



Greetings!

Adelle Lin: adelle@codeliberation.org

Rachel Bazelaïs: rachel@codeliberation.org



Who are you?

Tell me about yourself!



What is Code Liberation?

A little about us.



Many popular games are made of:

- **Characters:** beings with which players interact
- **Mechanics & Rules:** systems that determine how players interact with the game
- **Goal:** what players must do to finish the game
- **Story:** narrative driving the game's goal & rules



I also like to think about games in this way:

- **Relations:** Narrative driving the game's systems
- **Systems:** Rule sets and goals that determine how players interact with the game
- **Environment** World that the narrative and systems are set in - in game and out



What is HTML?

HTML is a **Markup language** (code-based annotation system)

as it enables its members to insert spaces among canonical (i.e. predetermined) practices in which to develop non-canonical views – that is, ones richer and more flexible and subject to constant change. Within these spaces, there develops and is preserved a situated knowledge which becomes a collective asset and the source of idiosyncratic power. Brown and Duguid's contribution has given rise to a set of studies, ~~still relatively little developed~~, that seek to understand innovating as a situated activity.

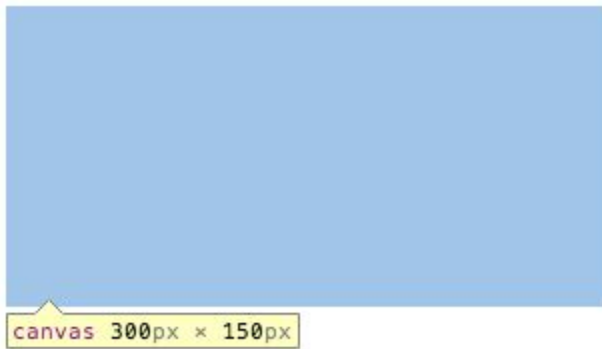
The assumptions on which innovation may be considered as a continuous process situated in work practices are the following:

- Knowledge is produced through participation in a set of practices.



What is <canvas>?

An HTML5 tag that allows you to draw things in your browser using JavaScript.




It's a box that can make anything happen!



Structure of JavaScript

Commands **usually** require **;** at the end.

Code runs once from top to bottom unless any parts are commanded to repeat.



```
wakeUp;  
dance;  
shower;  
goOutAndGetSunshine;
```




Structure of JavaScript

In JavaScript, the concept of variables is important.

Variables hold many **data types**: numbers, strings, arrays, objects, booleans and more. So you can declare most types as variables:

```
var name = "Adelle";  
var likes = ["coffee", "dancing", "making things"];  
var happy = true;  
var worth = 10000000000000000000;
```



What is p5.play?

- A p5.js library that facilitates the making of games and playthings
- p5.js is a javascript library that stems from processing
- p5.play has a built in sprite class that manages basic collisions, user input and virtual camera



What is p5.play?

p5 has very little setup overhead. It comes with it's own editor and compiler which allows for initial prototyping locally.

Sketches can then be quickly uploaded to any web browser for sharing.



Setting up p5.play



Download the p5 libraries and the editor

<http://p5js.org/download/>

Press “Download ZIP”



