Listing 1: Transcript of Usability Study

```
R: (Researcher)
P: (Participant)

====Flyouts On====

    First system, Flyouts ON.

R:  Start the system by pressing Apple K.
P:      Okay, is there an Apple on here?
R:  Yeah, I've mapped Ctrl to Apple.
P:      Okay, great.
P:      I'll do what I have to do to get started, and...
        *Presses Apple K*
        *Chibipoint interface appears*
P:      Okay.
```

0.1 ./README

```
=Rights:
ChibiPoint is made by Alex Birch.
Source code license is GNU GPL v3. Full text available in file `LICENSE`.


=Installation:
Using Google Chrome,
1. Go to chrome://extensions/
2. Ensure `Developer mode` at the topis checked.
3. `Load unpacked extension`
4. Choose this folder, `chibipoint`.
5. ChibiPoint should appear in your list of Extensions, and be `Enabled`.
6. If you wish, you can customise the invocation shortcut using the `Keyboard Shortcuts` link at the bottom of chrome://extensions/


=Invocation:
Browse the web as usual.
Press Cmd+K (Mac) / Ctrl+K (Windows) to invoke ChibiPoint interface.

You should see a grid. If you don't, that's bad. Probably an installation or keybinding problem.


=Usage:
Press the shortcuts you see on the grid to `drill down' in that direction.
Labels fly out from suggested clickables. These `flyouts' can be invoked by pressing their corresponding key.

By default, you drill with the numpad directions.
Flyout shortcuts are mapped to a 3x3 grid on the left hand of a Qwerty keyboard.

Caps lock is not supported.


=Customizing keys:
By default we use numpad to `drill down'. If you don't have a numpad, you should probably change the code a little.

Lines 80-154 concern keybindings. Sorry the world isn't nicer.
There are a few defaults you might want to try out, so fiddle around with
```

```
numpad = true and Dvorak =false.

Just edit the code, and refresh any page that was using ChibiPoint; it should update immediately.
```

0.2 ./manifest.json

```
{
"name": "ChibiPoint",
"description": "Point at webpages using the keyboard",
"version": "0.5",
"content_scripts": [
  {
                      "matches":["http://*/*","https://*/*"],
    "css": ["styles.css"],
    "js": ["src/lib/require.js","src/lib/require-cs.js","src/depend.js"],
    "run_at": "document_start"
  }
],
"permissions": [
  "tabs",
  "activeTab"
],
"background": {"scripts":["src/background.js"]},
"commands": {
  "toggle-feature-foo": {
    "suggested_key": {
      "default": "Ctrl+K",
      "mac": "Command+K"
    },
    "description": "Invoke ChibiPoint"
  }
},
"manifest_version": 2,
"web_accessible_resources": [
  "*.js"
]
}
```

0.3 ./styles.less

```
// out: styles.css
// compress: false, strictMath: false


@gridcol: #ccc;
@gridthick: 1px;

@growcoeff: 1.05;


body {
    padding: 0;
}

.chibiPoint_gridContainer {
  padding: 0;
  //top: 8.3%;
  top:0;
  left:calc(((100%*(@growcoeff -1))/2) / @growcoeff);

  position:fixed;
  width:calc(100%/@growcoeff);
  height:calc(100%/@growcoeff);

  pointer-events: none;

  z-index:2147483646;
}

.chibiPoint_hiddenGridContainer {
  display:none;
}

.chibiPoint_grid {

  position:relative;
  left:calc(-(100%*(@growcoeff - 1))/2);

  //top:-10%;
  top:0;
  //float:left;

  width:calc(100%*@growcoeff);
  height:calc(100%*@growcoeff);

    //overflow: auto;
  display: table;
    table-layout: fixed;

  pointer-events: inherit;
}

.chibiPoint_gridBorderOn {
  border: @gridthick solid @gridcol;
  border-width: @gridthick @gridthick @gridthick @gridthick;
  //outline: 1px solid 0;
}

.chibiPoint_gridBorderOff {
}

.chibiPoint_row {
    //border: 1px solid #ccc;
    //border-width: 0 1px 1px 0;

    display: table-row;
  pointer-events: inherit;


}
```

```

@cellratio:calc(100%/3);

.chibiPoint_cell {
  padding:0;

  width:@cellratio;
  height:@cellratio;
  display: table-cell;


  // text
  line-height: @cellratio;
  text-align:center;
  vertical-align:middle;
  color: #000;
  color:rgba(0, 0, 0, 0.5);
  font-family:Geneva, Arial,Helvetica, sans-serif;

  pointer-events: inherit;

  position:relative;
  overflow:visible;
  clear:both;
}

.chibiPoint_cellBorderOn {
  /*border: @gridthick solid @gridcol;
  border-width: @gridthick @gridthick @gridthick @gridthick;
  //border-style:double;
  outline: 1px solid 0;
  outline-offset: 1px;*/

 //this one!
