Processing transition

Michael Hadley edited this page on Mar 18 · 29 revisions

Edit New Page

Overview of differences

The p5.js language looks very similar to the Processing language with a few changes:

- Because you can think of your sketch as more than just the drawing canvas, `size()` has been replaced with `createCanvas()`, to suggest the possibility of creating other elements.
- `frameRate(num)` sets the frame rate, but the `frameRate` variable has been removed. To get the current frame rate, call `frameRate()` with no arguments.
- JavaScript doesn't always load things synchronously, there are a couple options to deal with this:
    - All load methods take an optional callback argument. That is, a function that gets called after the file has been loaded.
    - Alternatively, you can place load calls in a `preload()` method that happens before `setup()`. If a preload method exists, setup waits until everything inside is loaded, see this image example.
- The variable `mousePressed` has been replaced with `mouseIsPressed`.
- In addition to mouse events, there are touch events, the mapping is like this:
    - `mouseX` ~ `touchX`
    - `mouseY` ~ `touchY`
    - `mousePressed()` ~ `touchStarted()`
    - `mouseDragged()` ~ `touchMoved()`
    - `mouseReleased()` ~ `touchEnded()`
    - There is a `touches[]` array that contains a series of objects with x and y properties corresponding to the position of each finger.
- `push/popMatrix()`, and `push/popStyle()` have been replaced with `push()` and `pop()`, the equivalent of calling both matrix and style methods together.
- By default, everything is in the global namespace, and you can create your sketches like you do with Processing. However, there is something we call "instance mode" for creating a p5 sketch that plays nice with the rest of the code running on your page. See this instance mode example and this global vs instance mode tutorial.
- In global mode, p5 variable and function names are not available outside `setup()`, `draw()`, `mousePressed()`, etc. (Except in the case where they are placed inside functions that are called by one of these methods.) What this means is that when declaring variables before `setup()`, you will need to assign them values inside `setup()` if you wish to use p5 functions. For example:

```
var n;
```

Clone this wiki locally

https://github.com/proces

Clone in Desktop

```
function setup() {
  createCanvas(100, 100);
  n = random(100);
}
```

- Not everything in Processing is implemented in p5.js, but we are working on it! Right now there is no PShape equivalent. The camera model in p5js is yet very basic, with only eye position and no "look at" or axis direction. See the reference for up to date documentation of what functions work.

Some things about JavaScript

- Variables do not have a type. Use var instead of float, int, double, long, char, String, Array, etc. You do not need to specify return types or parameter types for functions.
- A var can be anything -- any of the types mentioned, but also functions.
- Arrays are constructed very simply (no need for Processing ArrayList anymore) and have many built-in features, see this array example and more about JS arrays here
- JavaScript uses something called prototypes to form something similar to Java class objects. See this objects example and more about JS objects here.

Conversion examples

Basic sketch

This is the basic setup for a Processing and p5.js sketch. Note that p5.js will also require an empty HTML file that links to the p5.js library and your sketch file in the header (see getting started).

```
void setup() {
  // setup stuff
}

void draw() {
  // draw stuff
}
```

```
function setup() {
  // setup stuff
}

function draw() {
  // draw stuff
}
```