Editor

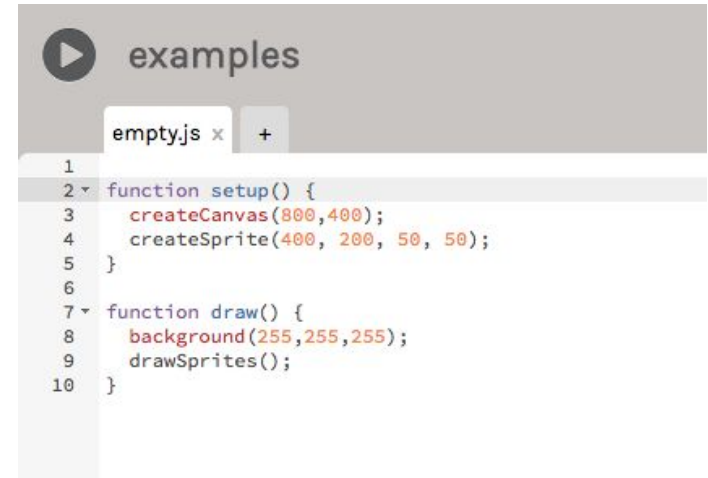
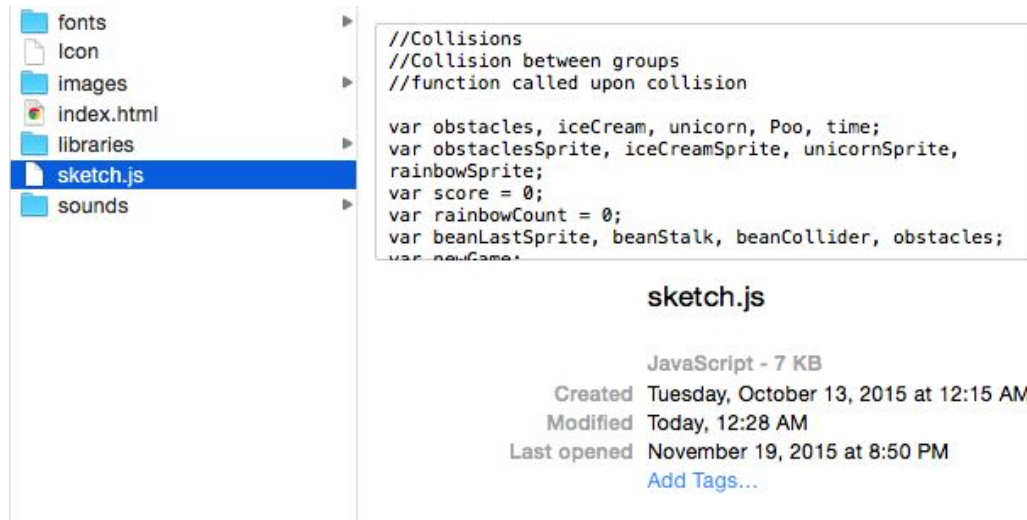
The p5.js editor is currently in development, try out a beta version of it now. Help out by posting [feedback and bugs](#). Support for Linux coming soon, along with more [features](#).





Install the p5 editor

Open **sketch.js** in the p5 editor





Download and unzip the p5.play library

<http://p5play.molleindustria.org/>

p5.play

Download * Examples * Get Started * Reference * Source on Github

p5.play is a [p5.js](#) library for the creation of games and playthings.

p5.play provides a [Sprite](#) class to manage visual objects in [2D space](#) and features such as [animation support](#), [basic collision detection](#) and [resolution](#), sprite [grouping](#), helpers for mouse and keyboard [interactions](#), and a [virtual camera](#).



Link the p5.play.js from your HTML file

`<script src="libraries/p5.play.js" type="text/javascript"></script>`

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>unicornsEatIceCream</title>
5      <!-- link p5.js and its addons like p5.dom.js or p5.sound.js -->
6      <script src="libraries/p5.js" type="text/javascript"></script>
7      <!-- link p5.play.js -->
8      <script src="libraries/p5.play.js" type="text/javascript"></script>
9      <!-- link your javascript sketch -->
10     <script src="sketch.js" type="text/javascript"></script>
11     <!-- link sound library -->
12     <script src="libraries/p5.sound.js" type="text/javascript"></script>
13   </head>
14   <body>
15   </body>
16 </html>
```




To see on a local browser

Open index.html



Preparation

Get the starter code for today's game on GitHub!

The URL: github.com/adellelin/unicornEatsIceCream

Click the **Download ZIP** button on the right side.



Let's code!



We'll learn:

- Loading images
- Animating sprites
- Adding keyboard interactivity to your game
- Basic collision
- Adding sound to your game



Images



You can download placeholder art

Find art here:

<http://opengameart.org/>



Loading Images

There are several ways to load images in p5:

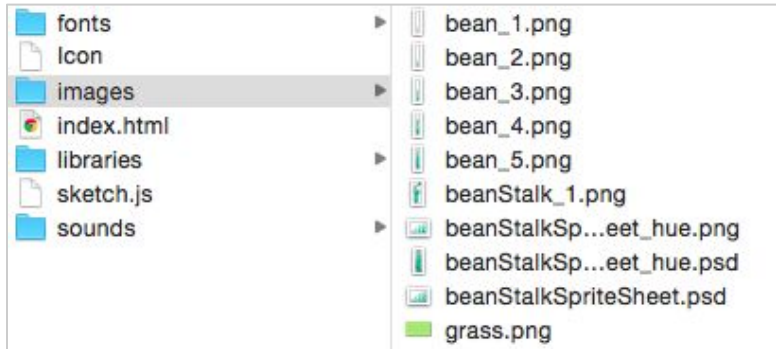
- **image** - static, no animation
- **animation** - a series of numbered static images



Locating image files

All files should be referenced from the **root**, the main folder where your project is located.

`'images/grass.png'`





Sprites



Loading sprites

Sprites are loaded within the **preload()** function. We use this function to load external files such as images, animations, fonts and sounds.

