Make your own *Print & Play* card game using SVG and JavaScript

Kevin Hakanson

5 April 2014

Abstract

Want to leverage your creativity, love of board games, and web platform experience to do something different? Turn your imagination into a Print & Play card game using only a modern web browser, color printer and text editor.

This session will use the Scalable Vector Graphics (SVG) image format and JavaScript programming language to make a deck of cards for a simple game. Creating a few cards in graphics software like Inkscape is one thing, but what about 50 or 100 cards? What happens when you need to update them all? That's the value of generating your SVG using JavaScript.

We will start with a blank screen, adding color and graphics elements like lines, shapes, text and images. Learn about container elements and defining content for re-use. Understand how units in the SVG coordinate system can transform our on-screen creation into an 8.5 by 11 inch printed page (or PDF). SVG examples will be both in their native XML format and created from JavaScript using Snap.svg, an open source library from Adobe designed for modern web browsers.

You will leave this session with a basic knowledge of SVG concepts, how to programmatically generate SVG using JavaScript, and how to make your SVG creation printer friendly.

TL;DR

- *Print & Play* Card Game
  - Scalable Vector Graphics (SVG)
  - JavaScript
  - Snap.svg
  - 8.5 by 11 Printing

Kevin Hakanson

@hakanson
#tccc16

kevin.hakanson@gmail.com
+KevinHakanson

https://github.com/hakanson

http://stackoverflow.com/users/22514/kevin-hakanson

Bio

Kevin Hakanson is an application architect for Thomson Reuters where he is focused on highly scalable web applications, especially the JavaScript and security aspects. His background includes both .NET and Java, but he is most nostalgic about Lotus Notes. He has been developing professionally since 1994 and holds a Master's degree in Software Engineering. When not staring at a computer screen, he is probably staring at another screen, either watching TV or playing video games with his family.

Bio

Kevin Hakanson is an application architect for Thomson Reuters where he is focused on highly scalable web applications, especially the JavaScript and security aspects. His background includes both .NET and Java, but he is most nostalgic about Lotus Notes. He has been developing professionally since 1994 and holds a Master's degree in Software Engineering. When not staring at a computer screen, he is probably staring at another screen, either watching TV or playing video games with his family.

YouTube

Geek & Sundry
EST 2012


HOW to PLAY
STAR FLUXX
OVERVIEW
SETUP
GOING FIRST
HOW TO PLAY
CARD TYPES
GOAL
KEEPERS
How to Begin
THE EVER-CHANGING CARD GAME...IN SPACE!
STAR FLUXX
FLUXX
THAT'S NO MOON
The player who has the Small Moon and the Space Station on the table wins.
NEW
WORMHOLE
To play this card, place it face up in the center of the table. New rules take effect
Wormhol
Once per turn, you take the top card fr draw pile and imm play it. Repeat if y a Creeper.
This does not coun of your Draw or Pla
EXPENDABLE CREWMAN
KEE
To play this card up on the table in fr
Expendabl Crewman
Any time any player discards or takes away one of your Keepers, they must take this one instead (if you have it on the table).
BRAIN PARASITE
ard to replace it.
Brain Parasites
You can't win if you have this unless the Goal says otherwise.
If you have Keepers with brains in play, you must choose one to attach this to.
Both cards stay together until discarded.
IT'S A TRAP
This card can be played at any time, for one of the functions described below, or to cancel a Surprise which another player has just played.
It's a Trap!
Out of turn: Cancel any single game action in which another player is stealing something from you, and instead you take that action against them.
During your turn: All other players must discard a card, while you draw 2.

DRAW

DISCARD

FLUXX

STAR FLUXX

HOW TO PLAY

Deal 3 cards to each person

ON YOUR TURN:

1. Draw the number of cards currently required
2. Play the number of cards currently required
3. Discard down to the current Hand Limit (if any)
4. Comply with the current Keeper Limit (if any)

WINNING:

The game ends when a player has the keepers in play that match the current goal. That player is the winner.

LOO-615-04 LooneyLabs.com

PLAYER 1 KEEPERS

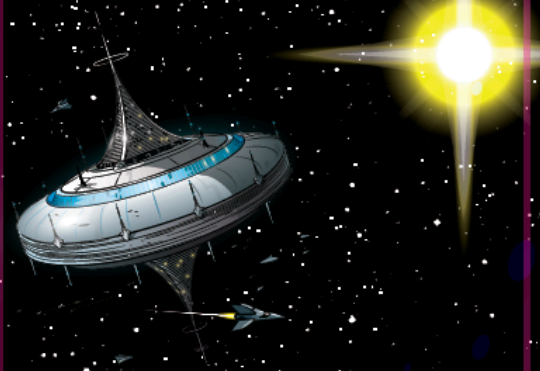
FLUXX BASIC RULES

To begin, shuffle up the deck and deal 3 cards to each player. Choose someone to go first. Keep this card on the table.

DRAW 1 * 1 **Draw 1** Draw 1 card per turn.

PLAY 1 * 1 **Play 1** Play 1 card per turn.

PLAYER 3 KEEPERS



GOALS

RULES

PLAYER 2 KEEPERS


STAR FLUXX
BASIC RULES
NEW RULE
CREEPER
KEEPER
The Robot
Expendable Crewman
The Captain
FLUXX


THE CARD GAME WITH EVER-CHANGING RULES!
FLUXX


THE CARD GAME THAT WILL DRIVE YOU INSANE!
CTHULHU FLUXX


A WONDERFUL WHIRLWIND OF FUN!
OZ FLUXX


THE EVER-CHANGING CARD GAME...IN SPACE!
STAR FLUXX


THE EVER-CHANGING PIRATE CARD GAME!
PIRATE FLUXX


ARE YOU READY TO CONQUER THE EARTH?
MARTIAN FLUXX


THE EVER-CHANGING MONTY PYTHON CARD GAME
MONTY PYTHON FLUXX


THE EVER-CHANGING ZOMBIE CARD GAME
ZOMBIE FLUXX
From the dark side of Looney Labs


THE NATURE GAME WITH EVER-CHANGING RULES!
ECO FLUXX


THE EVER-CHANGING FAMILY CARD GAME!
FAMILY FLUXX

More Fun From Looney Labs®

$1.00

INTERNATIONAL

TABLETOP DAY

LOONEY LABS

TIE-DYE BONUS *

NEW RULE

To play this card, place it face up in the center of the table. New rules take effect instantly.

Tie-Dye Bonus

If you are wearing tie-dyed clothing of some kind, Draw and Play 1 extra card per turn.

Hastur LOO-???-??

Fluxx Blanxx

- Add your own zany ideas
- Each pack contains one of each of the five card types
- Cards are "halfway-blank" with standard-issue text and the stripe of color
- Just grab your trusty permanent marker and customize the fun!

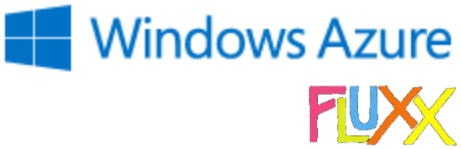
http://store.looneylabs.com/Fluxx-Blanxx

Print & Play

- "*Print & Play* games are those which are often free to any player who wishes to print them off themselves. Many are available on the Internet."
  - http://boardgamegeek.com/boardgamecategory/1120/print-play

- *Print & Play* games Boardgamegeek wiki entry:
  - http://boardgamegeek.com/wiki/page/Print_and_Play_Games

Azure Fluxx



- Version of Fluxx based in the Window Azure “universe”:
  - Software Development
  - Microsoft Products
  - Web Technologies

Fair Use?

- Fluxx Copyright Looney Labs
- Same card types
- Same basic gameplay and rules wording
- Similar, but not exact, colors and artwork
- Different fonts
- Different physical card size and deck count

Azure Fluxx Cards

JAVASCRIPT

KEEPER

To play this card, place it face up on the table in front of you.

JavaScript

WEB PLATFORM

GOAL

To play this card, place it face up in the center of the table. Discard previous Goal, if any.

Web Platform

The player who has any 2 of the following Keepers on the table wins: HTML5, CSS3, JavaScript

Goals and Keepers

- Goals
  - Old School
  - New School
  - Single Page App
  - Polyglot
  - Web Platform
  - Tool Master
  - .NET Platform
  - Entry Level
  - Mistaken Identity
  - Selectors
  - Transpile
  - Odd Couple
  - Double MVC
  - Intellisense
  - Edge.js
  - UI Bootstrap

- Keepers

Scalable Vector Graphics (SVG)

- Text-based graphics language that describes images with vector shapes, text, and embedded raster graphics.
- SVG files provide resolution independent, high resolution dots per inch (HiDPI) graphics on the web, in print, and on mobile devices in a compact format.

http://www.adobe.com/devnet/svg.html

SVG Essentials

"This insightful book takes you through the ins and outs of SVG, from the basics to more complicated features."

http://commons.oreilly.com/wiki/index.php/SVG_Essentials

SVG Primer

"The book attempts to discuss SVG in broader terms, but at the same time to illustrate how one can write JavaScript programs that use and manipulate SVG."

http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

Snap.svg

“the JavaScript SVG library for the modern web”

“makes working with your SVG assets as easy as jQuery makes working with the DOM”

http://snapsvg.io/

The `svg` Element

- `<svg>` is both root element and used to nest standalone SVG fragments
- Each standalone fragment has its own viewPort and coordinate system

<svg> XML Example

```
<svg
  width="8.5in"
  height="11in"
  viewBox="0 0 215.9 279.4"
  version="1.1"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
</svg>
```

<svg> JavaScript Example

```
var svg = Snap("8.5in", (pageCount*11)+"in")
          .attr({id:"SnapDeck", style:"display: block;"});

var pageAttr = { width:"8.5in", height:"11in",
                 "viewBox" : "0 0 215.9 279.4"};

for (pageIndex = 1; pageIndex<=pageCount; pageIndex++) {

    page = svg.el("svg", pageAttr)
           .attr({ id : "SnapDeckPage"+pageIndex,
                   y : ((pageIndex-1)*11)+"in" });
}
```

<svg> XML Example

```
<svg width="8.5in" height="22in" id="SnapDeck">

  <svg width="8.5in" height="11in" y="0in"
   viewBox="0 0 215.9 279.4" id="SnapDeckPage1">
  </svg>

  <svg width="8.5in" height="11in" y="11in"
   viewBox="0 0 215.9 279.4" id="SnapDeckPage2">
  </svg>

</svg>
```

Grouping and Reusing

"The <use> (reuse) and <g> (or group) tags bear similarity to the variables and objects encountered in programming languages."

http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html#operations

The g Element

- Gathers child elements into a group
- Uses id attribute to give a unique name
- Can have <title> and <desc>
- Styles apply to child elements
- Can nest groups within one another

<g> XML Example

```
<g id="plus"
   opacity=".4"
   stroke="black"
   stroke-width="0.1">

  <path d="M -3 0 H 3 M 0 -3 V 3" />

</g>
```

The `defs` Element

- Putting grouped objects inside of `<defs>` tells SVG to define them without displaying them
- SVG recommendation is to put all objects intended for re-use inside `<defs>`