Converting a Processing sketch to p5.js

Here are two examples of sketches that have been converted from Processing to p5.js. The changes made are shown in the comments, all the other lines remained the same.

```
/**
 * This example can be found in the Processing examples package
 * that comes with the Processing PDE.
 * Processing > Examples > Basics > Form > Bezier
 * Adapted by Evelyn Eastmond
 */

function setup() {           // **change** void setup() to function setup()
  createCanvas(640, 360);    // **change** size() to createCanvas()
  stroke(255);               // stroke() is the same
  noFill();                  // noFill() is the same
}

function draw() {                            // **change** void draw() to function d
  background(0);                             // background() is the same
  for (var i = 0; i < 200; i += 20) {        // **change** int i to var i
    bezier(mouseX-(i/2.0), 40+i, 410, 20, 440, 300, 240-(i/16.0), 300+(i/8.0)); /
  }
}
```

```
/**
 * This example can be found in the Processing examples package
 * that comes with the Processing PDE.
 * Processing > Examples > Topics > Interaction > Follow3
 * Adapted by Evelyn Eastmond
 */

var x = new Array(20);  // **change** float[] x = new float[20] to new Array(20)
var y = new Array(20);  // **change** float[] y = new float[20] to new Array(20)
var segLength = 18;                                  // **change** float to var

function setup() {                            // **change** void setup() to functio
  createCanvas(640, 360);                     // **change** size() to createCanvas(
  strokeWeight(9);                            // strokeWeight() is the same
  stroke(255, 100);                           // stroke() is the same
  for(var i=0; i<x.length; i++) {           // initialize the array
    x[i]=0;
    y[i]=0;
  }
}

function draw() {                             // **change** void draw() to function
  background(0);                              // background() is the same
  drawSegment(0, mouseX, mouseY);             // functions calls, mouseX and mouseY
  for(var i=0; i<x.length-1; i++) {           // **change** int i to var i
    drawSegment(i+1, x[i], y[i]);             // function calls are the same
  }
}

function drawSegment(i, xin, yin) {           // **change** void drawSegment() to f
  var dx = xin - x[i];                        // **change** float to var
  var dy = yin - y[i];                        // **change** float to var
  var angle = atan2(dy, dx);                  // **change** float to var, atan2() i
  x[i] = xin - cos(angle) * segLength;        // cos() is the same
  y[i] = yin - sin(angle) * segLength;        // sin() is the same
  segment(x[i], y[i], angle);                 // function calls are the same
}

function segment(x, y, a) {                   // **change** void segment() to funct
  push();                                         // pushMatrix() becomes push()
  translate(x, y);                            // translate() is the same
  rotate(a);                                  // rotate() is the same
  line(0, 0, segLength, 0);                   // line() is the same
  pop();                                          // popMatrix() becomes pop()
}
```

Converting a p5.js sketch to Processing

Here are two examples of sketches that have been converted from p5.js to Processing. The changes made are shown in the comments, all the other lines remained the same.

```
/**
 * This example can be found in the Processing examples package
 * that comes with the Processing PDE.
 * Processing > Examples > Basics > Form > Bezier
 */

void setup() {                        // **change** function setup() to void set
  size(640, 360);                     // **change** createCanvas() to size()
  stroke(255);
  noFill();
}

void draw() {                         // **change** function draw() to void draw
  background(0);
  for (int i = 0; i < 200; i += 20) {  // **change** var i to int i
    bezier(mouseX-(i/2.0), 40+i, 410, 20, 440, 300, 240-(i/16.0), 300+(i/8.0));
  }
}
```

```
/**
 * This example can be found in the Processing examples package
 * that comes with the Processing PDE.
 * Processing > Examples > Topics > Interaction > Follow3
 * Based on code from Keith Peters.
 */

float[] x = new float[20];                    // **change** array of 0's to be
float[] y = new float[20];                    // **change** array of 0's to be
float segLength = 18;                         // **change** var to float

void setup() {                                // **change** function setup() t
  size(640, 360);                             // **change** createCanvas() to
  strokeWeight(9);
  stroke(255, 100);
}

void draw() {                                 // **change** function draw() vo
  background(0);
  dragSegment(0, mouseX, mouseY);
  for(int i=0; i<x.length-1; i++) {           // **change** int i to var i
    dragSegment(i+1, x[i], y[i]);
  }
}

void dragSegment(int i, float xin, float yin) {  // **change** function drawSegme
  float dx = xin - x[i];                      // **change** var to float
  float dy = yin - y[i];                      // **change** var to float
  float angle = atan2(dy, dx);                // **change** var to float
  x[i] = xin - cos(angle) * segLength;
  y[i] = yin - sin(angle) * segLength;
  segment(x[i], y[i], angle);
}
```

```
void segment(float x, float y, float a) {        // **change** function segment()
  pushMatrix();
  translate(x, y);
  rotate(a);
  line(0, 0, segLength, 0);
  popMatrix();
}
```

About variables

In p5.js, all variables (whether they are numbers, strings, arrays, functions, objects, whatever!) are declared using the symbol "var". In Processing, you must specify the variable type.

For example, instead of:

```
boolean button = false;
```

you'd write

```
var button = false;
```

or

instead of:

```
float x = 100.3;
```

you'd write

```
var x = 100.3;
```

Here is a summary of the supported Processing data types (table borrowed from Getting Started with Processing).

Name	Description	Range of values
int	Integers (whole numbers)	-2,147,483,648 to 2,147,483,647
float	Floating-point values	-3.40282347E+38 to 3.40282347E+38
boolean	Logical value	true or false
char	Single character	A-z, 0-9, and symbols
String	Sequence of characters	Any letter, word, sentence, and so on
PImage	PNG, JPG, or GIF image	N/A
PFont	VLW font; use the Create Font tool to make	N/A
PShape	SVG file	N/A