 box-shadow: 0 0 0 1px #fff, 0 0 02px #888;
}

.chibiPoint_cellBorderOff {
}

@backingprop: 0.8;
@radius: 15%;
@backinggrey: 230;

.chibiPoint_backing {
  position:absolute;

  top:calc(100%*(1-@backingprop)/2);
  left:calc(100%*(1-@backingprop)/2);

  width:calc(100%*@backingprop);
  height:calc(100%*@backingprop);
  background:rgba(@backinggrey, @backinggrey, @backinggrey, 0.25);

  z-index:-1;
```

```

    -moz-border-radius: @radius @radius;
    -webkit-border-radius: @radius @radius;
    border-radius: @radius @radius;

    //background-blend-mode:difference, normal; // Chrome Canary
    //-webkit-background-blend-mode:difference, normal; // WebKit Nightly
}

@haircol:rgba(255, 0, 0, 1);
@hairthick:1px;
@hairborderthick:1px;

.chibiPoint_fullContainer {
  top:0;
    left:0;

    width:100%;
    height:100%;

    position:fixed;
    padding:0;

    pointer-events: none;
}

.chibiPoint_shownCrosshairs {
    .chibiPoint_fullContainer;

    z-index:2147483645;
}

.chibiPoint_hiddenCrosshairs {
    pointer-events: none;

    display:none;
}

.chibiPoint_hairBorder {
    //border-width: @hairborderthick @hairborderthick @hairborderthick @hairborderthick;
    //border-color:#00ffff;

    //outline-color:;
    //outline: 1px solid 0;
}

.chibiPoint_hairlinecommon {
    padding:0;

    position:relative;

    pointer-events: inherit;

    background:@haircol;

  .chibiPoint_hairBorder;
}

.chibiPoint_horizontalHair {
  .chibiPoint_hairlinecommon;

  left:inherit;

  width:inherit;
  height:@hairthick;
}

.chibiPoint_verticalHair {
  .chibiPoint_hairlinecommon;

  top:-@hairthick;

  width:@hairthick;
  height:inherit;
}

.chibiPoint_targeted {
    background: yellow!important;
    //padding: 3px 5px;
    //margin: -3px -5px;
    //line-height: 1.7;

    border-radius: 3px;
    /*border-color: red!important;
    border: 3px solid black;
    box-sizing: border-box;*/
    //border-radius: 3px;
    box-shadow:  0 0 10px #ff00ff;

  // turn off croshairs:
   //display: none;

    background-blend-mode: multiply, normal; // Chrome Canary
    -webkit-background-blend-mode: multiply, normal; // WebKit Nightly

    //display:inline-block;
}

.chibiPoint_painted {
    background: cyan!important;
    //padding: 3px 5px;
    //margin: -3px -5px;
    //line-height: 1.7;

    border-radius: 3px;
    /*border-color: red!important;
    border: 3px solid black;
    box-sizing: border-box;*/
    //border-radius: 3px;
    box-shadow:  0 0 10px #ff0000;

    background-blend-mode: multiply,
        normal; // Chrome Canary
    -webkit-background-blend-mode: multiply, normal; // WebKit Nightly
    //display:inline-block;
}

.chibiPoint_paintedUnique0 {
  background: rgba(0,173,0,@flyoutAlpha) !important;
}
.chibiPoint_paintedUnique1 {
  background: rgba(13,255,150,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique2 {
  background: rgba(249,25,137,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique3 {
  background: rgba(249,112,12,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique4 {
  background: rgba(186,167,19,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique5 {
  background: rgba(172,0,237,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique6 {
  background: rgba(240,172,24,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique7 {
  background: rgba(8,160,94,@flyoutAlpha)!important;
}
.chibiPoint_paintedUnique8 {
  background: rgba(135,54,94,@flyoutAlpha)!important;
}

.chibiPoint_cluck {
    background: red!important;
    //padding: 3px 5px;
    //margin: -3px -5px;
    //line-height: 1.7;

    /*border-radius: 3px;
    border-color: red!important;
    border: 3px solid black;*/

    //display:inline-block;
}

.chibiPoint_flyoutContainer {
 .chibiPoint_fullContainer;
```

```
  z-index:2147483647;
}

.chibiPoint_shownfC {
}

.chibiPoint_hiddenfC {
 display:none;
}

.chibiPoint_aFlyout {
  position: absolute;
 width:100%;
 height:100%;
}

.chibiPoint_aFlyoutHidden {
 display: none;
}

.chibiPoint_flyout {
  //background: rgba(0,173,0, 0.30);
  //outline: 1px solid 0!important;
  border: 1px solid black;
  position: absolute;
  top:0;
  left: 0;
 width:20px;
 text-align: center;
  font-family:Geneva, Arial,Helvetica, sans-serif;

  /*-webkit-text-stroke: 1px #000000;
  -webkit-text-fill-color: #FFFFFF;*/
  color: white;
  text-shadow:
   -1px -1px 0 #000,
    1px -1px 0 #000,
    -1px 1px 0 #000,
     1px 1px 0 #000;
}

@flyoutAlpha: 1.0;

.chibiPoint_flyoutUnique0 {
  background: rgba(0,173,0,@flyoutAlpha);
}
.chibiPoint_flyoutUnique1 {
  background: rgba(13,255,150, @flyoutAlpha);
}
.chibiPoint_flyoutUnique2 {
  background: rgba(249,25,137, @flyoutAlpha);
}
.chibiPoint_flyoutUnique3 {
  background: rgba(249,112,12, @flyoutAlpha);
}
```

```
.chibiPoint_flyoutUnique4 {
    background: rgba(186,167,19, @flyoutAlpha);
}
.chibiPoint_flyoutUnique5 {
    background: rgba(172,0,237, @flyoutAlpha);
}
.chibiPoint_flyoutUnique6 {
    background: rgba(240,172,24, @flyoutAlpha);
}
.chibiPoint_flyoutUnique7 {
    background: rgba(8,160,94,@flyoutAlpha);
}
.chibiPoint_flyoutUnique8 {
    background: rgba(135,54,94, @flyoutAlpha);
}

.chibiPoint_lineUnique0 {
    stroke: rgb(0,173,0,@flyoutLineAlpha);
}
.chibiPoint_lineUnique1 {
    stroke: rgb(13,255,150, @flyoutLineAlpha);
}
.chibiPoint_lineUnique2 {
    stroke: rgb(249,25,137, @flyoutLineAlpha);
}
.chibiPoint_lineUnique3 {
    stroke: rgb(249,112,12, @flyoutLineAlpha);
}
.chibiPoint_lineUnique4 {
    stroke: rgb(186,167,19, @flyoutLineAlpha);
}
.chibiPoint_lineUnique5 {
    stroke: rgb(172,0,237,@flyoutLineAlpha);
}
.chibiPoint_lineUnique6 {
    stroke: rgb(240,172,24, @flyoutLineAlpha);
}
.chibiPoint_lineUnique7 {
    stroke: rgb(8,160,94,@flyoutLineAlpha);
}
.chibiPoint_lineUnique8 {
    stroke: rgb(135,54,94,@flyoutLineAlpha);
}


.chibiPoint_lineDiv {
}
```

```

