All the syntax and stuff I need:

* “use strict”; //activates strict mode //good while learning
* You can assign function, element and all sort of stuff to a variable:

Let function\_name = function() {//code here};

This requires you to declare before call

* Declaring function function\_name() {//code here}; solves it
* Composite variables can be declared and accessed as:
  + var variable\_name = { name: value, name: value, name: value; };
  + console.log(variable\_name.name);
* Types in Js are like class (variables with predefined values and function):
  + Like Image();
* document.addEventlistener(“DOMContentLoaded”, function);
* you use () to execute a function immediately and without () to pass a reference to a function that will be executed later, often by an event or loop like requestAnimationFrame.

1. Canvas element needs id, width and height attribute. (width = “800”)
2. You need two main variables: canvas and canvasContext (name can be different)
   1. Canvas = document.getElementByID(“id\_here”);
   2. canvasContext = canvas.getContext(“2d”); //i.e. getContext() method
3. Main loop requires update() and draw() function call repeatedly.
4. window.setTimeout( function, time in ms); can be used to add delay
   1. You can omit the window.
   2. Better option is requestAnimationFrame(function);
5. Sprites can be declared as image type:
   1. Game.sprite\_name = new Image();
   2. Game.sprite\_name.src = “source\_here”;
6. drawImage method allows you to draw images loaded:
   1. Game.canvasContext.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);

Img is sprite, sx where to start clipping sprite, swidth is width to clip, x is where to place, width to stretch or reduce the image.

* 1. Game.canvasContext.drawImage(sprite, 0,0, sprite.width, sprite.height, -origin.x, -origin.y, sprite.width, sprite.height); //This is good too

1. canvasContext has fillStyle property and fillRect method:
   1. canvasContext.fillStyle = ”blue”;
   2. canvasContext.fillRect(0,0,canvas.width, canvas.height);
2. canvasContext allows you to save() and restore() drawing state:
   1. translate(x,y) translates the drawing state
   2. rotate(angle) rotates drawing state. (angle in radian) (Use trigonometry to calculate angle)
3. Event handlers in JS allow you to execute instructions when specified event occurs.
   1. document.onmouse = function\_name;

This function when called is passed an evt parameter which depends on event that occurred.

* 1. evt has pageX and pageY which gives mouse position taking scrolling into account.
  2. Also, clientX and clientY which doesn’t take scrolling into account.
  3. For keyboard, there is document.onkeydown = function\_name; The evt passed has keyCode. Similarly, there is onkeyup event. Combine those and you can react to keyboard. A is 65, B is 66… R is 82, G is 71, Y is 89. You can add a keyDown variable inside keyboard object with initial value of -1. Change it to evt.keyCode on keypress and back to -1 on release.
  4. Also for mouse, onmousedown and onmouseup can be used evt has a which variable which has 1 for left, 2 for middle and 3 for right button. For mouse, you require Down and Pressed variable for each button. i.e. For left you need leftPressed as well as leftDown.
  5. Explanation for d and e: With mouse you need to track mouse held as well as mouse clicked but for keyboard you just require which key is being pressed only.

1. LABjs is a dynamic script-loading tool.

<script src=”../LAB.min.js”></script>

<script>

$LAB.script(‘input/Keyboard.js’).wait()

.script(‘input/Mouse.js’).wait()

.script(‘Canvas2D.js’).wait()

.script(‘Cannon.js’).wait(function() {

Game.start(‘myCanvas’);

});

</script>

1. Loading Sprites before starting game:
   1. Add method loadSprite to the Game object
   2. Add a variable spritesStillLoading to the Game object
   3. Inside loadSprite function, increment value of spritesStillLoading
   4. Also add evenet listener to the image you are loading i.e. image.onload = function() {};

Which decreases vale of spritesSillLoading. When it reaches 0, start initialize() method and main loop. Event listener on image not Image.