<defs> JavaScript Example

```
var plus = svg.g(
 svg.path("M -3 0 H 3 M 0 -3 V 3")
);
plus.attr({id:"plus",
   opacity:"0.4",
   stroke:"black",
   "stroke-width":"0.1"});
plus.toDefs();
```

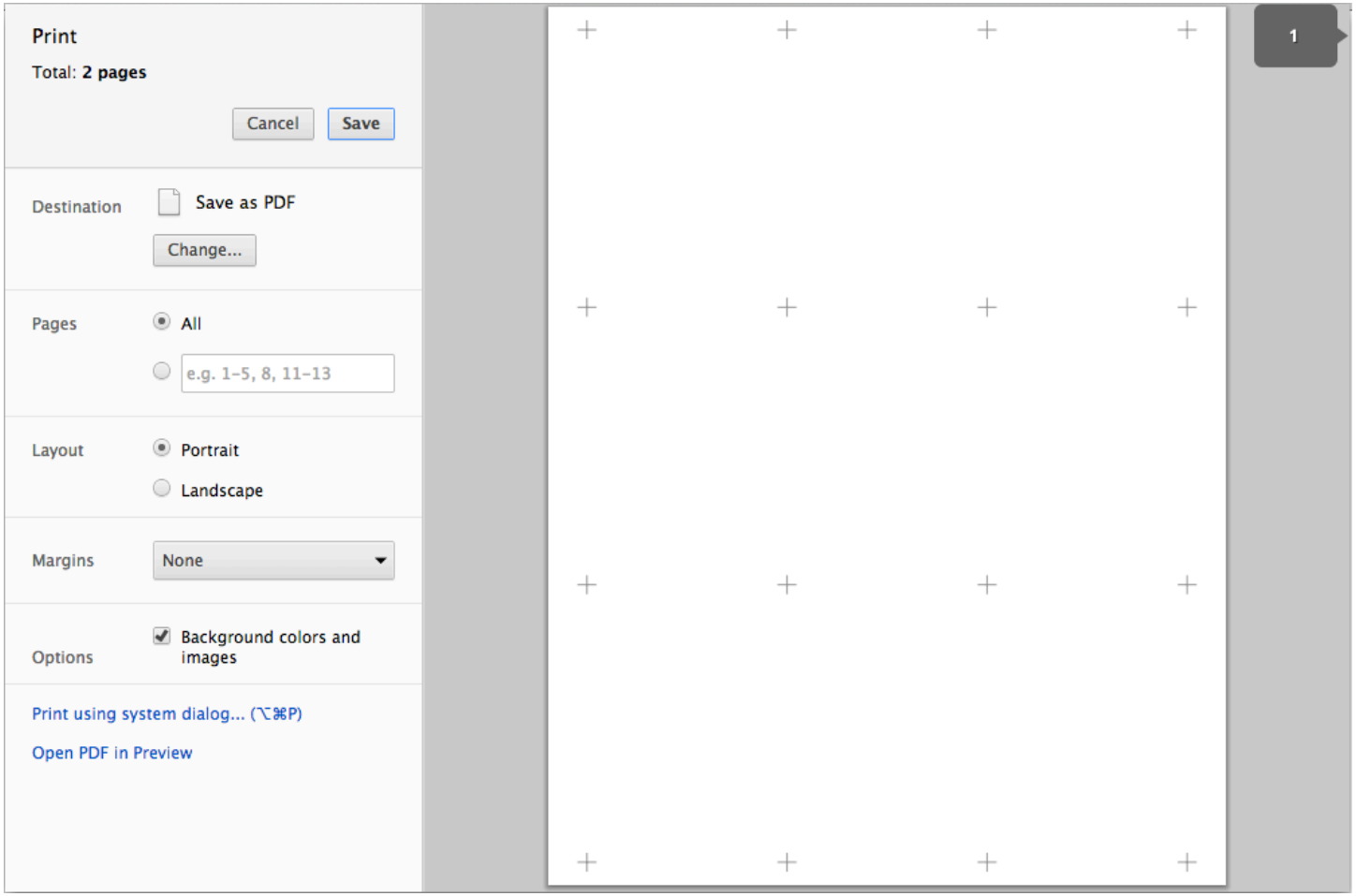
The `symbol` Element

- `<symbol>` element provides another way of grouping elements
- Never displayed, so don't have to enclose it in `<defs>` but customary to do so
- `<symbol>` can specify `viewBox` and `preserveAspectRatio` attributes

The `use` Element

- “copy-and-paste” of a defined group
- Specify with an `xlink:href` attribute
- Specify the `x` and `y` location for the group's `(0, 0)` point

Chrome Print Dialog



```
<g id="pluses">
    <use xlink:href="#plus" x="12.7" y="7.7"/>
    <use xlink:href="#plus" x="12.7" y="95.7"/>
    <use xlink:href="#plus" x="12.7" y="183.7"/>
    <use xlink:href="#plus" x="12.7" y="271.7"/>

    <use xlink:href="#plus" x="76.2" y="7.7"/>
    <use xlink:href="#plus" x="76.2" y="95.7"/>
    <use xlink:href="#plus" x="76.2" y="183.7"/>
    <use xlink:href="#plus" x="76.2" y="271.7"/>

    <use xlink:href="#plus" x="139.7" y="7.7"/>
    <use xlink:href="#plus" x="139.7" y="95.7"/>
    <use xlink:href="#plus" x="139.7" y="183.7"/>
    <use xlink:href="#plus" x="139.7" y="271.7"/>

    <use xlink:href="#plus" x="203.2" y="7.7"/>
    <use xlink:href="#plus" x="203.2" y="95.7"/>
    <use xlink:href="#plus" x="203.2" y="183.7"/>
    <use xlink:href="#plus" x="203.2" y="271.7"/>
</g>
```

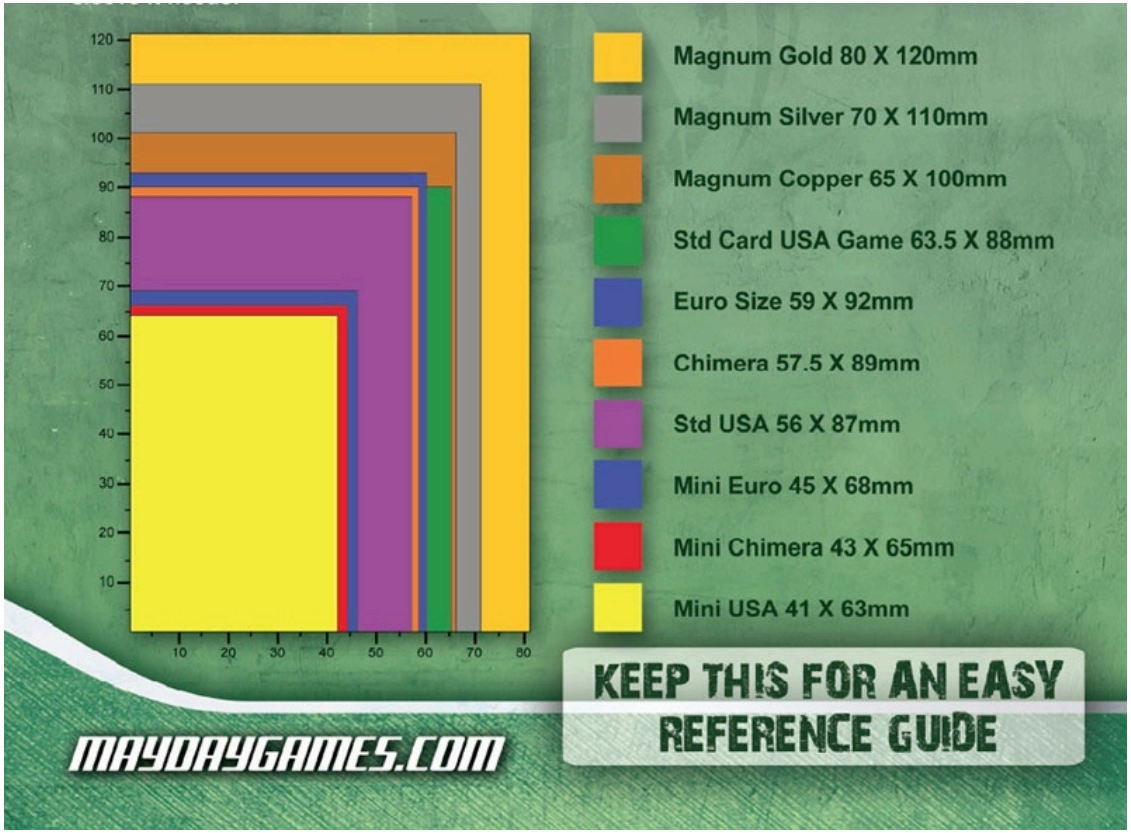
<use> XML Example

```
<g id="pluscolumn">
    <use xlink:href="#plus" y="7.7" />
    <use xlink:href="#plus" y="95.7" />
    <use xlink:href="#plus" y="183.7" />
    <use xlink:href="#plus" y="271.7" />
</g>

<g id="pluses">
    <use xlink:href="#pluscolumn" x="12.7" />
    <use xlink:href="#pluscolumn" x="76.2" />
    <use xlink:href="#pluscolumn" x="139.7" />
    <use xlink:href="#pluscolumn" x="203.2" />
</g>
```

<use> JavaScript Example

```
// width: 63.5, height: 88
var xstops = [12.7, 76.2, 139.7, 203.2];
var ystops = [7.7, 95.7, 183.7, 271.7];

pages.forEach(function (page) {
  ystops.forEach(function (y) {
    xstops.forEach(function (x) {
      var p = plus.use().attr({ x: x, y: y });
      page.append(p);
    });
  });
});
```

Why 63.5 x 88 mm ?