.chibiPoint_lineSvg {
}

@flyoutLineAlpha:1;
@flyoutLineWidth:2;
.chibiPoint_line {
 //stroke:rgb(0,173,0);
 stroke-width:@flyoutLineWidth;
}

.chibiPoint_evaluator {
  position:fixed;

  pointer-events: none;

  font-family:Geneva, Arial,Helvetica, sans-serif;

  color:red;

  width:100%;
  height:100%;

  display:none;

}

.chibiPoint_timer {
  .chibiPoint_evaluator;

  left:10;
  top:10;
}

.chibiPoint_keyCount {
  .chibiPoint_evaluator;

  left:10;
  top:30;
}
```

0.4 ./src/depend.js

```
//// Cheekily change DOM prototype before page loads
// http://stackoverflow.com/questions/8915461/how-to-find-out-what-event-listener-types-attached-to-specific-html-element-in-c
// from spektom, thanks to kdzwinel

var injectedJS = "\
(function(original) { \
  Element.prototype.addEventListener= function(type, listener, useCapture) { \
    var attr =this.getAttribute('_handlerTypes'); \
  var types = attr ?attr.split(',') : []; \
  var found = false; \
  for (var i = 0; i < types.length;++i) { \
    if (types[i] == type) { \
      found = true; \
      break; \
    } \
  } \
  if (!found) { \
    types.push(type); \
  } \
this.setAttribute('_handlerTypes', types.join(',')); \
  return original.apply(this,arguments); \
 } \
})(Element.prototype.addEventListener); \
\
(function(original) { \
Element.prototype.removeEventListener= function(type, listener,useCapture) { \
  var attr =this.getAttribute('_handlerTypes'); \
  var types = attr ?attr.split(',') : []; \
  var removed = false; \
  for (var i = 0; i < types.length;++i) { \
    if (types[i] == type) { \
      types.splice(i, 1); \
      removed = true; \
      break; \
    } \
  } \
  if (removed) { \
this.setAttribute('_handlerTypes', types.join(',')); \
  } \
  return original.apply(this,arguments); \
 } \
})(Element.prototype.removeEventListener); \
";

var script =document.createElement("script");
script.type = "text/javascript";
script.appendChild(document.createTextNode(injectedJS));
document.documentElement.appendChild(script);

//////

url = "/src/";
```

```
if (chrome.extension != undefined) {
 url =chrome.extension.getURL("/src/");
}

require({ baseUrl:url },["lib/jquery-2.1.0.min","lib/within", "lib/Blob","lib/FileSaver", "evaluator", "test","trace", "lookup", "testonly","setup", "Grid", "Crosshairs","Flyout"], function ($, within, blob,saver, evaluator, test, trace,lookup, testonly, setup, gridclass,crosshairclass, flyout) {
 // default namespace
 window.birchlabs = {};

   //This function is called when scripts/helper/util.js is loaded.
   //If util.js calls define(), then this function is not fired until
   //util's dependencies haveloaded, and the util argument willhold
   //the module value for"helper/util".

   if (typeof(chrome) == "object" && chrome.extension) {
   //chrome.extension.sendRequest({'get_options': true},function(response) {
       setup.main();
     //});
   } else {
     setup.main();
   }
});
```

0.5 ./src/test.js

```
define(["lib/jquery-2.1.0.min","lib/within", "lib/Blob","evaluator", "trace", "lookup","testonly", "Grid", "Crosshairs","Flyout"], function (jq, within,blob, evaler, tracer, lookup,testonly, gridclass, crosshairclass,flyout) {
var trace = tracer.trace;
var supertrace = tracer.supertrace;

window.birchlabs =window.birchlabs||{};
var birchlabs = window.birchlabs;

window.evaluator =window.evaluator||{};
var evaluator = window.evaluator;

//console.log(birchlabs);
```

```javascript

function makeContainers() {
  var Flyout = birchlabs.Flyout;
  var Grid = birchlabs.Grid;
  var Crosshairs = birchlabs.Crosshairs;

  if (birchlabs.flyoutsOn) {
    Flyout.makecontainer(document.documentElement);
    var flyouts = [];
  }
  Flyout.hide();

  var grid = new Grid();
  var container = makeGridContainer(document.documentElement);

  var crosshairs = new Crosshairs(document.documentElement,grid);
  crosshairs.hide();

  birchlabs.grid = grid;
  birchlabs.container = container;
  birchlabs.crosshairs = crosshairs;
  //birchlabs.theFlyout = flyout;
  birchlabs.flyouts = flyouts;
}

function makeGridContainer(root) {
  var parent =document.createElement('div');
  parent.className ='chibiPoint_gridContainer chibiPoint_hiddenGridContainer';
  $(root).append(parent);
  return parent;
}

function init() {
  birchlabs.evaluateTabMode = false;
  birchlabs.disallowTabbing = false;
  birchlabs.flyoutsOn = true;
  if (birchlabs.evaluateTabMode) {
    birchlabs.flyoutsOn = false;
  }
  makeContainers();
  evaler.makeEvaluators();
  var Flyout = birchlabs.Flyout;
  var Grid = birchlabs.Grid;
  var Crosshairs = birchlabs.Crosshairs;

  var flyouts = birchlabs.flyouts;
  var grid = birchlabs.grid;
  var crosshairs = birchlabs.crosshairs;

  birchlabs.targetedElement = null;
```

```

var keycodes = {
  "backspace": 8,
  "tab": 9,
  "enter": 13,
  "spacebar": 32,
  "escape": 27,
  "n": 78,
  "p": 80,
  "g": 71,
  "f3": 114,
  "f4": 115,
  "dot": 46,
  "zero": 48,
  "activate": 167
};

// Do you have a numpad?
var numpad = true;

if (numpad) {
  // Drilling: numpad
  // 7 8 9
  // 4 5 6
  // 1 2 3
  var numpadMappings ={"7":[55,7], "↙
     ↪ 8":[56,8], "9":[57,9],
                         "4":[52,4],"5"↙
                            ↪ :[53,5], "↙
                            ↪ 6":[54,6],
                                       "1":[49,1],"2"↙
                                          ↪ :[50,2], "↙
                                          ↪ 3"↙
                                          ↪ :[51,3]};
  var zeroKey = keycodes.zero;
  // Undo drilling:
  // 0
}

// Are you one of.. those?
var dvorak = false;

if (dvorak) {
  // Flyouts: right hand dvorak grid
  // G C R
  // H T N
  // M W V
  var flyoutShortcuts = [103,99,114,
                         104,116,110,
                         109,119,118];

  if (!numpad) {
    // Drilling: left hand dvorak↙
       ↪ grid
    // ' , .
