunciar que romperia o desenvolvimento de Flash para dispositivos móveis e sim, redirecionar seus conhecimentos e esforços para desenvolver ferramentas que utilizassem o HTML5.

2 HTML5

Hypertext Markup Language (HTML) é uma linguagem responsável por estruturar e apresentar conteúdos expostos na Web (*World Wide Web*) denominado então por marcador de hipertexto. O HTML5 foi proposto por Opera Software com o intuito de otimizar meios tecnológicos para melhor visualização de dados. Os códigos em HTML estam em constante ativação a cerca de dez (10) anos obtendo uma ampla aceitação de todos os adeptos do mesmo. O HTML está em sua quinta versão, buscando cada vez demonstrar importantes mudanças que foram realizadas em relação a sua implementação na Web e as suas novas funcionalidades, tais como: semântica e acessibilidade. HTML5 é uma válvula de escape para outros padrões como HTML, XHTML e HTML DOM. A linguagem de marcação é atribuída como tentativa para definição da escrita em HTML ou em sintaxe para XHTML. Acerca disso, os modelos incluídos de processos detalhados visam incentivar a implementação de vários outros meios interoperáveis, isso ocorre como forma de racionalização e melhoramento de documentos disponíveis introduzindo marcações de interfaces de programação de aplicativos (APIs). O HTML5 possui um potencial alto em situações de aplicações

em multiplataformas móveis. Vários recursos da linguagem são construídos pensando em dispositivos de baixa potência. O HTML5 após sua implementação objetivou facilitar o manuseio de dados trazendo para os desenvolvedores do mesmo, mais características não intrusivas, fazendo com que a sua utilização seja transparente para a pessoa que adquirirá no final. Sendo assim, as versões do HTML5 oferecem ferramentas para CSS e o JavaScript realizarem melhores análises fazendo com que, os web sites que o utilizam continuem com leveza e funcionalidade em suas aplicações. A importação de novas funções sintáticas tornou-se de eximia importância para o desenvolvimento de tags empregues no HTML5. As tags incluídas na nova versão do HTML foram as de ¡video¿, ¡audio¿, ¡header¿ e elementos ¡canvas¿, bem como a integração de conteúdos como SVG que recolocam as tags ¡object¿ genéricas. Exemplificando tais tags, com o HTML5 é possível assistir vídeos interativos na plataforma YouTube sem utilizar o Flash. Essas funções são aplicações que tencionam a projeção e manipulação de conteúdos gráficos e de multimídia na web sem a necessidade de plug-ins e APIs. Já as tags ¡section¿, ¡article¿, ¡header¿ e ¡nav¿, foram projetados com o intuito de enriquecer conteúdos considerados semânticos. O HTML5 estrutura formas que processe erros necessários de sintaxe dos documentos inválidos e assim sejam convencionados para abranger métodos uniformes em todos os browsers e usuários que aceitem suas congruências. Criado para ser utilizável como Web Aberta por desenvolvedores, situa-se em ligações que fazem referências a inúmeros recursos baseando-se em funções atípicas como conectividade, multimídia, gráfico, performance, entre outros.

3 Canvas

Tag utilizada na aplicação de gráficos e efeitos 3D, ele viabiliza uma vasta diversificação de representação gráfica utilizando JavaScript. Canvas é um elemento novo no HTML5, ele oferece técnicas programáticas para desenhar gráficos gozando de meios generalizados sendo por sua vez, disponível nas versões mais recentes do *Google Chrome, Firefox, Opera, Safari, Android* e *Mobile Safari.* Canvas renderiza imagens em bitmap, suas tags são específicas para edição através de *JavaScript* ou até mesmo das APIs. Para desenvolver formas básicas basta adquirir um contexto gráfico e utilizar a API do mesmo para suceder as mudanças necessárias no efeito. A tag ¡canvas¿ é definida desta forma em linguagem HTML5, sendo extensivamente difundida em formas personalizadas (semicírculos, quadrados, círculos e retângulos). O tipo de edição exemplificada ocorre de maneira exclusiva e pura, estando paralelamente a geradores de imagem de duas dimensões (2D), sendo elas compatíveis com CSS.