Ultra•PRO
Non-Glare
PRO-MATTE
50
IDEAL FOR
PRO-MATTE
TOURNAMENT USE
• Professional Play MATTE Deck Protector® sleeves
• Non glare / Non reflective
• Easy shuffling
• Prevents bent corners
• No PVC / Acid free
66mm x 91mm
Sized for STANDARD Gaming Cards
• NEW sleeve size for better fit

Paper Cutter

The `rect` Element

- Specify x- and y-coordinates of the upper left corner, width, and height
  - `x`, `y`, `width` and `height` attributes
- Interior is filled with a fill color (default black)
- Outline is drawn with strokes (default none)

```
<rect x="0" y="0"
   width="10" height="10" fill="black"/>
```

The `circle` Element

- Specify center x-coordinate, center y-coordinate, and radius
  - `cx`, `cy`, and `r` attributes
- Interior is filled with a fill color (default black)
- Outline is drawn with strokes (default none)

```
<circle cx="5" cy="5" r="3.5"
   fill="white" stroke-width="0" />
```

The `ellipse` Element

- Like `<circle>`, specify center x-coordinate, center y-coordinate
- Also needs an x-radius and a y-radius
  - attributes for these radii are named `rx` and `ry`

```
<ellipse cx="4" cy="4"
   rx="0.25" ry="0.5" />
```

The `line` Element

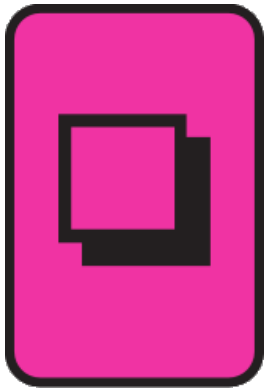
- Specify the x- and y-coordinates of the line's endpoints
  - `x1`, `y1`, `x2` and `x2` attributes

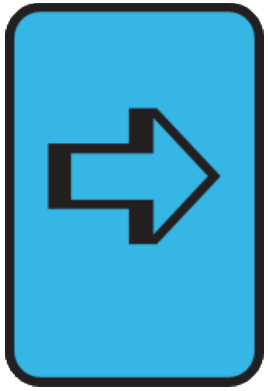
The path Element

- Draws the outline of any arbitrary shape by specifying a series of connected lines, arcs, and curves
- Must begin with a moveto command
- Followed by one or more lineto commands
- Arc draws a section of an ellipse that connects two points

```
<path fill="none" stroke-linecap="round"
      d="M 3.5 6.75 a 1.5,1.5 0 1 1 3 0" />
```

Fluxx Card Types



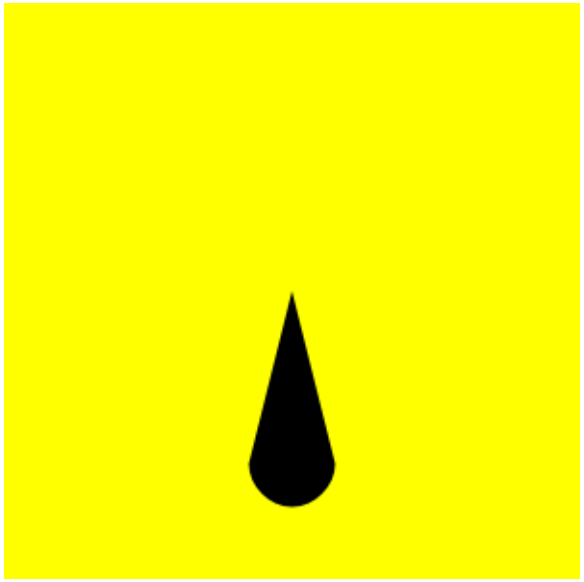


```
<rect x="0" y="0" width="10" height="10" fill="black" />
<g id="frown" stroke="black" fill="black" stroke-width=".5">
  <circle cx="5" cy="5" r="3.5" fill="white" stroke-width="0" />

  <circle cx="4" cy="4" r="0.25" />
  <circle cx="6" cy="4" r="0.25" />

  <path fill="none"
        d="M 3.5 6.75 a 1.5,1.5 0 1 1 3 0" />
</g>
```

```
<rect x="0" y="0" width="10" height="10" fill="black" />
<g id="frown" stroke="black" fill="black" stroke-width=".5">
  <circle cx="5" cy="5" r="3.5" fill="white" stroke-width="0" />

  <ellipse cx="4" cy="4" rx="0.25" ry="0.5" />
  <ellipse cx="6" cy="4" rx="0.25" ry="0.5" />

  <path fill="none" stroke-linecap="round"
        d="M 3.5 6.75 a 1.5,1.5 0 1 1 3 0" />
</g>
```

```
<rect x="0" y="0" width="10" height="10" fill="black" />
<g id="frown" stroke="black" fill="black" stroke-width=".5">
  <circle cx="5" cy="5" r="3.5" fill="white" stroke-width="0" />

  <ellipse cx="4" cy="4" rx="0.25" ry="0.5" />
  <ellipse cx="6" cy="4" rx="0.25" ry="0.5" />

  <path fill="none" stroke-linecap="round"
        d="M 3.5 6.75 C 4 5 6 6 6.5 7" />
</g>
```

```
svg.rect(0, 0, 10, 10);

var frown = svg.g()
    .attr({ id: "frown", stroke: "black",
            fill: "black", "stroke-width": 0.5 });

frown.append(svg.circle(5, 5, 3.5)
    .attr({ fill: "white", "stroke-width": 0 }));
frown.append(svg.ellipse(4, 4, 0.25, 0.5));
frown.append(svg.ellipse(6, 4, 0.25, 0.5));

frown.append(svg.path("M 3.5 6.75 C 4 5 6 6 6.5 7")
    .attr({ fill: "none",
            "stroke-linecap": "round" }));
```



```
<rect x="0" y="0" width="10" height="10" fill="yellow"/>

<g id="flowerpedal" stroke="black" stroke-width="0">
  <path fill="black"
        d="M 5 5 l -0.75 3 a 0.75,0.75 0 1 0 1.5 0 Z" />
</g>
```

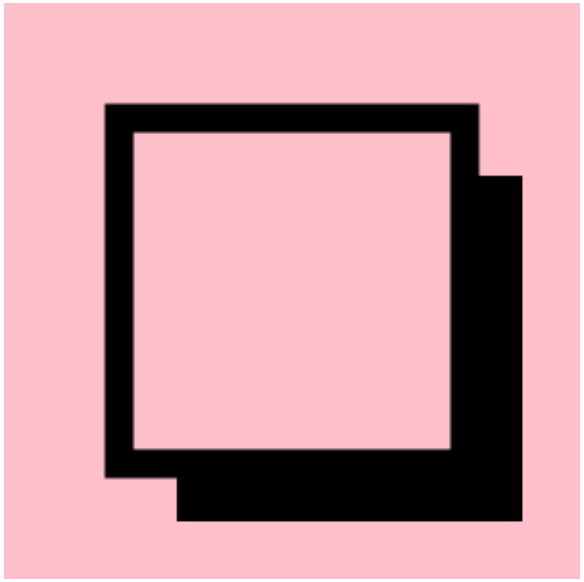
The `transform` Attribute

- matrix(<a> <b> <c> <d> <e> <f>)
- translate(<x> [<y>])
- scale(<x> [<y>])
- rotate(<a> [<x> <y>])
- skewX(<a>)
- skewY(<a>)

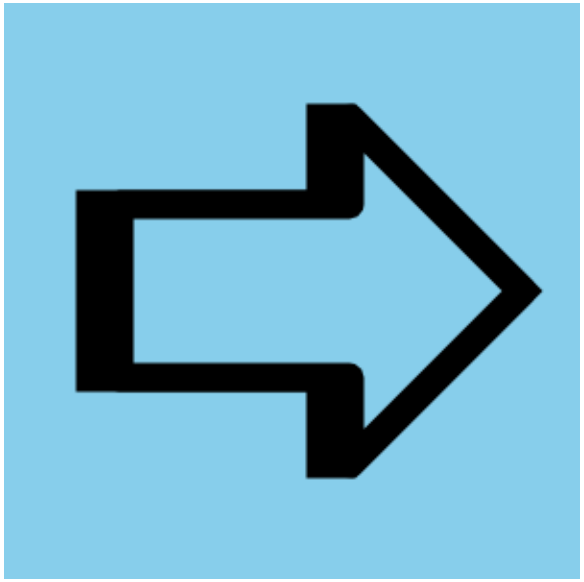
https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/transform

```
<rect x="0" y="0" width="10" height="10" fill="yellow" />
<g id="flowerpedal" stroke="black" stroke-width="0">
  <path fill="black" d="M 5 5 l -0.75 3 a 0.75,0.75 0 1 0 1.5 0 Z" />
</g>
<use xlink:href="#flowerpedal" transform="rotate( 60, 5, 5)" />
<use xlink:href="#flowerpedal" transform="rotate(120, 5, 5)" />
<use xlink:href="#flowerpedal" transform="rotate(180, 5, 5)" />
<use xlink:href="#flowerpedal" transform="rotate(240, 5, 5)" />
<use xlink:href="#flowerpedal" transform="rotate(300, 5, 5)" />
```

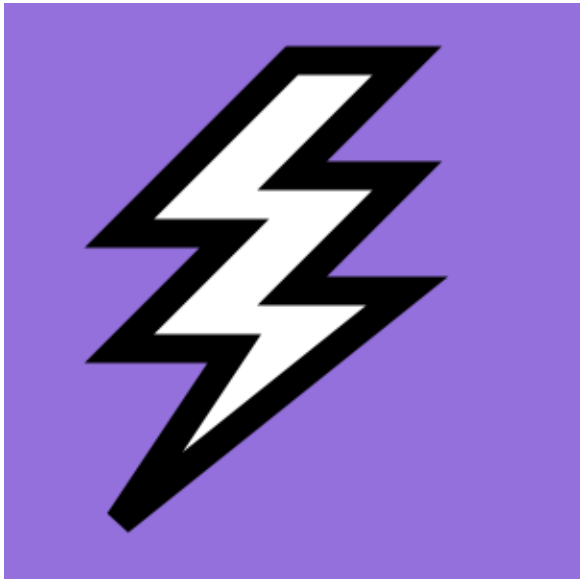
```
<rect x="0" y="0" width="10" height="10" fill="greenyellow"/>
<g id="check" stroke="black" stroke-width="1">
  <polyline stroke-linejoin="round"
            stroke-linecap="round"
            fill="none"
            points="2 5, 3 8, 8 2" />
</g>
```



```
<rect x="0" y="0" width="10" height="10" fill="pink" />
<g id="checkbox" stroke="black">
  <rect x="3" y="3" width="6" height="6"
        fill="black" stroke-width="0" />
  <rect x="2" y="2" width="6" height="6"
        fill="pink" stroke-width="0.5" />
</g>
```