    // A O E
    // ; Q J
    var numpadMappings ={"7":[39,7],↙
       ↪  "8":[44,8], "9":[46,9],
"4":[97,4], "5":[111,5], "6":[101,6],
"1":[59,1], "2":[113,2],"3":[106,3]};
    // Undo drilling:
    // K
    var zeroKey = 107;
   }
  } else {
   // Flyouts: left hand qwerty grid
   // Q W E
   // A S D
   // Z X C
   var flyoutShortcuts =[113,119,101,
             97,115,100,
             122,120,99];

   // alphabet flyouts
   // nah, these won't help anyone
   /*var flyoutShortcuts = [97,98,99,
             100,101,102,
103,104,105];*/

   if (!numpad) {
    // Flyouts: right hand qwerty grid
    // U I O
    // J K L
    // N M ,
    var flyoutShortcuts =[117,105,111,
             106,107,108,
             109,44,46];
    // Drilling: left hand qwerty grid
    // Q W E
    // A S D
    // Z X C
    var numpadMappings ={"7":[113,7], "8":[119,8],"9":[101,9],
"4":[97,4], "5":[115,5], "6":[100,6],
"1":[122,1], "2":[120,2],"3":[99,3]};
    // Undo drilling:
    // V
    var zeroKey = 118;
   }
  }

  birchlabs.numpadMappings = numpadMappings;

  if (birchlabs.flyoutsOn) {
   for (var i=0; i<flyoutShortcuts.length; i++) {
    flyouts.push(new Flyout(flyoutShortcuts[i]));
   }
  }

  //createGrid();
```

```
   function setEvents(rootNode) {
     var doc = rootNode.contentDocument || rootNode;
     var body = rootNode.body;

     if (!body ||!body.addEventListener) {
       return;
     }

     lookup.setAlternativeActiveDocument(document);


    if (document.getElementById("chibipointtestmode") != null) {
     birchlabs.testmode = true;
    }

    //trace(birchlabs.testmode);

     // usually on keydown
     //processSearch();
     //draw();
    //var container = birchlabs.container;

    if (birchlabs.testmode) {
     toggleGrid();
    }

    var $window   = $(window),
      resize_ok = true,
      timer;

    timer = setInterval(function() {
      resize_ok = true;
    }, 50);

    liveTargeter = function() {
     if (resize_ok === true) {
        resize_ok = false;
//trace($window.width());
       //trace($(".verticalHair").first().offset().left);
       //highlightTarget();
crosshairs.highlight(getDocRoot());
       pointFlyouts();
      }
    };

    $window.resize(liveTargeter);
    $window.scroll(liveTargeter);

    // some keys can only be caught by keydown
doc.addEventListener('keydown',function(
     ev) {
          var code = ev.keyCode;
          var ascii =String.fromCharCode(code);

          if (code == keycodes.enter) {
evaler.evaluatorIncrementKeys(code);
            if(birchlabs.evaluateTabMode) {
              evaler.evaluatorWriteState(document.activeElement);
            } else {
              gridclick(grid);

              lookup.stopEvent(ev);
            }
          }

          if (code == keycodes.tab) {
            if(birchlabs.evaluateTabMode) {
              // negative, for shift-tab!
              var code2 = ev.shiftKey? -code : code;

evaler.evaluatorIncrementKeys(code2);
evaler.checkFocusAfterWait();
            } else {
              if(birchlabs.disallowTabbing) {
                alert("Please do not press tab during this experiment!");
                lookup.stopEvent(ev);
document.activeElement.blur();
              }
            }
          }
          /*if (code ==keycodes.spacebar) {
evaler.evaluatorIncrementKeys(code);
          }*/
          if(!birchlabs.evaluateTabMode) {
            if (code ==keycodes.escape) {
              if (!gridIsEmpty()) {
evaler.evaluatorIncrementKeys(code);
                closeGrid();

                lookup.stopEvent(ev);
              }
            }
          }
        });
```

```
    // no key bindings in tab mode
    if(!birchlabs.evaluateTabMode) {
doc.addEventListener('keypress',function (ev) {
     if(lookup.isInputElementActive (doc)) {
      var tracer = doc.getElementById("trace");
      if (tracer != null) {
doc.getElementById("trace").innerHTML= code;
      }
      return;
     }
     var code = ev.keyCode;
     var ascii =String.fromCharCode (code);

     if (!lookup.is_shortcut(ev)&& ascii &&[keycodes.enter].indexOf(code) == -1){
      var tracer = doc.getElementById("trace");
      if (tracer != null) {
doc.getElementById("trace").innerHTML= code;
      }

      /*if (numpadtype) {
       var num = code-numpadstart;
       if (num<10 && num > 0){
        if (gridIsInUse()) {
         drill(num, grid,crosshairs);

         lookup.stopEvent(ev);
        }
       } else if (code ==keycodes.zero) {
        backup(grid,crosshairs);

        lookup.stopEvent(ev);
       }
      }*/
      for (var j in numpadMappings) {
       if (code ==numpadMappings[j][0]) {
        if (gridIsInUse()) {
evaler.evaluatorIncrementKeys(code);
drill(numpadMappings[j][1], grid,crosshairs);

         lookup.stopEvent(ev);
         break;
        }
       }
```

```
            }
            if (code == zeroKey) {
evaler.evaluatorIncrementKeys(code);
              backup(grid, crosshairs);

              lookup.stopEvent(ev);
            }
            if (code ==keycodes.activate ) {
evaler.evaluatorIncrementKeys(code);
              toggleGrid();

              lookup.stopEvent(ev);
            }
            if (birchlabs.flyoutsOn) {
              for (i=0; i<flyoutShortcuts.length; i++) {
                if (code == flyoutShortcuts[i]) {
evaler.evaluatorIncrementKeys(code);
                  evaler.evaluatorWriteState(flyouts[i].getTarget());
                  flyouts[i].doClick();

                  closeGrid();

                  lookup.stopEvent(ev);
                  break;
                }
              }
            }
          }
        }, false);
        }
      }

      // guess this defers setting events until root is loaded?