```
function preload() {  
  playerSprite = loadImage("images/player.png");  
}
```



Drawing sprites

You can draw, or place, objects onscreen using p5.play's **createSprite()** function. If you have loaded an image, you can add this to the sprite.

```
var player = createSprite(0, 0, 10, 10);  
                                            
                        X, Y      size  
player.addImage(playerSprite);
```



Drawing sprites

Your sprite object has properties which you can access such as position, velocity, scale, collider.

```
player.position.x = mouseX;  
player.position.y = mouseY;
```



Drawing sprites

Don't forget to draw your sprites!

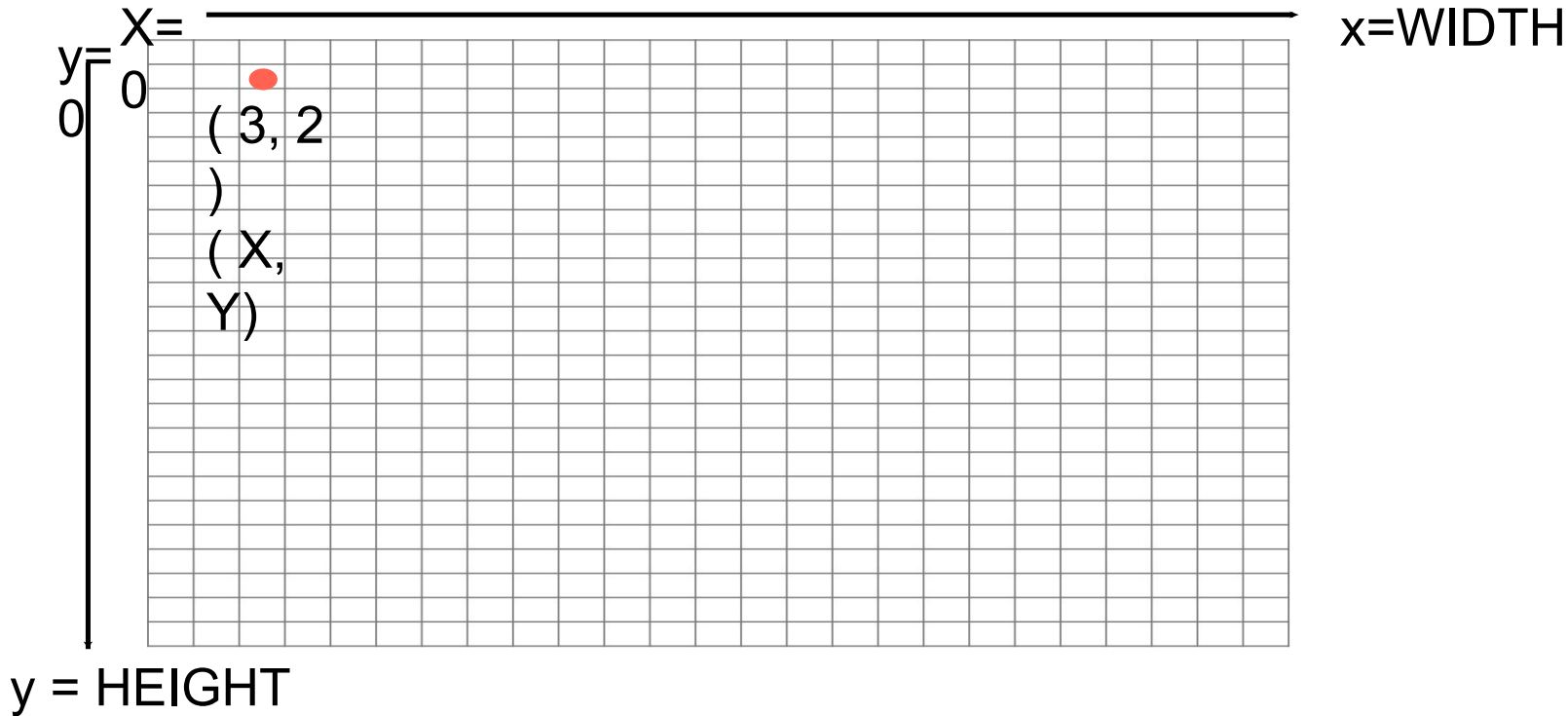
```
function draw() {  
  var player = createSprite(0, 0, 10, 10);  
  player.addImage(playerSprite);  
  drawSprites();  
}
```



Interactivity & Animation



Positioning within the canvas





Keyboard input

In p5.play :

- up
- right
- down
- left

```
keyDown (UP_ARROW)  
keyDown (RIGHT_ARROW)  
keyDown (DOWN_ARROW)  
keyDown (LEFT_ARROW)
```

keyDown checks if a key currently pressed and returns a boolean value.

keyWentDown checks if a key was pressed in the last cycle and returns a boolean value. Used to trigger discrete events like a jump or shoot.

