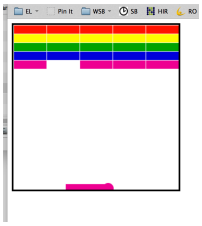
Canvas Tutorial
continue
Simple Game: Break Out

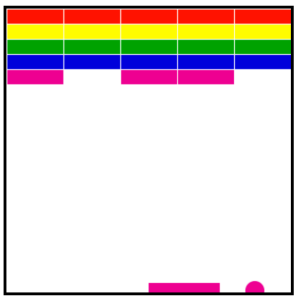
Break Out

- Animate, Game Elements
  - Breakout
  - Bill Mill
    - Modified Tutorial
    - Software developer in Maryland



What it Looks Like

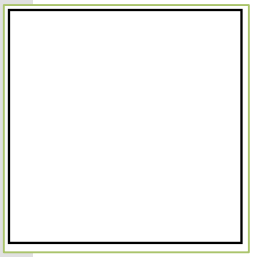
- Elements:
  - Color
  - Collision Detection
    - Simple
  - Interaction with User
    - Mouse
    - Keyboard



11.bricks-really-pretty.html

Review: Rough Structure

```
<!DOCTYPE html>
<html>

<head>
    <style type="text/css">
            canvas { border: 3px solid black; }
    </style>
<script type="text/javascript">
.
.
.
</script>
</head>

<body>
<canvas id="canvas" width="300" height="300">
Your browser does not support the canvas element.
</canvas>
</body>
</html>
```

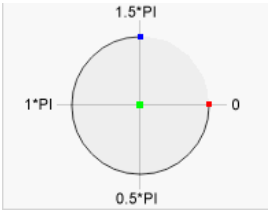


Abstracted: 1.intro.html

JavaScript Structure (Finished Code)

```
// Global variables
// Initialization methods
// Mouse & Keyboard specifications
// Basic Shapes
// Game logic
```

Circle



- ctx.beginPath()
- ctx.arc(75, 75, 10, 0, Math.PI*2, true);

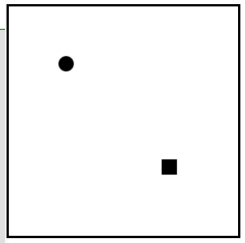
  center — Start angle — End angle — CCW (False is default and is CW)

- ctx.endPath()
- ctx.stroke();
- ctx.fill();

http://www.w3schools.com/tags/canvas_arc.asp

Review: Step 1 : JavaScript Simple Shapes
Ball in Break out (a circle), Bricks 'rectangle')

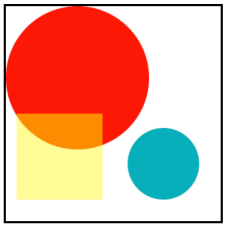
```
<script type="text/javascript">
var x=200;
var y=200;
window.onload=function()
{
var c = document.getElementById( "canvas" );
var ctx = c.getContext( "2d" );

//draw a circle
ctx.beginPath();
ctx.arc(75, 75, 10, 0, Math.PI*2, true);
ctx.closePath();
ctx.fill();  // actually draws it
</script>
```



```
// draw a rectangle
    ctx.beginPath();
    ctx.rect( x, y, 20, 20);
    ctx.fill();
    ctx.closePath();
```

1.intro.html

Color Fun



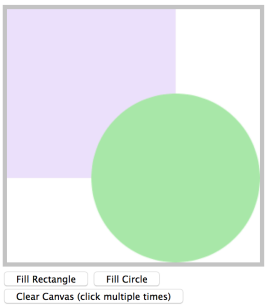
- 3 Circles
- ctx.fillStyle = "#FF1C0A";
- ctx.fillStyle = "#0FB2BB";

- ctx.fillStyle = "rgba(255, 255, 0, .5)"
  - 0 is completely transparent
  - 1 is completely opaque
  - 0.5 half transparent

2.color.html
2.color-fun-with-alpha.html

Buttons: onclick

- <**button** onclick="fillrectangle()">Fill Rectangle</button>



2.RectCircle-Buttons.html

- Stroke.
- Move to.
- 2-mozilla-face.html

Adding Animation

- Create a function that
  (1) clears canvas, and
  (2) then draws objects

in a repeatable manner (e.g., a draw() function).

