HTML5 and JavaScript provide facilities to monitor the position of the mouse, to re-position elements on the screen, and to make calculations to provide an unpredictable factor in a game. In this article, I will describe individual features and then put them together to show you how to create a chasing game.

This application was inspired by a political website that kept the visitor from clicking on a button that promised more details on a candidate's position. As is my practice, I created a game in which the player could move the mouse and attempt to reach a picture of my granddaughter. The cursor would change from time to time, but with no particular pattern, between photos of her brothers.

The opening screen would look something like Figure 1.

![Opening screen](image1)

The application actually is quite simple to put together, primarily because of the benefits of event programming provided by JavaScript. One event is the mouse moving over the image of the wagon. The other type of event is the timed interval event. You can use the distinct parts of this program for many other applications.

**Controlling the icon for the mouse cursor**

The icon for the mouse cursor, the image that appears on the screen when someone moves the mouse, can be set by code to either one of the built-in icons or an image from an image file. The programmer can "hedge" the decision by specifying a string such as "url('liam2.gif'), pointer" to specify use of the `liam2.gif` file, if it exists, or, if not, the standard `pointer` icon. This is similar to how fonts are specified either in the style element or the script element: we programmers can specify our first choice and then fallback choices.

The application uses two variables to toggle between the Grant or Liam choice. The way to set a cursor for the mouse moving over an element, in this case a canvas referenced by the variable `c`, is to use an assignment statement setting the `cursor` property of the `style` property. For example,

```
c.style.cursor = othercursor;
```
where at some previous point the variable has been set using an assignment statement such as
othercursor = "url('liam2.gif'),pointer";

sets the cursor over the canvas element to be the image in the liam2.gif file. You can change the cursor, as I do for this application. You also can set up distinct cursor icons for movement over different elements in your document.

**Periodic event handling and random processing**

Random processing, also termed pseudo-random, probabilistic or stochastic processing, is an important feature of games and also simulations of real-life phenomena. I implement the unpredictable changing of the cursor icon by setting up a predictable timed event. The timed event is implemented using the setInterval function in the init function shown in Listing 3. The function specified as the event handler, maybeswitch, explained later and shown in Listing 5, is invoked every 2 seconds. Within this function, my code invokes the Math.random function to make the determination of whether or not to switch cursor pictures. As a consequence, the switching appears unpredictable. To put it another way, at predictable intervals of time, a calculation is performed using the pseudo-random facility.

**The <body> element and <style> directives for setting up the application**

This application makes use of a canvas element, an img element inside an a element, and the cursor for the mouse. The only use I made of the canvas is to draw a rectangle. You can imagine a more interesting picture to serve as background to the girl in a wagon. The canvas and the img and the a elements are shown in the body element displayed in Listing 1.

```html
<body id="body" onLoad="init();">
<canvas id="canvas" width="1000px" height="650px">
Your browser does not recognize canvas</canvas>
</body>
```

Listing 1: the body element

A function I wrote and named init is invoked by action of the onLoad attribute in the <body> tag. This is standard for these types of applications. The style section, shown in Listing 2, sets up the positioning of the canvas and the wagon elements to be absolute, with the canvas positioned in the upper left corner. The wagon is given a z-index to put it on top of the canvas. I gave the body a margin directive to provide some screen space outside of the body.

```html
<style>
body { width:100%; height: 100%; margin:5px; }
#wagon {position:absolute; top:0px; left:0px; }
#canvas {position:absolute; top:40px; left:40px; }
</style>
```

Listing 2: the style element

Do keep in mind that the girl in wagon image is a distinct element on top of the canvas. My code does not draw this image on the canvas.

**Initializing the application, including setting up the events**

As you just read, the call to my init function is set up by the onLoad attribute in the <body> tag. Listing 3 shows the global variables and the init function.

```javascript
var nextcursor="url('grant1.gif'),pointer";
var othercursor = "url('liam2.gif'),pointer";
var prob = .3333333;
var w;
var mx;
var my;
var c;
var ctx;
var cwidth = 1000;
var cheight = 600;
var halfwidth;
var halfheight;
var tid; //not used at present. Interval keeps going
function init(){
    w = document.getElementById("wagon");
    w.style.top="200px";
    w.style.left="200px";
    halfwidth = .5* Number(document.pic.width);
    halfheight = .5* Number(document.pic.height);
    w.addEventListener("mouseover",jump,false);
    c = document.getElementById("canvas");
    c.style.cursor=othercursor;
    ctx = c.getContext("2d");
    ctx.strokeRect(0,0,cwidth,cheight);
    tid = setInterval(maybeswitch,2000);
}
```

Listing 3: The global variables and the init function

The nextcursor and othercursor variables are used to implement the toggling between images representing the two brothers, who do spend time chasing their sister.

As I indicate in my comment, more to myself than anyone else, the variable tid is not used. My code never stops the timed event.