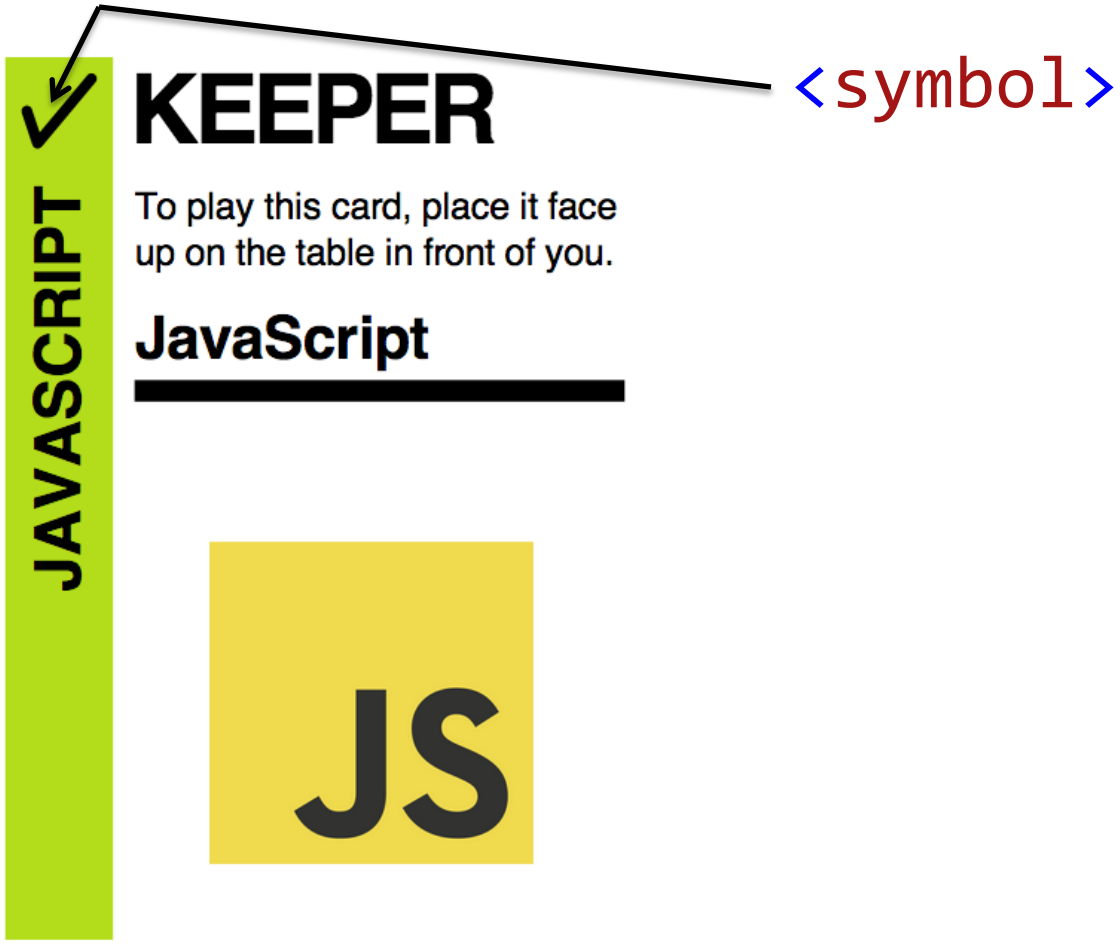


```
<rect x="0" y="0" width="10" height="10" fill="skyblue" />
<g id="arrow" stroke="black" stroke-width="0.5">
  <path fill="black" d="M 5.5 2 l .5 0 l 3 3 l -3 3 l -0.5 0 l 0
   -1.5 l -4 0 l 0 -3 l 4 0 Z" />
  <path stroke-linejoin="round" fill="skyblue" d="M 6 2 l 3 3 l
   -3 3 l 0 -1.5 l -4 0 l 0 -3 l 4 0 Z" />
</g>
```

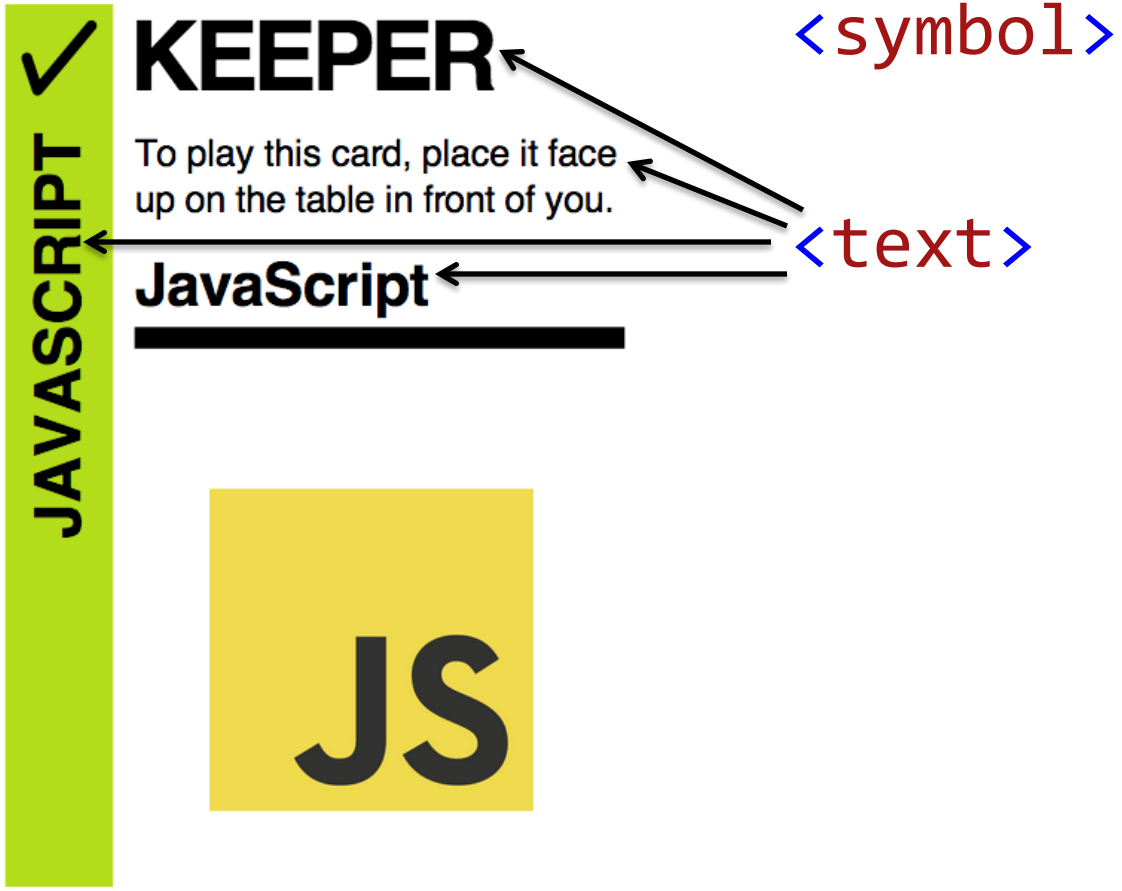


```
<rect x="0" y="0" width="10" height="10" fill="mediumpurple" />

<g id="bolt" stroke="black" stroke-width="0.5">
  <path fill="white" d="M 5 1 l -3 3 l 2 0 l -2 2 l 2 0 l -2 3 l
   5 -4 l -2 0 l 2 -2 l -2 0 l 2 -2 Z" />
</g>
```

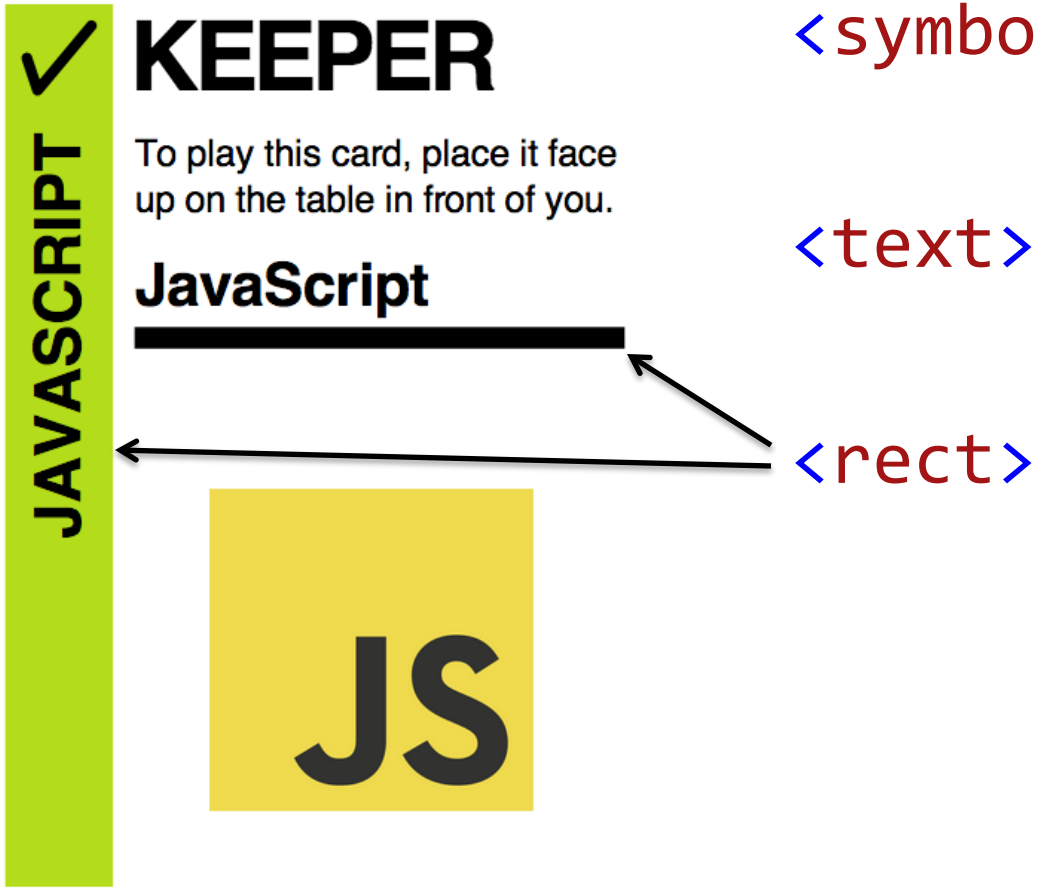
How-To Make a Keeper



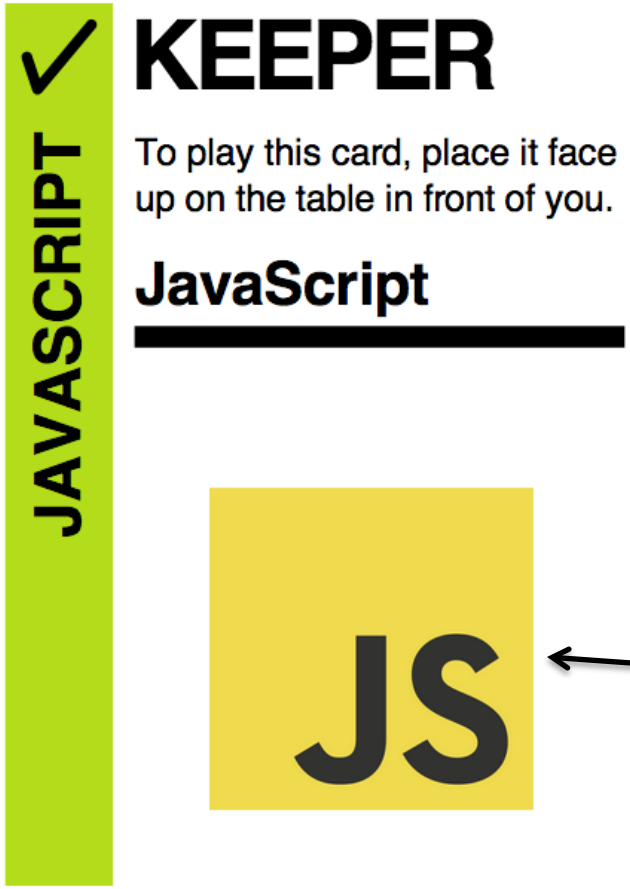
How-To Make a Keeper



How-To Make a Keeper



How-To Make a Keeper



<symbol>

<text>

<rect>

<image>

The `text` Element

- `<text>` requires only two attributes, `x` and `y`
- Default style is to have a fill color of black and no outline
- font-family, font-size, font-weight, font-style, text-decoration, word-spacing, letter-spacing
- "SVG performs no automatic line breaking or word wrapping"
  http://www.w3.org/TR/SVG11/text.html#Introduction