- Primitive:
  - setInterval(function, timeout) in the init()
    - Using : SetInterval() for now
      - CAVEAT: HW read the below links (use RequestAnimationFrame())

http://stackoverflow.com/questions/13935262/settimeout-or-setinterval-or-requestanimationframe
http://creativejs.com/resources/requestanimationframe/

Example Animation

```
var x = 0;
var y = 0;
var dx = 2;
var dy = 2;
var interval =10;
var ctx;

function init()
    {  // set ctx here //
    return setInterval( draw, interval );
    }

function draw() {  ctx.fillStyle = "red"
  ctx.clearRect(0,0,300,300);
  ctx.beginPath();
  ctx.arc(x, y, 10, 0, Math.PI*2, true);
  ctx.closePath();
  ctx.fill();
  x += dx;
  y += dy;
}
```

3.action.html

Exercises: How would you make it continue to animate?

1) draw() - Draws a ball

2) Then that draw function needs to be called periodically.

How about changing dx, dy? At each time draw is called

Modularize

```
//BEGIN ----- LIBRARY CODE
var x = 150;        var y = 150;
var dx = 2;         var dy = 4;
var WIDTH;          var HEIGHT;
var ctx;

function init()
{
      ctx=document.getElementById('canvas').getContext('2d');
      WIDTH = ctx.canvas.width;
      HEIGHT = ctx.canvas.height;
      return setInterval( draw, interval );
}

function circle(x,y,r)
{
      ctx.beginPath();
      ctx.arc(x, y, r, 0, Math.PI*2, true);
      ctx.closePath();
      ctx.fill();
}

function rect(x,y,w,h)
{
      ctx.beginPath();
      ctx.rect(x,y,w,h);
      ctx.closePath();
      ctx.fill();
}

function clear()
{
      ctx.clearRect(0, 0, WIDTH, HEIGHT);
}

//END ----- LIBRARY CODE
```

```
function draw()
{
      clear();
      circle(x, y, 10);

      x += dx;
      y += dy;
}
```

Circle Code
Rectangle Code
Clear

3.library.html

Bounce

- **Add Some Physics**: Collision and Gravity
- Detect when the ball is 'beyond' the canvas boundaries.

```
function draw()
      {
      clear();
      circle(x,y,10);

      // if outside the width canvas, change direction of ball.
      if (x + dx > WIDTH  || x + dx < 0)
            dx = -dx;
      if (y + dy > HEIGHT || y + dy < 0)
            dy = -dy; // -.5;
      x += dx;
      y += dy;
      }
```

6.bounce.html

- Linear Motion (speed depends on direction up or down)
  - More realistic accelerates while descending, (b/c of gravity, and slows down while bouncing up).

Gravity

- **Reading Assignment:**
  - http://codetheory.in/basics-of-implementing-gravity-with-html5-canvas/
  - Rishabh's Code Theory Web Site
  - Freelance Web & Mobile Developer from India:

◎ **Let's try it : Skim it for now… later on, on your own recreate some of the ideas at home.**

◎ **Simple example for now : Lets look at it:**

◎ **6.bounce-gravity.html**

- Gravity
  - Force affecting the speed of ball's 'y' coordinate
    - Slows the ball as it goes up
    - Speeds up the ball as it goes down.
  - Check the code
    - Exercise:
      - **Debug it so it doesn't fully sink down beyond the floor.**

… adding an Paddle

- Add a non-moving `paddle' (rectangle)
  - Allow ball only to bounces off the paddle, otherwise ball is out of bound.
    - (only 'beyond floor')
    - init_paddle(); // sets the paddle's variables
    - In draw function add code that decides whether to bounce, or 'disappear'
      - when it hits (ball bounces) or misses (ball disappears) the paddle

7.paddle.html

… add **User Interaction**: KeyBoard Control

- Buttons & links have **onClick** event handlers
- Keyboard handlers have to install handler manually, as **keyboard listener.**
  - `addEventListener( event_type, event_handler, capture )`
  - https://www.w3.org/TR/2003/NOTE-DOM-Level-3-Events-20031107/events.html#Events-phases
- Allow the paddle to move
  - Left Arrow Input
  - Right Arrow Input
- Resources:
  - http://www.asquare.net/javascript/tests/KeyCode.html
  - http://www.w3schools.com/jsref/event_key_charcode.asp
  - Key UP, Key DOWN
- 'Who' monitors the input?
  - Canvas, Browser, Window Manager