The init function acquires references to the canvas and wagon elements. The reference to the wagon is used to position it and also set up the handling of the mouse being over the image. The variables halfwidth and halfheight are used for the calculation of the jump of the wagon. The addEventListener call sets up the mouseover event: when the player moves the mouse cursor over the wagon, JavaScript invokes the function named jump.

I decided it was useful to draw a rectangle so that the player knew the boundaries of the game. Notice that the wagon can poke outside of the rectangle. The last task performed by the init code is to set up the timed event.
Programming the jump

My goal is to make the wagon jump when the player attempts to move the cursor on top of it. This application takes advantage of the particular image being somewhat bigger than it appears. The redwagon1.png image actually is a rectangle. There is a transparent background that has the effect that the mouse can be on top of the image without being on top of the visible wagon and girl.

The event handler, the jump function, has access to the mouse location relative to the wagon, NOT relative to the canvas or the screen. This proved useful in my decision to calculate a jump that is half the wagon width horizontally and half the wagon height vertically away from the mouse.

The next decision I needed to make was what to do when the wagon was pushed off of the canvas. I decided to make it jump to the other side of the canvas: to the top if it was moving off the bottom; the bottom, if it was moving up above the top; the left side, if it was being moved to the right; and the right side, if it was moving to the left. The code is shown in Listing 4.

```
function jump(ev) {
    var nx; var ny;
    if (ev) {
        if (ev.layerX || ev.layerX == 0) {
            mx = ev.layerX;
            my = ev.layerY;
        } else if (ev.offsetX || ev.offsetX == 0) {
            //
            mx = ev.offsetX;
            my = ev.offsetY;
        }
    }
    else {
        return; //no ev event
    }
    var curx = parseInt(w.style.left,10);
    var cury = parseInt(w.style.top, 10);
    nx = curx + .5*(halfwidth-mx);
    ny = cury + .5*(halfheight-my);
    if (nx<0) {
        nx = cwidth-halfwidth;
    }
    if (nx>cwidth) {
        nx = 0;
    }
    if (ny<0) {
        ny = cheight-halfheight;
    }
    if (ny>cheight) {
        ny = 0;
    }
    w.style.left = nx +"px";
    w.style.top = ny +"px";
    nextcursor = othercursor;
    c.style.cursor=nextcursor;
    if (Math.random()<prob) {
        function maybeswitch() {
            if (Math.random()<prob) {
                othercursor=c.style.cursor;
                c.style.cursor=nextcursor;
                nextcursor = othercursor;
            }
        }
        Listing 5: The maybeswitch function
    }
}
```

Putting it together

The trick to this application is the fact that the two objectives: random switching of the cursor and the jumping to avoid the mouse, are programmed independently. You have seen all of the code. To re-enforce what has already been written, here is a function calling/called table for the 3 functions. I included the 3rd column to be consistent with the function tables I and others use in documentation but in this case the column has no entries. The functions used built-in JavaScript functions, but nothing else.

<table>
<thead>
<tr>
<th>Function</th>
<th>Called by / invoked by</th>
<th>calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>Action of onLoad in &lt;body&gt;</td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td>Action of addEventListener in init</td>
<td></td>
</tr>
<tr>
<td>maybeswitch</td>
<td>Action of setInterval in init</td>
<td></td>
</tr>
</tbody>
</table>

There are many ways you can make this application your own. You will, of course, use different pictures. You also may consider changing the target, which for me is the redwagon1.png picture, based on player action and/or random processing. You can program a different type of jump. As I indicated, the contents of the canvas can itself be a picture. Notice that the z-levels in the style element specify that the img element is on top of anything drawn on the canvas. The z axis is moving out from the screen towards the user. When the z-level is referenced or set in JavaScript, as opposed to Cascading Style Sheets, it is called zlevel.

Learn more

There are many sources, online and in-print and some sort of e-books, for learning HTML5 and JavaScript techniques. Here are links to my recent books and the website for this example.

- The Essential Guide to HTML5: Using Games to learn HTML5 and JavaScript, http://www.friendsofed.com/book, html?isbn=9781430233831. This is a text for beginners at programming as well as more experienced programmers who want to learn about HTML and JavaScript, including the new features of HTML5.
• HTML5 and JavaScript Projects, http://www.apress.com/9781430240327. This book is more advanced than the first one. There are several chapters on canvas and on mouse events.

• To see the chasing image application in action and to view the source code, go to http://faculty.purchase.edu/jeanine.meyer/html5/chasingwagon1.html.

Jeanine Meyer lives just north of New York City and currently teaches at Purchase College/SUNY after many years at IBM, doing research on robotics and manufacturing and consulting on educational grants. She likes providing programming examples for her Mathematics/Computer Science and New Media students and really, really likes working with images and video clips of her granddaughter and other family members.