      var rootNodes = [document].concat(lookup.getRootNodes());
      for (var i = 0; i < rootNodes.length; i++) {
      var rootNode = rootNodes[i];
      if (rootNode.contentDocument) {
rootNode.addEventListener('load', function(ev) {
setEvents(ev.target.contentDocument);
        });
      }
      else if(!rootNode.contentDocument || rootNode.contentDocument.readyState == 'complete') {
setEvents(rootNode.contentDocument ? rootNode.contentDocument : rootNode);
      }
```

```
  }
    //draw();
}

function getDocRoot() {
  var rootNodes = [window].concat(lookup.getRootNodes());
  for (var x = 0; x <rootNodes.length; x++) {
    var doc2 =rootNodes[x].document || rootNodes[x].contentDocument;
    if (!doc2 || !doc2.body)
      continue;
    return doc2;
  }
}

function getAllClickablesInBoundsAndSort( root,rect, rects) {
    var found = [];
    var buckets = [];
    for (var j=0; j<rects.length;j++) {
      buckets.push([]);
    }

    var left = rect.left;
    var top = rect.top;
    var width = rect.width;
    var height = rect.height;

    var left2;
    var top2;
    var width2;
    var height2;

    var cache = false;

    var r;
    var i;
    var res;
    var res2;
    //var q;
    //var ew;
    var itRect;
    // some of the obscure ones are guesses
    var careElements = {"A":true,
                        "INPUT":true,
"TEXTAREA":true,
                        "SELECT":true,
"BUTTON":true//,
//"output":true,
//"command":true,
                        //"kbd":true
                        };

    $(root).find("[_handlerTypes],A, INPUT, SELECT, TEXTAREA,
BUTTON").each(function() {
        var found = false;
        var handlers =this.getAttribute('_handlerTypes');
        if (handlers != null ) {
          var h = handlers.split(",");
          for (i=0; i<h.length; i++) {
            if (h[i] == 'click' ||h[i] == 'mousedown') {
              found = true;
              break;
            }
          }
        }
        if (!found) {
if(careElements[this.nodeName]) found= true;
        }
        if (found) {
          //if ($(this).css("visibility").indexOf("hidden") != -1 || !$(this).css("display").indexOf("none") !=-1) return;
          var visible =!$(this).is(':hidden') &&$(this).is(':visible');
          if (visible) {
            //if(this ==document.documentElement ) return this.ret.push(this);
            if(this == document.documentElement ) return;

            itRect = this.getBoundingClientRect();

            res = !( (itRect.top >top+height) || (itRect.top+itRect.height < top) || (itRect.left> left+width ) ||(itRect.left+itRect.width < left));

          // it's certainly in thegrid somewhere
            if(res) {

              // now find whichbuckets to put it in
                for (i=0;i<rects.length;i++) {
                  r = rects[i];
                  left2 = r.left;
                  top2 = r.top;
                  width2 = r.width;
                  height2 = r.height;
                  res2 =  !((itRect.top > top2+height2) ||(itRect.top +itRect.height < top2) ||(itRect.left > left2+width2 ) ||(itRect.left+itRect.width < left2));

                  if (res2) {
buckets[i].push(this);
                  }
                }
            }
          }
        }
      });
      return buckets;
    }

  function highlightTarget() {
    var crosshairs = birchlabs.crosshairs;
    var flyouts = birchlabs.flyouts;
    var grid = birchlabs.grid;

    var doc2 = getDocRoot();
```

```

      var crosshaired = crosshairs.highlight(doc2).get(0);

      // some of the obscure ones are guesses
      var careElements = {"A":true,
                         "INPUT":true,
"TEXTAREA":true,
                         "SELECT":true,
"BUTTON":true//,
//"output":true,
//"command":true,
                         //"kbd":true
                         };

      var rect = grid.getLastGrid().get(0).getBoundingClientRect();

      var rects = [];
      for (var i=0; i<3; i++) {
        for (var j=0; j<3; j++) {
          rects.push(grid.getLastRows().eq(i).children().eq(j).get(0).getBoundingClientRect());
        }
      }
      if (birchlabs.flyoutsOn) {
```

```
  var buckets =getAllClickables↙
    ↪ InBoundsAndSort($(doc2.body↙
    ↪ ), rect,rects);

 // which element was selectedby ↙
   ↪ each segment
 var selected = [];
//console.log(crosshaired);

 var bucketIndex = 0;
 // one bucket for each segment
 for (var i=0; i<3; i++) {
  for (var j=0; j<3; j++) {
   var fulfilled = false;
   for (var b=0;b<buckets.length;↙
     ↪  b++) {
    // check everything in my↙
      ↪ allocated bucket first
    // but be prepared tocycle ↙
      ↪ round to another bucket↙
      ↪ !
    var myBucket = buckets[(b↙
      ↪ ucketIndex+b)%↙
      ↪ buckets.length];
    for (var n = 0;↙
      ↪ n<myBucket.length; n++)↙
      ↪  {
     var inSelectedAlready =↙
       ↪ false;
     if (myBucket[n] !=↙
        ↪ crosshaired) {
        for (var s = 0;↙
          ↪ s<selected.length; ↙
          ↪ s++) {
         // skip over
         if (selected[s] ==↙
           ↪ myBucket[n])
         {
          inSelectedAlready= ↙
            ↪ true;
          break;
         }
        }
        if(!inSelectedAlready) {
         // add to selected
selected[bucketIndex] = myBucket[n];
         fulfilled = true;
         break;
        }
       }
      }
      if (fulfilled) {
       break;
      }
     }
     bucketIndex++;
    }
   }

   for (var i=0; i<flyouts.length;i↙
      ++) {
      var f = flyouts[i];
      var targ = selected[i];
      f.unsetTarget();
      f.hide();
      if (targ) {
        f.setTarget(targ, grid.getFirstGrid().get(0));
      }
    }
//    var delayPoint =setInterval( function () {
//        clearInterval(delayPoint);
        for (var i=0;i<flyouts.length; i++) {
          var f = flyouts[i];
          var targ = selected[i];
          if (targ) {
            f.show();
          }
        }
      pointFlyouts();
    //  }, 50);
    //trace(selected);
  }
}

function unpointFlyouts() {
  if (birchlabs.flyoutsOn) {
    var flyouts = birchlabs.flyouts;

    for (var i=0; i<flyouts.length;i++) {
      var f = flyouts[i];
      f.unsetTarget();
      f.hide();
    }
  }
}

function pointFlyouts() {
  var grid = birchlabs.grid;

  if (birchlabs.flyoutsOn) {
    var flyouts = birchlabs.flyouts;

    for (var i=0; i<flyouts.length;i++) {
      var f = flyouts[i];
f.point(grid.findPointerCoords());
    }
  }
}

function createGrid(root) {
  var numpadMappings = birchlabs.numpadMappings;

  var anticipatedHeight =$(root).