<text> XML Example

```
<g text-anchor="start"
   font-size="3.5"
   letter-spacing="-0.05">

 <text x="15" y="18">To play this card, place it face</text>

 <text x="15" y="22.375">up on the table in front of you.</text>

</g>
```

The image Element

- <image> element includes an entire SVG or raster file (JPEG or PNG)
- x, y, width, and height attributes

```
<image xlink:href="images/JavaScript-logo.png"
preserveAspectRatio="meet" x="22" y="48" width="30"
height="30" />
```

<image> XML Example

```
<image
xlink:href="images/JavaScript-logo.png"
x="22"
y="48"
width="30"
height="30"
        preserveAspectRatio ="meet" />
```

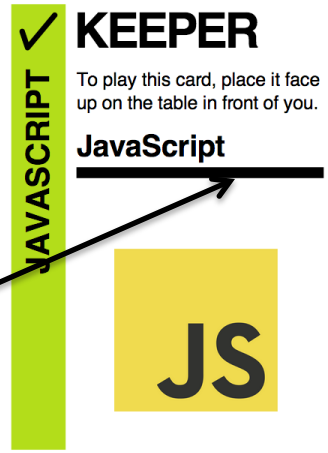
<image> JavaScript Example

```
s.image(
                "images/JavaScript-logo.png",
    22,
    48,
        30,
         30)
.attr({ "preserveAspectRatio":"meet" })
```

```
<svg id="SnapDeckCard1"
 width="63.5" height="88"
 xmlns="http://www.w3.org/2000/svg"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 font-family="sans-serif">

  <defs>

    <!-- ... -->
  </defs>
 <!-- ... -->
</svg>
```

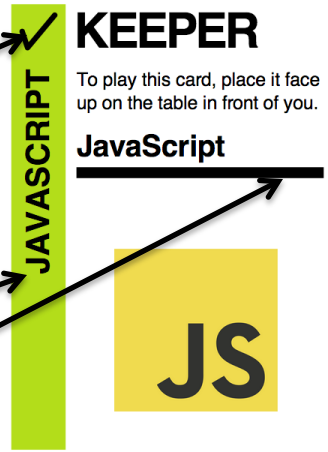
```
<svg id="SnapDeckCard1"
 width="63.5" height="88"
 xmlns="http://www.w3.org/2000/svg"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 font-family="sans-serif">

  <defs>
    <rect id="sidebar" x="3" y="3" width="10" height="82" />

    <!-- ... -->
  </defs>
 <!-- ... -->
</svg>
```

```
<svg id="SnapDeckCard1"
 width="63.5" height="88"
 xmlns="http://www.w3.org/2000/svg"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 font-family="sans-serif">

  <defs>
    <rect id="sidebar" x="3" y="3" width="10" height="82" />
    <rect id="hr" x="15" y="33"
          width="45.5" height="2" fill="black" />

    <!-- ... -->
  </defs>
 <!-- ... -->
</svg>
```

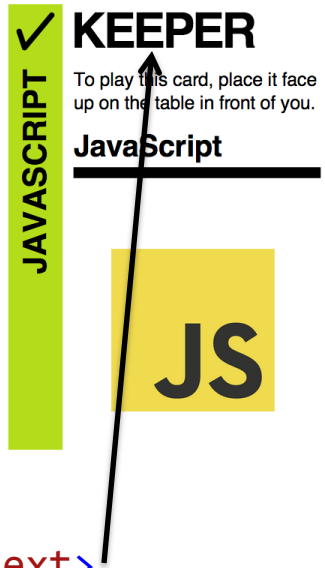


```
<svg id="SnapDeckCard1"
 width="63.5" height="88"
 xmlns="http://www.w3.org/2000/svg"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 font-family="sans-serif">

  <defs>
    <rect id="sidebar" x="3" y="3" width="10" height="82" />
    <rect id="hr" x="15" y="33"
          width="45.5" height="2" fill="black" />

    <symbol id="check" stroke="black" stroke-width="1">
      <polyline points="5,8,6,11,11,5" fill="none"
                stroke-linejoin="round" stroke-linecap="round" />
    </symbol>
    <!-- ... -->
  </defs>
 <!-- ... -->
</svg>
```

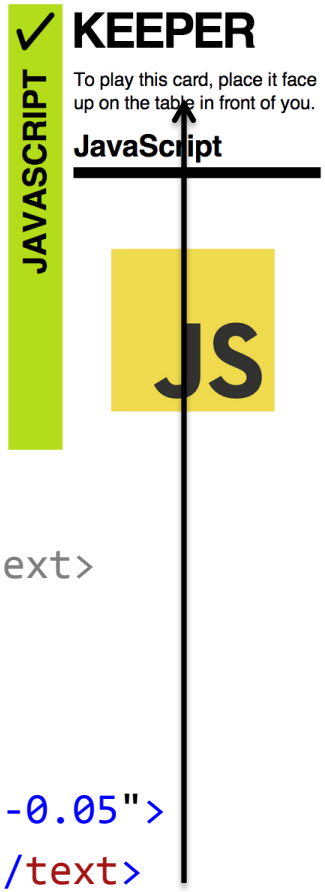
```
<svg>
  <defs>
    <!-- ... -->
    <g id="keeper">
      <use xlink:href="#check" />
      <use xlink:href="#sidebar" fill="#b3dd1a" />

      <use xlink:href="#hr" />

    </g>
  </defs>
 <!-- ... -->
</svg>
```



```
<svg>
  <defs>
    <!-- ... -->
    <g id="keeper">
      <use xlink:href="#check" />
      <use xlink:href="#sidebar" fill="#b3dd1a" />

      <text text-anchor="start" x="14.5" y="11" font-size="9"
            letter-spacing="-.5" font-weight="bold">KEEPER</text>

      <use xlink:href="#hr" />









    </g>
  </defs>
 <!-- ... -->
</svg>
```



```
<svg>
  <defs>
    <!-- ... -->
    <g id="keeper">
      <use xlink:href="#check" />
      <use xlink:href="#sidebar" fill="#b3dd1a" />

      <text text-anchor="start" x="14.5" y="11" font-size="9"
            letter-spacing="-.5" font-weight="bold">KEEPER</text>

      <use xlink:href="#hr" />

      <g text-anchor="start" font-size="3.5" letter-spacing="-0.05">
        <text x="15" y="18">To play this card, place it face</text>
        <text x="15" y="22.375">up on the table in front of you.</text>
      </g>
    </g>
  </defs>
 <!-- ... -->
</svg>
```



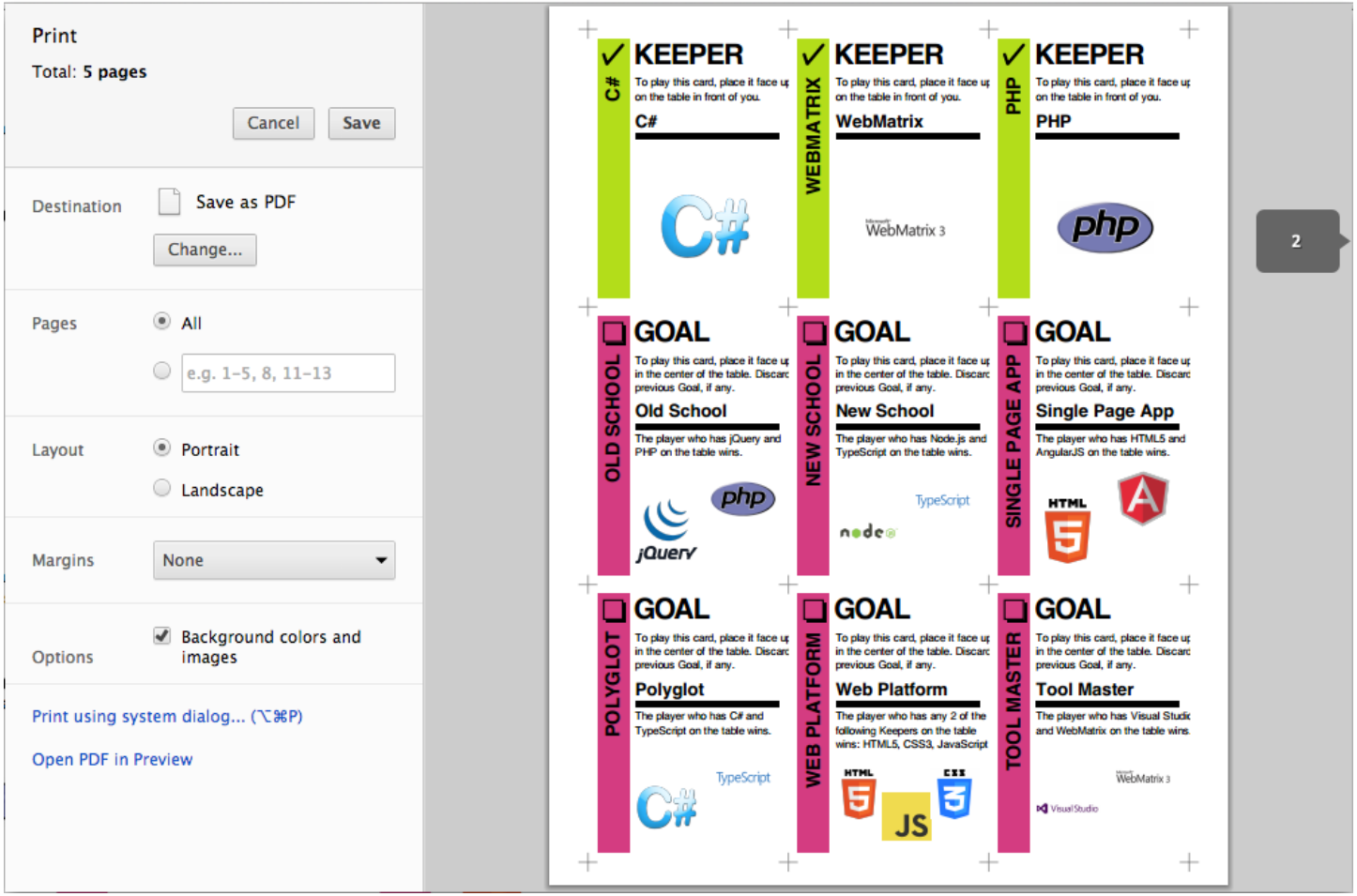
```
<svg>
  <defs>
    <!-- ... -->
  </defs>

  <use xlink:href="#keeper" />

  <g text-anchor="start" font-size="5.5"
     letter-spacing="-0.05" font-weight="bold">
    <text x="15" y="31">JavaScript</text>
  </g>

</svg>
```

```
<svg>
  <defs>
    <!-- ... -->
  </defs>

  <use xlink:href="#keeper" />

  <g text-anchor="start" font-size="5.5"
     letter-spacing="-0.05" font-weight="bold">
    <text x="15" y="31">JavaScript</text>
  </g>

  <text x="10" y="15" transform="matrix(0,-1,1,0,-5,25)" font-size="6"
        text-anchor="end" font-weight="bold">JAVASCRIPT</text>



</svg>
```

```
<svg>
  <defs>
    <!-- ... -->
  </defs>

  <use xlink:href="#keeper" />

  <g line-spacing="0" paragraph-spacing="0" text-anchor="start"
     font-size="5.5" letter-spacing="-0.05" font-weight="bold">
    <text x="15" y="31">JavaScript</text>
  </g>

  <text x="10" y="15" transform="matrix(0,-1,1,0,-5,25)" font-size="6"
        text-anchor="end" font-weight="bold">JAVASCRIPT</text>

  <image xlink:href="images/JavaScript-logo.png"
         preserveAspectRatio="meet" x="22" y="48"
         width="30" height="30" />
</svg>
```

```svg
<svg id="SnapDeckCard1" width="460" height="640" viewBox="0 0 63.5 88" xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink" font-family="sans-serif”>
  <defs>
    <rect x="3" y="3" width="10" height="82" id="sidebar" />
    <rect x="15" y="33" width="45.5" height="2" id="hr" fill="black" />

    <symbol id="check" stroke="black" stroke-width="1">
      <polyline points="5,8,6,11,11,5" fill="none" stroke-linejoin="round" stroke-linecap="round" />
    </symbol>

    <g id="keeper">
      <use xlink:href="#sidebar" fill="#b3dd1a" />
      <use xlink:href="#check" />
      <text text-anchor="start" x="14.5" y="11" font-size="9" letter-spacing="-.5" font-weight="bold">KEEPER</text>

      <use xlink:href="#hr" />
      <g text-anchor="start" font-size="3.5" letter-spacing="-0.05">
        <text x="15" y="18">To play this card, place it face</text>
        <text x="15" y="22.375">up on the table in front of you.</text>
      </g>
    </g>
  </defs>
  <use xlink:href="#keeper"></use>
  <g text-anchor="start" font-size="5.5" letter-spacing="-0.05" font-weight="bold">
    <text x="15" y="31">JavaScript</text>
  </g>
  <text x="10" y="15" transform="matrix(0,-1,1,0,-5,25)"
        font-size="6" text-anchor="end" font-weight="bold">JAVASCRIPT</text>
  <image xlink:href="images/JavaScript-logo.png" preserveAspectRatio="meet" x="22" y="48" width="30" height="30" />
</svg>
```