4 Exemplos

4.1 Localização do cursor

O código a seguir define um canvas no qual o mostra as coordenadas do cursor, dentro desse canvas, se o cursos estiver fora da área delimitada, as coordenadas não são mostradas.

```
1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <style>
```

```
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
    <canvas id="canvasPosicao" width="578" height="200"></canvas>
    <!--limite de deteccao-->
    <script>
      function escreverMensagem(canvas, mensagem) {
        var context = canvas.getContext('2d');
        context.clearRect(0, 0, canvas.width, canvas.height);
        context.font = '18pt Calibri';
        context.fillStyle = 'black';
        context.fillText(mensagem, 10, 25);
      }
      function getPosicao(canvas, evt) {
        var rect = canvas.getBoundingClientRect();
        return {
          x: evt.clientX - rect.left,
          y: evt.clientY - rect.top
        };
      }
      var canvas = document.getElementById('canvasPosicao');
      var context = canvas.getContext('2d');

      canvas.addEventListener('mousemove', function(evt) {
        var mousePos = getPosicao(canvas, evt);
        var mensagem = 'Posicao do Mouse : x = ' + mousePos.x + '; y = ' + mousePos.y;
        escreverMensagem(canvas, mensagem);
      }, false);
    </script>
  </body>
</html>
```

recursos/codigos/posicao.html

4.2 Mover dentro de um canvas

O código a seguir define um canvas no qual o retângulo percorre tal percurso até chegar as coordenadas indicadas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
    <canvas id="canvasMove" width="578" height="200"></canvas>
    <script>
      window.requestAnimFrame = (function(callback) {
        return window.requestAnimationFrame || window.
    webkitRequestAnimationFrame || window.mozRequestAnimationFrame
```

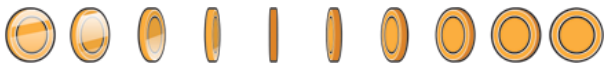
```
    || window.oRequestAnimationFrame || window.
    msRequestAnimationFrame ||
16        function(callback) {
17          window.setTimeout(callback, 1000 / 60);
18        };
19      })();
20
21      function desenhaRetangulo(retangulo, context) {
22        context.beginPath();
23        context.rect(retangulo.x, retangulo.y, retangulo.width,
    retangulo.height);
24        context.fillStyle = '#8ED6FF';
25        context.fill();
26        context.lineWidth = retangulo.borderWidth;
27        context.strokeStyle = 'black';
28        context.stroke();
29      }
30      function animate(retangulo, canvas, context, startTime) {
31        var time = (new Date()).getTime() - startTime;
32        var linearSpeed = 100;
33        var newX = linearSpeed * time / 1000;
34
35        if(newX < canvas.width - retangulo.width - retangulo.
    borderWidth / 2) {
36          retangulo.x = newX;
37        }
38
39        context.clearRect(0, 0, canvas.width, canvas.height);
40
41        desenhaRetangulo(retangulo, context);
42
43        requestAnimFrame(function() {
44          animate(retangulo, canvas, context, startTime);
45        });
46      }
47      var canvas = document.getElementById('canvasMove');
48      var context = canvas.getContext('2d');
49
50      var retangulo = {
51        x: 0,
52        y: 75,
53        width: 100,
54        height: 50,
55        borderWidth: 5
56      };
57
58      desenhaRetangulo(retangulo, context);
59
60      setTimeout(function() {
61        var startTime = (new Date()).getTime();
62        animate(retangulo, canvas, context, startTime);
63      }, 1000);
64    </script>
65  </body>
66 </html>
```

recursos/codigos/move_retangulo.html