```

```
    height();
  var cellHeight =anticipatedHeight/3;
  // do not display text smallerthan 12px
  // which on a low-density monitoris 12pt

    var parent =document.createElement('div');
    parent.className ='chibiPoint_grid chibiPoint_gridBorderOn';

    var drawBorders = false;
    // if it's not too small, giveit borders
    var rect = $(root).get(0).getBoundingClientRect();
    if (rect.width>60 &&rect.height >60) {
      drawBorders = true;
    }

    for (var i = 0; i < 3; i++) {
        var row =document.createElement('div');
        row.className ='chibiPoint_row ';
        for (var p = 0; p < 3; p++) {
            var cell =document.createElement('div');
            if (drawBorders) {
              cell.className ="chibiPoint_cell chibiPoint_cellBorderOn ";
            } else {
              cell.className ="chibiPoint_cell";
            }

            if (cellHeight>12) {
              cell.innerHTML = String.fromCharCode(numpadMappings[(7-(i*3-p)).toString()][0]).toUpperCase();
              cell.style.fontSize =Math.min(96,cellHeight*0.5)+"px"
              var trans =document.createElement('div');
              trans.className ="chibiPoint_backing";
              cell.appendChild(trans);
            }
            row.appendChild(cell);
        }
        parent.appendChild(row);
    }

    //root.appendChild(parent);
  $(root).append(parent);
}

function drill(num, grid,crosshairs) {
  var Flyout = birchlabs.Flyout;

  grid.getLastCells().text("");
  grid.getLastGrid().addClass("chibiPoint_gridBorderOff");
  grid.getLastCells().addClass("chibiPoint_cellBorderOff");
  grid.getLastGrid().removeClass("chibiPoint_gridBorderOn");
  grid.getLastCells().removeClass("chibiPoint_cellBorderOn");

  var rowIndexed1 =4-Math.ceil(num/3);
  var columnIndexed1 = ((num-1)%3)+1;
  //cell.innerHTML = 7-(i*3-p);

  selectArray = [];
  selectArray.push(grid.getLatestSelector());
selectArray.push(".chibiPoint_grid");
  selectArray.push(".chibiPoint_row:nth-of-type("+rowIndexed1+")");
selectArray.push(".chibiPoint_cell:nth-of-type("+columnIndexed1+")");
grid.pushHistory(selectArray.join(""));

  createGrid($(grid.getLatestSelector()).first());

  // move crosshairs
  crosshairs.updatePosition(grid);

  highlightTarget();
  Flyout.show();

}

function backup(grid, crosshairs) {
  var numpadMappings = birchlabs.numpadMappings;
  var Flyout = birchlabs.Flyout;

  if (gridIsInUse()) {

    // remove a grid!
$(grid.getLatestSelector()).empty();
    grid.getPreviousSelector();

    //trace(grid.getLatestSelector());

    var anticipatedHeight =$(grid.getLatestSelector()).height();
    var cellHeight =anticipatedHeight/3;
    grid.getLastRows().each(function(index) {
      var i = index;
      $(this).children().each(function(index) {
        var p = index;
        if (cellHeight>12) {
          $(this).text(String.fromCharCode(numpadMappings[(7-(i*3-p)).toString()][0]).toUpperCase());

          var trans =document.createElement('div');
          trans.className ="chibiPoint_backing";
          $(this).append(trans);
        }
      });
    });

    grid.getLastGrid().removeClass("chibiPoint_gridBorderOff");
    grid.getLastCells().removeClass("chibiPoint_cellBorderOff");
    grid.getLastGrid().addClass("chibiPoint_gridBorderOn");

    // if it's not too small, giveit borders
    var rect = $(grid.getLatestSelector()).get(0).getBoundingClientRect();
    if (rect.width>60 &&rect.height >60) {
      grid.getLastCells().addClass("chibiPoint_cellBorderOn");
    }

    unpointFlyouts();

    var exists =grid.getLastGrid().length>0;
    if (exists) {
      // move crosshairs
      crosshairs.updatePosition(grid);
      Flyout.show();
      highlightTarget();
    } else {
      closeGrid();
    }
  }
```

```
}

function clickOrFocus(element) {
  evaler.clickOrFocus(element);
  closeGrid();
}

function gridclick(grid) {
  var coords =grid.findPointerCoords();
  var centerX = coords.centerX;
  var centerY = coords.centerY;

  var rootNodes = [window].concat(lookup.getRootNodes());
  for (var i = 0; i <rootNodes.length; i++) {
    var doc2 =rootNodes[i].document ||rootNodes[i].contentDocument;
    if (!doc2 || !doc2.body)
      continue;


    var element =$(doc2.elementFromPoint(centerX,centerY));

    element.removeClass("chibiPoint_targeted");
element.addClass("chibiPoint_cluck");

    var flash =setInterval(function() {
      if (element.hasClass("chibiPoint_cluck")) {
        element.removeClass("chibiPoint_cluck");
      } else {
element.addClass("chibiPoint_cluck");
      }
    }, 100);

    var delayUnclick =setInterval(function () {
        clearInterval(flash);
        clearInterval(delayUnclick);
        element.removeClass("chibiPoint_cluck");
    clickOrFocus(element.get(0));
      pointFlyouts();
      }, 500);
    //element.get(0).click();


  }
}

function shortcut() {
  if (!birchlabs.evaluateTabMode) {
    toggleGrid();
    evaler.evaluatorIncrementKeys("Cmd_K");
  }
}

function toggleGrid() {
  if (!birchlabs.evaluateTabMode) {
    var Flyout = birchlabs.Flyout;

    var crosshairs = birchlabs.crosshairs;
    var container = birchlabs.container;
    var grid = birchlabs.grid;

    document.activeElement.blur();
//document.documentElement.focus();

    if (gridIsEmpty()) {
      // need to make a grid
      grid.initialize();
      $(".chibiPoint_gridContainer").removeClass("chibiPoint_hiddenGridContainer");
      createGrid(container);

      //crosshairs =crosshairs||new birchlabs.Crosshairs(document.documentElement, grid);
      //birchlabs.crosshairs = crosshairs;

      // in case crosshairs already instantiated
      crosshairs.updatePosition(grid);
      Flyout.show();
      highlightTarget();
      crosshairs.show();
    } else {
      // toggle grid off
      closeGrid();
    }
  }
}

function closeGrid() {
  var Flyout = birchlabs.Flyout;

  var crosshairs = birchlabs.crosshairs;
  var grid = birchlabs.grid;

  grid.initialize();
  $(grid.getLatestSelector()).empty();