Chrome Print Dialog



SVG 1.1 Text

Each `<text>` element causes a single string of text to be rendered. SVG performs no automatic line breaking or word wrapping

http://www.w3.org/TR/SVG11/text.html#Introduction

SVG Tiny 1.2

- The `<textArea>` element allows simplistic wrapping of text content
- The `<tbreak>` element is an empty element that forcibly breaks the current line of text
- The `line-increment` property provides limited control over the size of each line

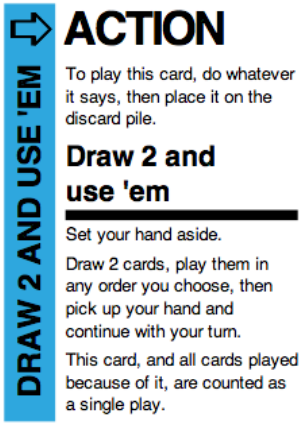
SVG 1.2

- Scalable Vector Graphics (SVG) Full 1.2 Specification
  - W3C Working Draft 13 April 2005 http://www.w3.org/TR/SVG12/
- "The next draft of SVG 1.2 Full will structured as a superset of the SVG 1.2 Tiny language"
- "At this time the refactored SVG 1.2 Full specification is not ready for publication."

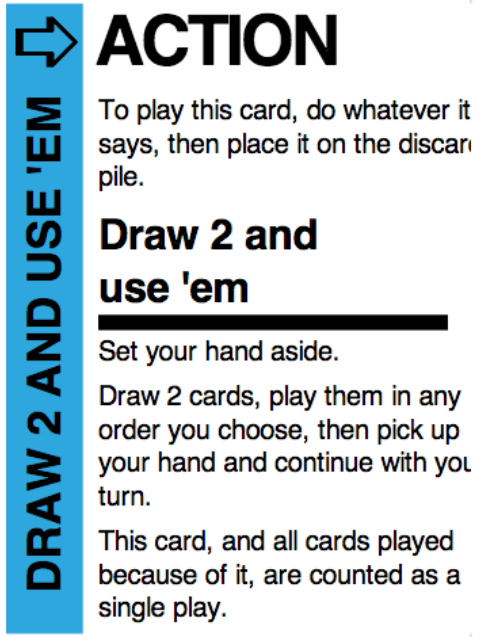
```
snapdeck.textArea = function (s, x, y, width, textToWrap, attr) {
    if (!textToWrap) return;

    attr = SnapDeck.defaultAttr(attr, { "text-anchor": "start", "font-size": 3.5, "letter-spacing": -.05, "line-spacing": 0, "paragraph-
spacing": 0 });
    var fontSize = attr["font-size"];
    width = width * (fontSize / 3);
    var yDelta = (fontSize * 1.25) + attr["line-spacing"];
    var tmpSvg, tmpRoot = deck.svg.g().attr(attr).toDefs();
    var g = s.g().attr(attr);
    var line = "", tmpLine = "", linecount = 0;
    var texts = textToWrap.split("\n");
    texts.forEach(function (text) {
        var words = text.split(" ");
        words.forEach(function (word) {
            if (line) {
                tmpLine = line + " " + word;
                tmpSvg = tmpRoot.text(0, 0, tmpLine);
                if (tmpSvg.node.getBoundingClientRect().width > width) {
                    g.text(x, y, line);
                    y = y + yDelta;
                    linecount++;
                    line = tmpLine = word;
                } else {
                    line = line + " " + word;
                }
                tmpSvg.remove();
            } else {
                line = word;
            }
        });

        g.text(x, y, line);
        y = y + yDelta + attr["paragraph-spacing"];
        linecount++;
        line = tmpLine = "";
    });

    tmpRoot.remove();
    return g;
};
```

Chrome Zoom

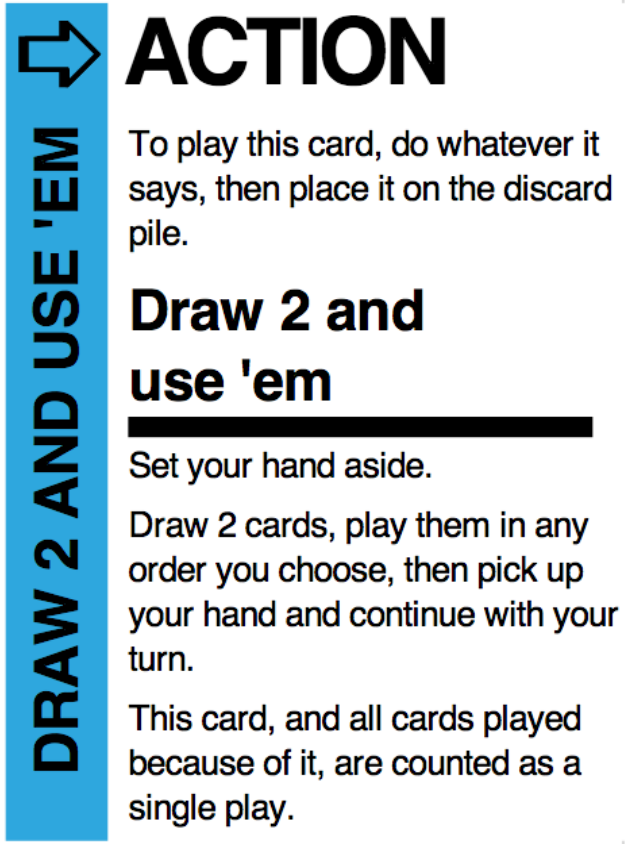
100%



150%



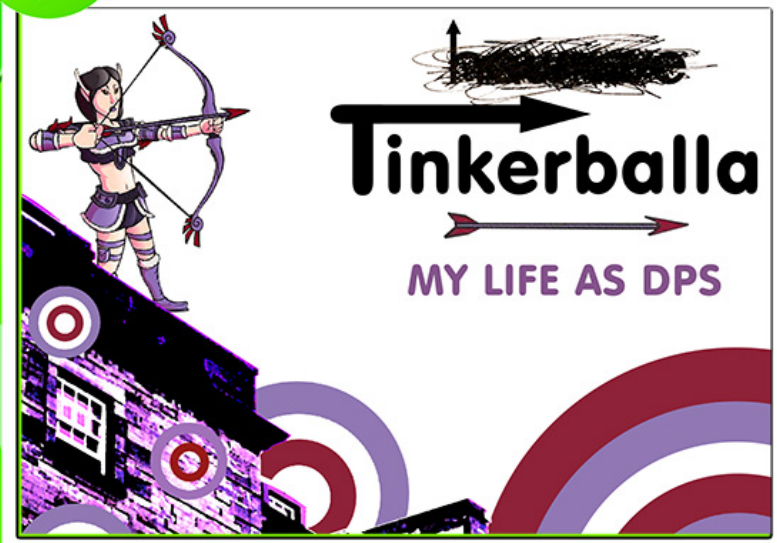
200%



snap.deck.js

```
(function(window) {

    function Deck(cardCount) {
        // ...
        this.svg = Snap("8.5in", (pageCount * 11) + "in")
            .attr({ id: "SnapDeck", style: "display: block;" });
        this.pages = [];
        this.cards = [];
        // ...
    }

    snapdeck.deck = function (cards) {
        return new Deck(cards);
    };

    window.SnapDeck = snapdeck;

})(window);
```

TINKERBALLA



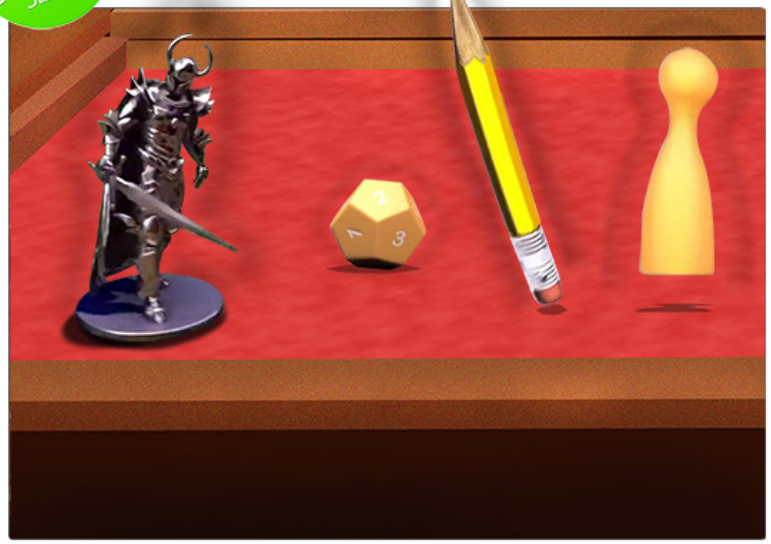
ranger

- Sniper arrow: Discard one Artifact card on table.
- AoE arrow: Deals 1 point of damage to all units on table.