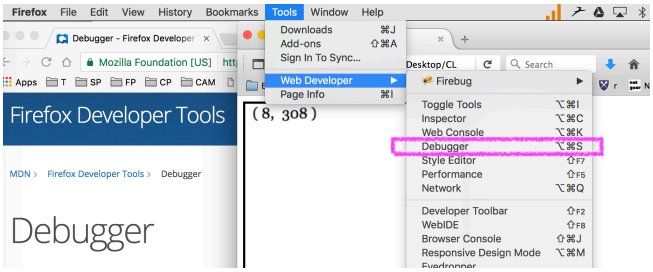
8-1.keyboard-simple.html
8-2.keyboard-purple-ball-keyboard-SHIFT.html
8-3.keyboard-paddle.html

… add **User Interaction**: Mouse Control

- mouseMove event to a user specified function:
  - onMouseMove function,
  - Checks to see if the mouse is within the borders of the paddle, and move the paddle if it is.
  - Movement and Distance of paddle

9.mouse.html

Firefox Debugger



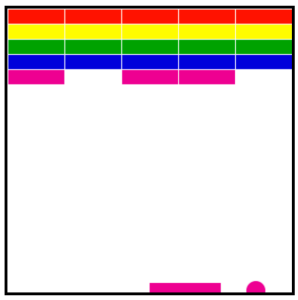
https://developer.mozilla.org/en-US/docs/Tools/Debugger

… Brick and Collisions

- See code, simple `collision detection' (looks for overlaps)
- More in-depth collision discussion next week.

What Game Looks Like …

- Features:
  - Color
  - Animation
  - Collision Detection
  - Interaction with User
    - Mouse
    - Keyboard



11.bricks-really-pretty.html

Next Up

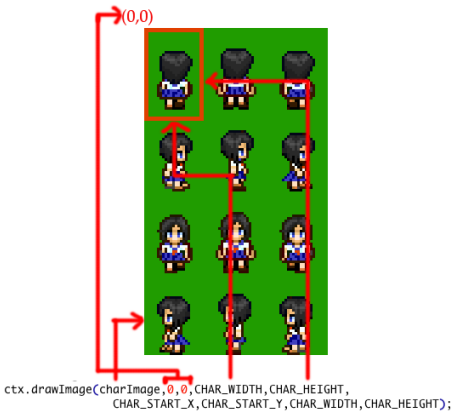
- Sprite:
  - What is a sprite?
  - Sprite movements.
- Parallax:
  - What is a parallax
  - From Simple Parallaxing to …

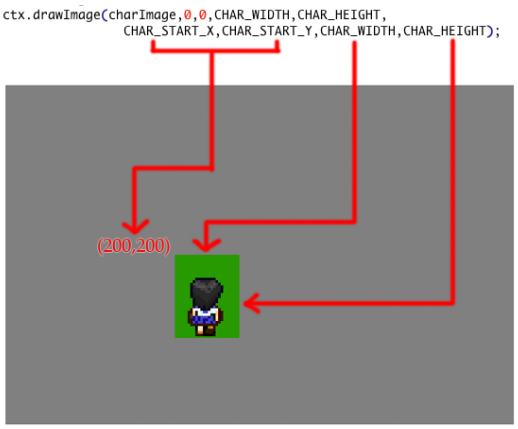
Sprite

- Load Image
- Draw Image onto Canvas
- Animate Sprite

Sprite Magic

```
ctx.drawImage( charImage,          // sprite sheet
currX, currY,                      // top left corner of sprite sheet
CHAR_WIDTH, CHAR_HEIGHT,           // size one instant
CHAR_START_X, CHAR_START_Y,        // game canvas location
CHAR_WIDTH, CHAR_HEIGHT );         // size on canvas (enables sizing
                                   // up or e down
```



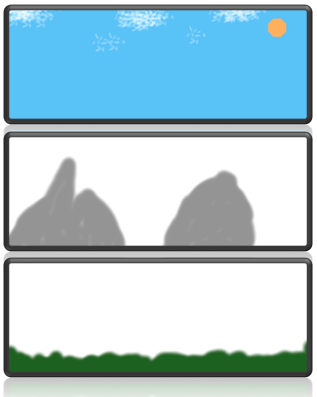


Parallax

- http://en.wikipedia.org/wiki/Parallax_scrolling
- http://javacoffee.de/?p=866

Layers

```
function Layer
      (
      s,   // path to image
      x, y
      )
{
      this.img = new Image();
      this.img.src = s;
      this.x = x;
      this.y = y;
}
```



Next week

- More about collision
- Physics
- Modularizing data with Javascript (Kandi.js)