  $(".chibiPoint_gridContainer").addClass("chibiPoint_hiddenGridContainer"
);

  unpointFlyouts();
  Flyout.hide();
  crosshairs.hide();
}

function gridIsEmpty() {
  return $(".chibiPoint_gridContainer").children().length == 0;
}

function gridIsInUse() {
  //var grid = birchlabs.grid;
  return $(".chibiPoint_grid").children().length > 0;
}

  return { init: init,
           createGrid: createGrid,
         shortcut: shortcut};
});
```

0.6 ./src/Grid.js

```
define(["lib/jquery-2.1.0.min","lib/within"], function(jq, within) {

window.birchlabs = window.birchlabs||{};
var birchlabs = window.birchlabs;

(function(){
  var Grid = function() {
    this.initialize();
  }
  var p = Grid.prototype;

  p.initialize = function() {
      this.selectHistory =[".chibiPoint_gridContainer"];
  };

  p.getLatestSelector = function() {
    return this.selectHistory[this.selectHistory.length-1];
  }

  p.getPreviousSelector = function() {
    if (this.selectHistory.length>1) {
      this.selectHistory.pop(this.selectHistory.length-1);
    }
    return this.getLatestSelector();
  }

  p.pushHistory = function(input) {
    this.selectHistory.push(input);
  }

  p.exists = function() {
    return this.selectHistory.length>1;
  }

  p.getLastRows = function() {
    return$(this.getLatestSelector()+".chibiPoint_row");
  };

  p.getLastCells = function() {
    return$(this.getLatestSelector()+".chibiPoint_cell");
  };

  p.getLastGrid = function() {
    return$(this.getLatestSelector()+".chibiPoint_grid").first();
  };

  p.getFirstGrid = function() {
    return $(this.selectHistory[0]+".chibiPoint_grid").first();
  };

  p.findPointerCoords = function() {
    var element = this.getLastGrid();
    // maybe there are no grids made
    if (element.get(0)) {
     // click
     //var offset =gridElement.offset();
     var offset = element.get(0).getBoundingClientRect();
     //console.log($this.get(0));
     //console.log($this.offset());
     var width = element.outerWidth();
     var height = element.outerHeight();

     var centerX = offset.left +width / 2;
     var centerY = offset.top +height / 2;

     //console.log("x: "+centerX+" y:"+centerY);
     return {centerX:centerX, centerY:centerY};

    } else {
     return null;
    }
  };

  birchlabs.Grid = Grid;
})();

});
```

0.7 ./src/Crosshairs.js

```
define(["lib/jquery-2.1.0.min","lib/within"], function(jq, within) {
```

```

window.birchlabs = window.birchlabs||{};
var birchlabs = window.birchlabs;

(function() {
  var Crosshairs = function(root,grid) {
    this.initialize(root, grid);
  };

  var p = Crosshairs.prototype;

  p.setX = function(x) {
    this.x = x;

    // convert to percent
    var elem = this.grid.getFirstGrid();
    var rect =elem.get(0).getBoundingClientRect();
    var pX = x*100/rect.width;
    $(".chibiPoint_verticalHair").first().css({left:pX+"%"});
  };

  p.setY = function(y) {
    this.y = y;

    // convert to percent
    var elem = this.grid.getFirstGrid();
    var rect =elem.get(0).getBoundingClientRect();
    var pY = y*100/rect.height;

    $(".chibiPoint_horizontalHair").first().css({top:pY+"%"});
  };

  p.show = function() {
    this.hairs.className ="chibiPoint_crosshairs chibiPoint_shownCrosshairs";
  };
  p.hide = function() {
    this.hairs.className ="chibiPoint_crosshairs chibiPoint_hiddenCrosshairs";
    this.removeHighlights();
  };

  p.createCrosshairs = function(root) {
    this.hairs =document.createElement('div');
    this.hairs.className ="chibiPoint_crosshairs";

    this.horizontalHair =document.createElement('div');
    this.horizontalHair.className ="chibiPoint_horizontalHair";

    this.verticalHair =
      document.createElement('div');
  this.verticalHair.className ="chibiPoint_verticalHair";

  $(this.hairs).append(this.horizontalHair);
  $(this.hairs).append(this.verticalHair);

  $(root).append(this.hairs);
};

p.updatePosition = function() {
  var coords =this.grid.findPointerCoords();
  if (coords) {
   var centerX = coords.centerX;
   var centerY = coords.centerY;

   this.setX(centerX);
   this.setY(centerY);
  }
};

p.highlight = function(root) {
  var coords =this.grid.findPointerCoords();
  if (coords) {
   var centerX = coords.centerX;
   var centerY = coords.centerY;

   var element =$(root.elementFromPoint(centerX,centerY));

   // unhighlight previous element
   this.removeHighlights();
   this.targetedElement = element;

   element.addClass("chibiPoint_targeted");

   // doesn't seem to work :(
   /*element.trigger('mouseover');
   element.trigger('hover');
   element.trigger('mouseenter');
   element.mouseover();
   element.hover();
   element.mouseenter();*/
   //element.get(0).mouseover();
   //element.get(0).hover();

   return element;
  } else {
   return null;
  }
};

p.removeHighlights = function() {
  if (this.targetedElement) {
       this.targetedElement.removeClass("chibiPoint_targeted");
       this.targetedElement.removeClass("chibiPoint_cluck");
     }
 };

  p.initialize = function(root, grid) {
    this.targetedElement = null;
    this.createCrosshairs(root);
    this.grid = grid;
    this.updatePosition();
    this.show();
  };

  birchlabs.Crosshairs = Crosshairs;
})();

});
```

0.8 ./src/Flyout.js

```
define(["lib/jquery-2.1.0.min","lib/within", "evaluator"],function(jq, within, evaler) {

window.birchlabs = window.birchlabs||{};
var birchlabs = window.birchlabs;

(function() {
var Flyout = function(root, label) {
  this.initialize(root, label);
};

Flyout.latestId = 0;

Flyout.getNextId = function() {
  return Flyout.latestId++;
};

Flyout.getContainer = function(root) {
 return this.container||this.makecontainer(root);
};

Flyout.makecontainer =function(root) {
  this.container =document.createElement('div');
  this.container.className ="flyoutContainer hiddenfC";

  $(root).append(this.container);
  return this.container;
};

var p = Flyout.prototype;

p.setX = function(x) {
  var rect = Flyout.getContainer().getBoundingClientRect();
  x = Math.min(Math.max(2, x), rect.width-22);
  this.x = x;

  // convert to percent