Tink's the best. That other guy's barely meh.

4 / 1 / 2

CURSED GAMING ACCESSORIES



artifact

Play this artifact on any enemy unit.

Unit with this artifact gets -1 to Combat.

Remove any other Artifact card the unit holds.

A set of cursed gaming accessories, animated by dark magicks. The dice always rolls a fumble, gaming pieces and miniatures move when you're not looking. And the damn pencil comes alive at night, changing values in your character sheet and erasing your equipment.

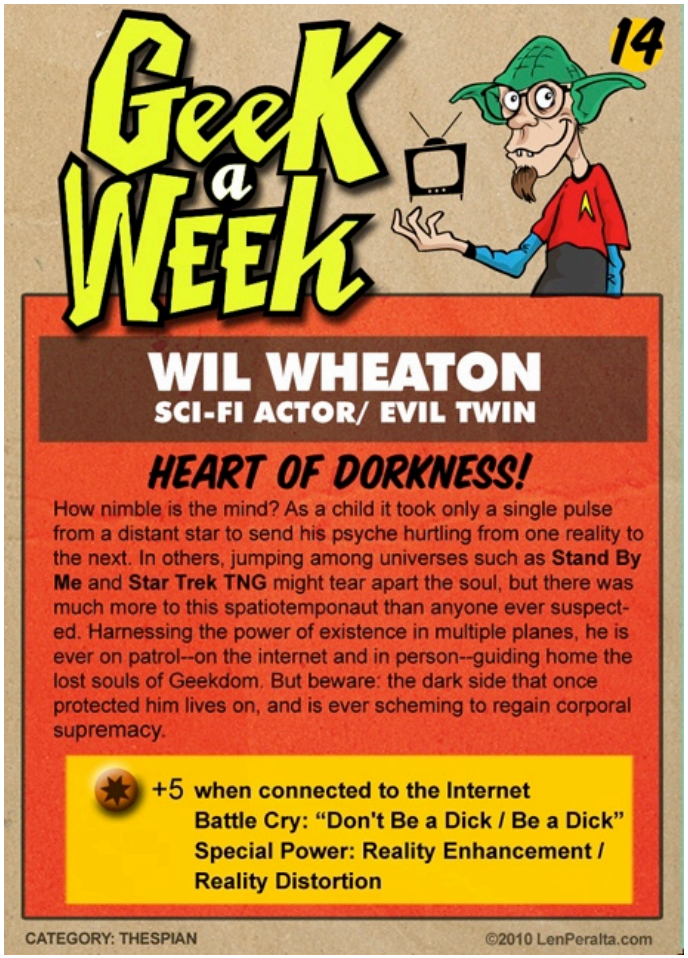
$12

PRINT-AND-PLAY GAME

Literati Trainee

A digital copy of each card, the rule book, and the digital wallpaper image of the Peeve Wars montage. Print and play!

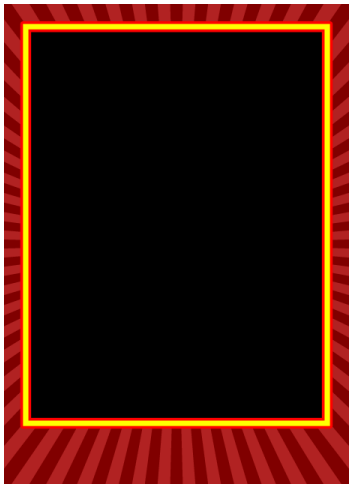
Len Peralta's Geek A Week



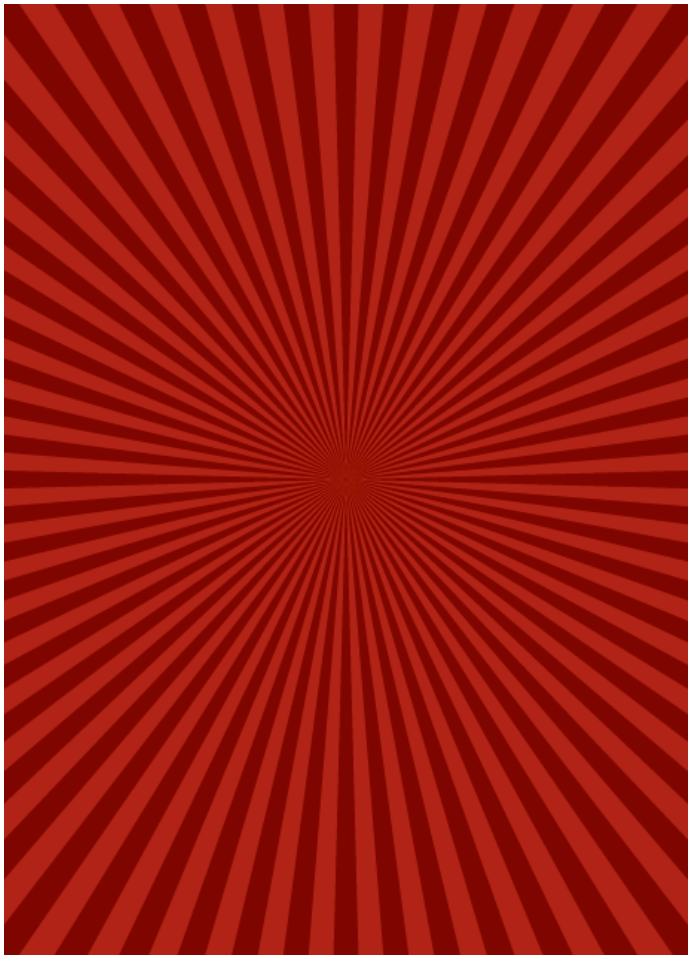
@hakanson Card

- Mimic look from the Geek A Week series
- Information and images from Twitter biography
- “More cool” than a business card

```
var s = Snap("460", "640")
        .attr({ "viewBox": "0 0 63.5 88",
                "font-family":"sans-serif" });

SnapDeck.pinwheel(s, 31.75, 44, 55, 120,
                  "firebrick", "maroon");
```

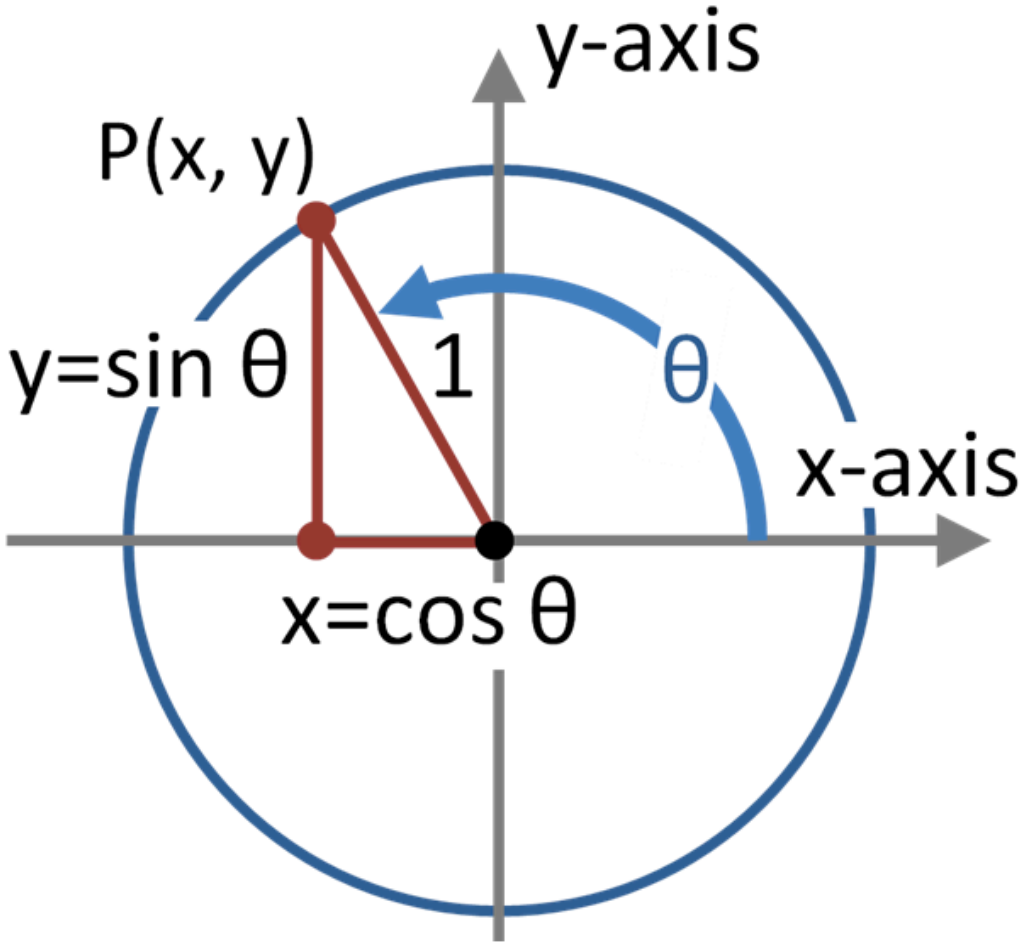


Pinwheel



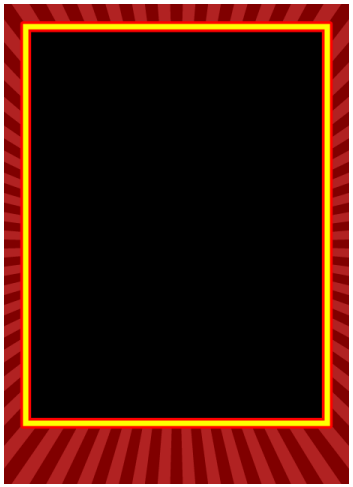
- 120 rays
- firebrick
- maroon

Unit Circle

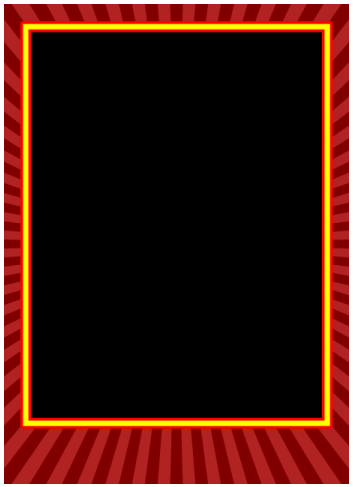


```
snapdeck.pinwheel = function (s, cx, cy, cr, count, background, foreground)
{
    var c = s.circle(cx, cy, cr);
    c.attr({ "stroke-width": 0, "fill": background });

    var deg = 360 / count;
    var offset = -deg / 2;
    for (var i = 0; i < count / 2; i++) {
        var start = "M " + cx + " " + cy;

        var rad = (2 * i * deg + offset) * (Math.PI / 180);
        var x = cx + (Math.cos(rad) * cr);
        var y = cy + (Math.sin(rad) * cr);

        var first = " L " + x + " " + y;

        rad = (2 * i * deg + deg + offset) * (Math.PI / 180);
        x = cx + (Math.cos(rad) * cr);
        y = cy + (Math.sin(rad) * cr);

        var second = " A " + cx + " " + cy + " 0 0 1 " + x + " " + y;

        s.path(start + first + second + " Z").attr({ "fill": foreground });
    }
    return s;
};
```