```
if (keyWentDown ("x")) {marksTheSpot () }
```




Animating sprites - .png

Load your animations within the **preload()** function.

Images must be in .png format!

You can set the sequence you would like:

```
var coffeeCoffee = loadAnimation("images/milk.png",  
  "images/sugar", "images/coffee", "images/hotWater.png");
```

Or number your images and p5.play will detect the sequence:

```
var coffeeCoffee = loadAnimation("makeCoffee_1.png",  
  "makeCoffee_4.png");
```



Animating sprites

Add the animation to a sprite:

```
var player = createSprite( 0, 0);  
player.addAnimation = ("playerMakesCoffee", coffeeCoffee);
```

Animations can also be their own objects:

```
animation(coffeeCoffee, 100, 100);
```



Animating sprites

Animations will play automatically in the draw loop, but you may not want it so.

```
coffeeCoffee.play(); or player.animation.play();
```

Animation object

Sprite Object

You may want to set the frame rate and whether the the animation loops

```
coffeeCoffee.frameDelay = 5;  
coffeeCoffee.looping = false;
```



Booleans

Booleans is a data type that returns two values --> true or false.
Booleans can be used like switches.



```
var dancing = false;

function musicStarts() {
  if (!dancing) {
    moveToTheBeat();
    dancing = true;
  } else {
    tired();
    dancing = false;
  }
}
```



Collisions!





Checking for collisions

Sprite objects have a collider property that is set to the size of the image. This can also be set using:

```
player.setCollider("circle", 10, 10, 20, 40);
```

Diagram illustrating the parameters of the `setCollider` method:

- `"circle"`: type
- `10, 10`: offset X and Y
- `20`: width
- `40`: height



Checking for collisions

There are a number of collisions types available - **collide()**, **displace()**, **overlap()**, **bounce()**. You can set a target object and callback function for each.

```
player.collide(wall, ouch);
               |      |
               target  function
                       called

function ouch() {
    player.animation.play();
}
```



for Loops

for loops are extremely useful. They let you run a block of code a certain number of times allowing you to go through a list of items in a collection.

```
for (initialize; test; increment) {  
    do that dance;  
}
```




Adding to groups

Groups are very useful array types as they can be used to apply common properties across sprites.

```
var friends = new Group();  
for (var i = 0; i < 3; i++) {  
    var adelle = createSprite(0, 0, 50, 100);  
    friends.add(adelle);  
}
```



Adding to groups

There is an all power group called `allSprites` that **all sprites** are automatically a part of. To access this group:

```
for (var i = 0; i < allSprites.length; i ++) {  
  var allFriends = allSprites[i];  
  allFriends.remove();  
}
```



Adding Sound



Find a sound!

Sites like **freesound.org** are great places to find placeholder sounds for your game.

username: uaicodelib

pass: uaixcodelib123



Preloading audio

You can preload audio using **load.audio()**. Declare a variable so you can reference the sound later.

```
var funnySound = loadSound("sounds/highGiggle.mp3");
```



Adding audio to your game

You can now access properties of the sound file. These can be used inside of functions and conditional statements to add fun to interactions:

```
funnySound.setVolume(0.2);  
funnySound.play();  
funnySound.pause();  
funnySound.stop();
```



Looping audio

The **loop()** function comes with additional settings that also allow you to modify or loop your sound.

```
funnySound.loop(5, 1, 1, 10, 30)
```

5,	1,	1,	10,	30)
start		amp	cue	duration
Time	rate		Loop	
			Start	



Dialogue and Game UI



Dialogs

Sometimes you might want to show **dialogs**, or windows with text in them, during your game.





Dialogs

There are several parts to creating dialogs:

- A background image
- Text content
- Easy way to change text





Adding Dynamic Text

It is pretty simple to add text in p5

```
text("let's start making the game already", 0, 0);
```

0, 0
X, Y
positions

```
var iceCream ++;
```

```
text("Number of ice creams" + iceCream, 0, 0);
```



Buttons

Within p5.play you can turn sprites into buttons by adding the **onMousePressed()** function.

```
funnySprite.onMousePressed = function() {  
  startLaughing();  
}
```

You can also use the p5.dom library to create buttons.