  var pX = x*100/rect.width;
  $(this.flyout).css({left:pX+"%"});
  //$(this.flyout).css({left:x});
  //$(this.flyout).css({left:x});
};

p.setY = function(y) {
  var rect = Flyout.getContainer().getBoundingClientRect();
   y = Math.min(Math.max(2, y), rect.height-22);
   this.y = y;

   // convert to percent
   var pY = y*100/rect.height;
   $(this.flyout).css({top:pY+"%"});
   //$(this.flyout).css({left:x});
 };

/*p.setY = function(y) {
  this.y = y;

  // convert to percent
  var elem = this.grid.getFirstGrid();
  var rect =elem.get(0).getBoundingClientRect();
  var pY = y*100/rect.height;

  $(".horizontalHair").first().css({top:pY+"%"});
};*/

Flyout.show = function() {
  Flyout.getContainer().className ="chibiPoint_flyoutContainer chibiPoint_shownfC";
  //this.hairs.className ="crosshairs shownCrosshairs";
};
Flyout.hide = function() {
 Flyout.getContainer().className ="chibiPoint_flyoutContainer chibiPoint_hiddenfC";
  //this.hairs.className ="crosshairs hiddenCrosshairs";
};

p.hide = function() {
 $(this.aFlyout).addClass("chibiPoint_aFlyoutHidden");
 //$(this.aFlyout).removeClass("birchAFlyoutShown");
};

p.show = function() {
 $(this.aFlyout).removeClass("chibiPoint_aFlyoutHidden");
};

p.build = function(label) {
  var container =Flyout.getContainer(this.root);

  this.aFlyout =document.createElement('div');
  this.aFlyout.className ="chibiPoint_aFlyout";

  this.lineDiv =document.createElement('div');
  this.lineDiv.className ="chibiPoint_lineDiv";
  this.svg =document.createElement('svg');
  this.svg.className ="chibiPoint_lineSvg"
  this.svgLine =document.createElement('line');
  this.svgLine.className ="chibiPoint_line chibiPoint_lineUnique"+this.unique;
  this.svgLine.id ="chibiPoint_flyoutLine"+this.unique;
 //svgLine.style="stroke:rgb(255,0,0);stroke-width:2";
  $(this.svgLine).attr({x1:"0",
                  y1:"0",
                  x2:"200",
                  y2:"200"});
  //$(svgLine).attr({"data-bind":"attr: {x1:x1,y1:y1,x2:x2,y2:y2}"});

  $(this.svg).append(this.svgLine);

  $(this.lineDiv).append(this.svg);

  this.flyout =document.createElement('div');
  this.flyout.className ="chibiPoint_flyout chibiPoint_flyoutUnique"+this.unique;
  this.flyout.innerHTML = String.fromCharCode(label).toUpperCase();

  $(this.aFlyout).append(this.flyout);
$(this.aFlyout).append(this.lineDiv);

  $(container).append(this.aFlyout);

 // refresh namespace
```

```
  $(this.lineDiv).html($(this.lineDiv).html());
};

/*p.updatePosition = function() {
  var coords = this.grid.findPointerCoords();
  var centerX = coords.centerX;
  var centerY = coords.centerY;

  this.setX(centerX);
  this.setY(centerY);
};*/

// root required only if it's thefirst and you haven't done a makeContainer yet
p.initialize = function(label, root) {
  this.root = root;
  this.unique = Flyout.getNextId();
  this.build(label);
  this.setX(this.unique*20);
  this.setY(this.unique*20);

  //this.updatePosition();
  //this.show();
};

p.point = function(avoidCoords) {
if (this.target) {
  var rect = this.target.getBoundingClientRect();
  var left = rect.left;
  var top = rect.top;
  var width = rect.width;
  var height = rect.height;

  var me = $(Flyout.getContainer());

 // adapted from within.js
 var myRect =me.get(0).getBoundingClientRect();
 var res = !( (myRect.top >top+height) || (myRect.top+myRect.height < top) || (myRect.left> left+width ) ||(myRect.left+myRect.width < left));

   if(res) {
    this.show();
    // disperse away from grid, assuming it exists
    var xdist = 0;
    var ydist = 0;
    var dist = 0;
    var angle = 0;
    if (avoidCoords) {
     xdist = avoidCoords.centerX-rect.left;
```

```
       ydist = avoidCoords.centerY-rect.top;
       dist =Math.sqrt(xdist*xdist+ydist*ydist);
       angle = Math.atan2(ydist, xdist);
      }
      var avoidDist = Math.min((1/(dist*dist*dist))*100000000, 60);

      this.setX((rect.left-20)-Math.sin(angle)*avoidDist);
      this.setY((rect.top-20)+Math.cos(angle)*avoidDist);
      var theX = left;
      var theY = top;

       $("#"+this.svgLine.id).attr({x1:this.x,
y1:this.y,
x2:theX,
y2:theY});
      this.paintTarget();
     } else {
      this.hide();
      this.unpaintTarget();
     }
  } else {
   this.hide();
  }
 };
```

```


  p.clickOrFocus = function(element) {
    evaler.clickOrFocus(element);
  }

 p.doClick = function() {
  if (this.target) {

   var element = $(this.target);
   $(this.target).addClass("chibiPoint_cluck");

   var flash = setInterval(function() {
     if (element.hasClass("chibiPoint_cluck")) {
       element.removeClass("chibiPoint_cluck");
     } else {
element.addClass("chibiPoint_cluck");
     }
   }, 100);

   var toClick = this.clickOrFocus;
   var thatTarget = this.target;
   var delayUnclick =setInterval(function () {
      clearInterval(flash);
      clearInterval(delayUnclick);
      $(thatTarget).removeClass("chib
    iPoint_cluck");
    toClick(thatTarget);
   }, 500);
 }
};

p.paintTarget = function() {
 if (this.target) {
  $(this.target).addClass("chibiPoint_painted");
  $(this.target).addClass("chibiPoint_paintedUnique"+this.unique);
 }
};

p.unpaintTarget = function() {
 if (this.target) {
  $(this.target).removeClass("chibiPoint_painted");
  $(this.target).removeClass("chibiPoint_paintedUnique"+this.unique);
 }
};

p.unsetTarget = function() {
 this.unpaintTarget();
 this.target = false;
};

p.getTarget = function() {
  return this.target;
};

p.setTarget = function(element, coordsyst) {
 /*if (element == null) {
  $("#"+this.svgLine.id).attr({x2:0,
                   y2:0});
  return;
 }*/
 this.unsetTarget();
 this.target = element;
 //this.point();

 /*var rect2 = coordsyst.getBoundingClientRect();
 var pX = theX*100/rect2.width;
 var pY = theY*100/rect2.height;*/

 /*$(this.svgLine).attr({x2:theX,
                   y2:theY});
 $(this.svgLine).attr({id:"hey"});
   //$(this.lineDiv).html($(this.lineDiv).html());
 $(this.lineDiv).html($(this.lineDiv).html());*/
};
```

```
    birchlabs.Flyout = Flyout;
})();

});
```