```
1   <head>
2     <style>
3       body {
```

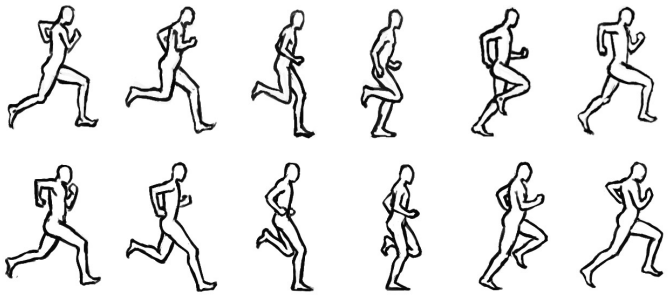
recursos/codigos/move_retangulo.html

Para criar uma animacao usando HTML5 Canvas, nós usamos o *requestAnimFrame* que habilita o navegador determinar o FPS adequado para cada animacao. Para cada frame de animação, nos atualizamos os elementos do canvas, limpando e redesenhando solicitando nova animação para dar aspecto de movimento.



O código é programado para ir imprimindo novos retângulos e excluir os anteriores, para dar a impressão de movimento, mas poderia imprimir uma nova versão da imagem, como uma moeda de um jogo que fique girando.

Caso essa exclusão não aconteça, ou não tenha sido programada, a mesma figura, vai continuar sendo exibida enquanto as novas são geradas quebrando todo o efeito de animação.



```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
    <canvas id="canvaClique" width="578" height="200"></canvas>
    <script>
      window.requestAnimFrame = (function(callback) {
        return window.requestAnimationFrame || window.
    webkitRequestAnimationFrame || window.mozRequestAnimationFrame
    || window.oRequestAnimationFrame || window.
    msRequestAnimationFrame ||
```

```
16          function(callback) {
17            window.setTimeout(callback, 1000 / 60);
18          };
19        })();
20
21        function desenhaRetangulo(retangulo, context) {
22          context.beginPath();
23          context.rect(retangulo.x, retangulo.y, retangulo.width,
    retangulo.height);
24          context.fillStyle = '#8ED6FF';
25          context.fill();
26          context.lineWidth = retangulo.borderWidth;
27          context.strokeStyle = 'black';
28          context.stroke();
29        }
30        function animate(lastTime, retangulo, runAnimation, canvas,
    context) {
31          if(runAnimation.value) {
32            var time = (new Date()).getTime();
33            var timeDiff = time - lastTime;
34            var linearSpeed = 100;
35            var linearDistEachFrame = linearSpeed * timeDiff / 1000;
36            var currentX = retangulo.x;
37
38            if(currentX < canvas.width - retangulo.width - retangulo.
    borderWidth / 2) {
39              var newX = currentX + linearDistEachFrame;
40              retangulo.x = newX;
41            }
42
43            context.clearRect(0, 0, canvas.width, canvas.height);
44            desenhaRetangulo(retangulo, context);
45            requestAnimFrame(function() {
46              animate(time, retangulo, runAnimation, canvas, context)
    ;
47            });
48          }
49        }
50        var canvas = document.getElementById('canvaClique');
51        var context = canvas.getContext('2d');
52
53        var retangulo = {
54          x: 0,
55          y: 75,
56          width: 100,
57          height: 50,
58          borderWidth: 5
59        };
60
61
62        var runAnimation = {
63          value: false
64        };
65
66        document.getElementById('canvaClique').addEventListener('
    click', function() {
67          runAnimation.value = !runAnimation.value;
68
69          if(runAnimation.value) {
70            var date = new Date();
```

```
71              var time = date.getTime();
72              animate(time, retangulo, runAnimation, canvas, context);
73            }
74          });
75          desenhaRetangulo(retangulo, context);
76
77      </script>
78    </body>
79  </html>
```

recursos/codigos/clique.html

5 Referências Bibliográficas

GEARY,David. **Core HTML5 Canvas: Graphics, Animation, and Game Development**. 2012.

https://www.tecmundo.com.br/navegador/2254-o-que-e-html-5-.htm

http://www.linhadecodigo.com.br/artigo/3488/entendendo-a-tag-canvas-no-html-5.aspx

https://pt.wikipedia.org/wiki/HTML5

https://developer.mozilla.org/pt-BR/docs/Web/HTML/HTML5

http://www.techtudo.com.br/artigos/noticia/2011/12/o-que-e-html5.html