# Processing.js Tutorial

Nathan Lapierre & Thomas Morrison
CSCI 4126, Ubiquitous Computing
June 12th, 2012

## Overview

This tutorial will walk you through creating an application using *Processing.js*. All code is included and screenshots are given. Instead of installing *Processing.js* on your own website you can access a browser based editor here to create & update applications on the fly without any setup. This tutorial assumes that you have a basic knowledge of Java or some C like language.

## Background

What is it?

A port of the popular open source programming language *Processing* used to create interactive visual programs. Ex1 Ex2

How is it used?

Code is written with the Processing language and is then compiled into JavaScript. Once compiled the program can be viewed in a web browser.

What does it require?

Requires an HTML5 compliant browser, but does not require any plugins etc.

# Pre-Tutorial

## Step 1: Setup

1. Ensure that your web browser of choice is updated to its most current iteration.
2. Open a browser tab to this URL, this page contains a web UI for easy creation of applications with no setup.

   It may be helpful to have the language reference open, located here

## Step 2: Understanding Processing.js

The syntax for *Processing.js* is very simple and can easily be understood by anyone with a background in programming. That being said there are three main points understand before creating an application.

1. At the start of every application the **setup()** function called, this is where variable values are set, application properties like canvas size are changed etc.
2. The **draw()** function is inherently called each time the screen updates, it is here that you decide what is drawn to the screen, do hit tests and alter existing objects.
3. Finally, calling **exit()** will end the *Processing.js* application, clean up all resources and ensuring that draw() is no longer called.

## Step 3: Setting up the Canvas

When creating a *Processing.js* application the first step is to call a few specific functions to setup the canvas & application as a whole.

In the setup() function place these statements.

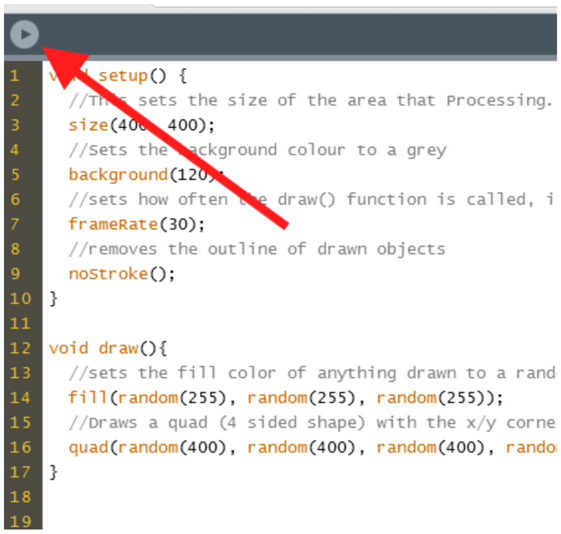
```
//This sets the size of the area that Processing.js will draw too
size(400, 400);
//Sets the background colour to a grey
background(120);
//sets how often the draw() function is called, in calls per second
frameRate(30);
//removes the outline of drawn objects
noStroke();
```

## Step 4: Drawing to the Screen

Finally, in the draw function place this statement.

```
//sets the fill color of anything drawn to a random colour
fill(random(255), random(255), random(255));
//Draws a quad (4 sided shape) with the x/y corners being random
quad(random(400), random(400), random(400), random(400), random(400),
random(400), random(400), random(400));
```

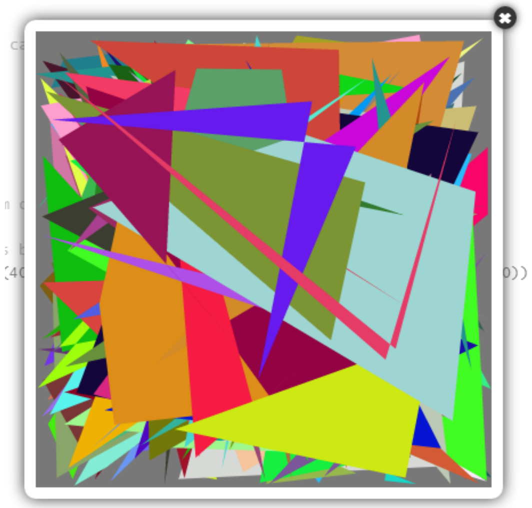
To run this application press the "play" button which looks like this:



```
void setup() {
  //This sets the size of the area that Processing.
  size(400, 400);
  //Sets the background colour to a grey
  background(120);
  //sets how often the draw() function is called, i
  frameRate(30);
  //removes the outline of drawn objects
  noStroke();
}

void draw(){
  //sets the fill color of anything drawn to a rand
  fill(random(255), random(255), random(255));
  //Draws a quad (4 sided shape) with the x/y corne
  quad(random(400), random(400), random(400), rando
}
```

After pressing this you should see something along the lines of this:



What you just completed was just a very basic intro tutorial to get you used to the web environment & see a bit of *Processing.js* syntax. The following steps will guide you through creating a basic interactive graphical application.

## Basic interaction

Global variables to define a random shape are defined at the start:

```
float q1 = random(400);
float q2 = random(400);
float q3 = random(400);
float q4 = random(400);
int r = random(255);
int g = random(255);
int b = random(255);

void setup() {
  size(500, 500);
  smooth();
}
```

Within our draw routine, we are now clearing the screen before every frame. We use a predefined variable to scale a shape based on the mouse value:

```
void draw() {

  //clears the screen
  background(127);
  stroke(0);

  //scales based on mouse value - predefined variable
  scale(1.0/(mouseY*0.5));

  translate(width/2, height/2);

  //show the quad
  fill(r, g, b);
  quad(q1, q2, q3, q4, q4, q3, q2, q1);

}
```

A defined event handler runs when the mouse is clicked:

```
//handle mouse clicks
void mouseClicked() {
  q1 = random(400);
  q2 = random(400);
  q3 = random(400);
  q4 = random(400);
  r = random(255);
  g = random(255);
  b = random(255);
}
```

There are many predefined variables and event handlers that can be added to your application.

For next steps to creating a useful Processing addition to your project, look at the many examples on processingjs.org, and check out:

- File I/O, input from web using JS integration
- Input from other sensors