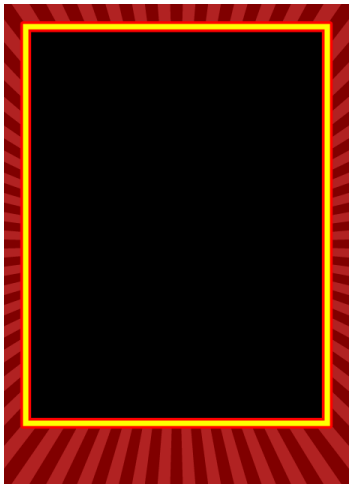
```
var s = Snap("460", "640")
        .attr({ "viewBox": "0 0 63.5 88",
                "font-family":"sans-serif" });

SnapDeck.pinwheel(s, 31.75, 44, 55, 120,
                  "firebrick", "maroon");

s.rect(3, 3, 57.5, 75, 0.5, 0.5)
 .attr({ "fill": "red" });
```



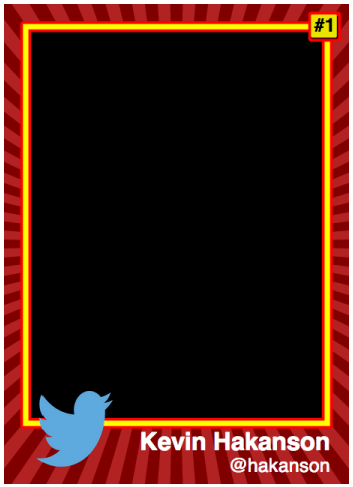
```
var s = Snap("460", "640")
        .attr({ "viewBox": "0 0 63.5 88",
                "font-family":"sans-serif" });

SnapDeck.pinwheel(s, 31.75, 44, 55, 120,
                  "firebrick", "maroon");

s.rect(3, 3, 57.5, 75, 0.5, 0.5)
 .attr({ "fill": "red" });

s.rect(4, 4, 55.5, 73, 0, 0)
 .attr({ "fill": "none",
         "stroke-width": 1, "stroke": "yellow", });
```



```
var s = Snap("460", "640")
        .attr({ "viewBox": "0 0 63.5 88",
                "font-family":"sans-serif" });

SnapDeck.pinwheel(s, 31.75, 44, 55, 120,
                  "firebrick", "maroon");

s.rect(3, 3, 57.5, 75, 0.5, 0.5)
 .attr({ "fill": "red" });

s.rect(4, 4, 55.5, 73, 0, 0)
 .attr({ "fill": "none",
         "stroke-width": 1, "stroke": "yellow", });

s.rect(5, 5, 53.5, 71)
 .attr({ "fill": "black" });
```

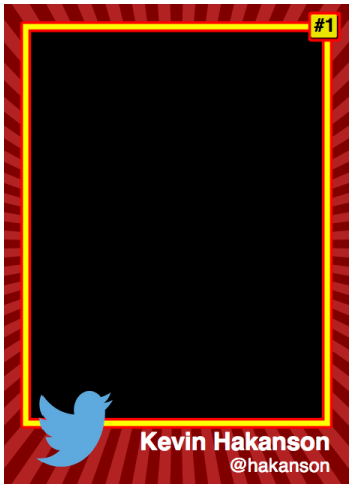


```
s.rect(56, 1.25, 6, 5.5, 0.5, 0.5)
 .attr({ "fill": "red" });
s.rect(56.5, 2, 5, 4.25, 0.25, 0.25)
 .attr({ "fill": "black", "opacity": 0.85 });
s.rect(56.5, 1.75, 4.75, 4.25, 0.25, 0.25)
 .attr({ "fill": "yellow", "opacity": 0.9 });
```

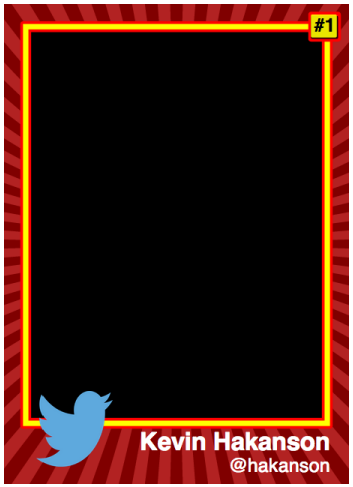



```
s.text(59, 5, "#1").attr({ "text-anchor": "middle", "font-size":
3.5, "fill": "black", "font-weight": "bold" });
```

```
s.rect(56, 1.25, 6, 5.5, 0.5, 0.5)
 .attr({ "fill": "red" });
s.rect(56.5, 2, 5, 4.25, 0.25, 0.25)
 .attr({ "fill": "black", "opacity": 0.85 });
s.rect(56.5, 1.75, 4.75, 4.25, 0.25, 0.25)
 .attr({ "fill": "yellow", "opacity": 0.9 });

s.text(59, 5, "#1").attr({ "text-anchor": "middle", "font-size":
3.5, "fill": "black", "font-weight": "bold" });

s.image("images/Twitter_logo_blue.png", 6, 71, 14, 14);
```



```
s.rect(56, 1.25, 6, 5.5, 0.5, 0.5)
 .attr({ "fill": "red" });
s.rect(56.5, 2, 5, 4.25, 0.25, 0.25)
 .attr({ "fill": "black", "opacity": 0.85 });
s.rect(56.5, 1.75, 4.75, 4.25, 0.25, 0.25)
 .attr({ "fill": "yellow", "opacity": 0.9 });

s.text(59, 5, "#1").attr({ "text-anchor": "middle", "font-size":
3.5, "fill": "black", "font-weight": "bold" });

s.image("images/Twitter_logo_blue.png", 6, 71, 14, 14);

s.text(60, 82, "Kevin Hakanson").attr({ "text-anchor": "end",
"font-size": 4.5, "fill": "white", "font-weight": "bold" });
s.text(60, 86, "@hakanson")
 .attr({ "text-anchor": "end",
         "font-size": 3.5, "fill": "white" });
```



```
s.rect(11.5, 7.5, 40, 40, 0.5, 0.5)
 .attr({ "fill": "white" });
s.image("images/hakanson.png", 12, 8, 39, 39);
```

```
s.rect(11.5, 7.5, 40, 40, 0.5, 0.5)
 .attr({ "fill": "white" });
s.image("images/hakanson.png", 12, 8, 39, 39);

var bioAttr = { "text-anchor": "middle",
         "font-size": 2.75, "fill": "white" };
s.text(31.75, 52,
   "Software Architect at Thomson Reuters.")
 .attr(bioAttr);
s.text(31.75, 56,
   "WestlawNext. Web Platform. JavaScript.")
 .attr(bioAttr);
s.text(31.75, 60,
   "Agile Software Development.").attr(bioAttr);
s.text(31.75, 64,
   "Information Security. Speaker.").attr(bioAttr);
s.text(31.75, 68, "Creator @LicPlateZone").attr(bioAttr);
```

```
s.rect(11.5, 7.5, 40, 40, 0.5, 0.5)
 .attr({ "fill": "white" });
s.image("images/hakanson.png", 12, 8, 39, 39);

var bioAttr = { "text-anchor": "middle",
        "font-size": 2.75, "fill": "white" };
s.text(31.75, 52,
   "Software Architect at Thomson Reuters.")
 .attr(bioAttr);
s.text(31.75, 56,
   "WestlawNext. Web Platform. JavaScript.")
 .attr(bioAttr);
s.text(31.75, 60,
   "Agile Software Development.").attr(bioAttr);
s.text(31.75, 64,
   "Information Security. Speaker.").attr(bioAttr);
s.text(31.75, 68, "Creator @LicPlateZone").attr(bioAttr);

s.text(56, 74, "http://about.me/kevin.hakanson")
 .attr({ "font-size": 2.5, "text-anchor": "end",
        "fill": "lightskyblue" });
```

```
var s = Snap("460", "640")
        .attr({ "viewBox": "0 0 63.5 88", style: "display: block;", "font-family":"sans-serif" });
var cx = 31.75, cy = 44, cr = 55;

SnapDeck.pinwheel(s, cx, cy, cr, 120, "firebrick", "maroon");

s.rect(3, 3, 57.5, 75, 0.5, 0.5).attr({ "fill": "red" });
s.rect(4, 4, 55.5, 73, 0, 0).attr({ "stroke-width": 1, "stroke": "yellow", "fill": "none" });
s.rect(5, 5, 53.5, 71).attr({ "fill": "black" });

s.rect(56, 1.25, 6, 5.5, 0.5, 0.5).attr({ "fill": "red" });
s.rect(56.5, 2, 5, 4.25, 0.25, 0.25).attr({ "fill": "black", "opacity": 0.85 });
s.rect(56.5, 1.75, 4.75, 4.25, 0.25, 0.25).attr({ "fill": "yellow", "opacity": 0.9 });

s.text(59, 5, "#1").attr({ "text-anchor": "middle", "font-size": 3.5, "fill": "black", "font-weight": "bold" });

s.rect(11.5, 7.5, 40, 40, 0.5, 0.5).attr({ "fill": "white" });
s.image("images/hakanson.png", 12, 8, 39, 39);

var bioAttr = { "text-anchor": "middle", "font-size": 2.75, "fill": "white" };
s.text(cx, 52, "Software Architect at Thomson Reuters.").attr(bioAttr);
s.text(cx, 56, "WestlawNext. Web Platform. JavaScript.").attr(bioAttr);
s.text(cx, 60, "Agile Software Development.").attr(bioAttr);
s.text(cx, 64, "Information Security. Speaker.").attr(bioAttr);
s.text(cx, 68, "Creator @LicPlateZone").attr(bioAttr);
s.text(56, 74, "http://about.me/kevin.hakanson")
 .attr({ "font-size": 2.5, "text-anchor": "end", "fill": "lightskyblue" });

// https://about.twitter.com/press/brand-assets
s.image("images/Twitter_logo_blue.png", 6, 71, 14, 14);

s.text(60, 82, "Kevin Hakanson")
 .attr({ "text-anchor": "end", "font-size": 4.5, "fill": "white", "font-weight": "bold" });
s.text(60, 86, "@hakanson").attr({ "text-anchor": "end", "font-size": 3.5, "fill": "white" });
```

Evil Twins?


#14
WIL NO MORE
Geek a Week
WIL WHEATON
SCI-FI ACTOR/ EVIL TWIN
©2010 LenPeralta.com


#1
Software Architect at Thomson Reuters.
WestlawNext. Web Platform. JavaScript.
Agile Software Development.
Information Security. Speaker.
Creator @LicPlateZone
http://about.me/kevin.hakanson
Kevin Hakanson
@hakanson

Questions?