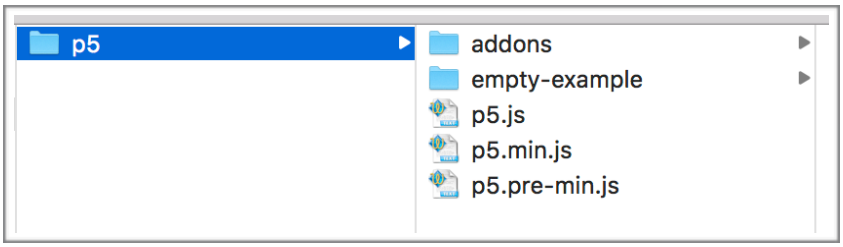
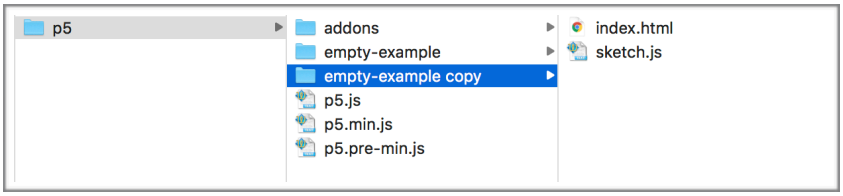
CREATING A NEW PROJECT

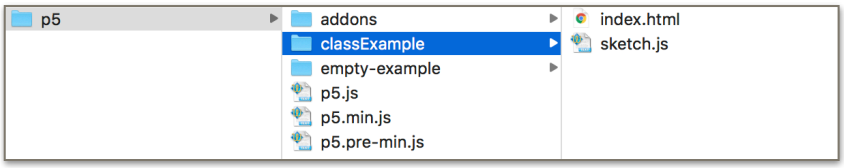
To create a new project, navigate to the “p5” folder on your computer. This folder contains the library files for p5.js, a folder called “addons,” and a folder called “empty-example”:



Within the p5 folder, duplicated the “empty-example” folder,



and rename it with your new project name:



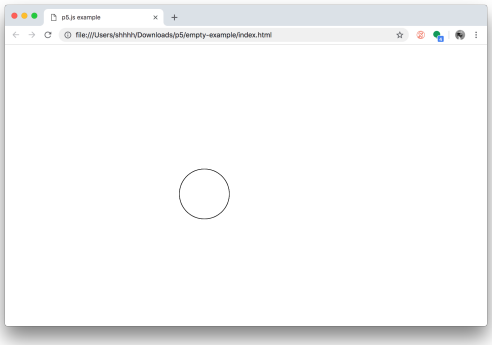
Open your “sketch.js” file in sublime text, and your “index.html” folder in a browser. When you make changes to your sketch.js file, you can simply refresh your browser to view your changes.

Check to make sure everything is working a-ok by drawing an ellipse in your p5.js folder by using the function `createCanvas(800,600);` to create the size of your working area, and the function `ellipse(400,300,100,100);` to draw a circle in the middle of your canvas (as described left).

```
1
2   function setup() {
3     // put setup code here
4     createCanvas(800,600);
5   }
6
7   function draw() {
8       ellipse(400,300,100,100);
9     // put drawing code here
10  }
```

When you refresh your index.html page in the browser, you should see a circle appear toward the center of the page.

Now you’re ready to get started on your new project!



BASIC DRAWING FUNCTIONS

Below are some helpful functions in p5.js for drawing to the screen with shapes and colors. Additional functions can be found at p5js.org/reference/ but this contains some of the most basic functions to get your started with drawing.

SETTING UP YOUR SKETCH

```
function setup() { }          //Anything inside of these brackets will occur once, when the page
                              loads. createCanvas(); (below) usually belongs here.

function draw() {}            //Anything inside of these brackets will repeat continuously, over
                              and over again, until you close the page. This is typically where
                              most of your sketch will go, especially anything that moves or
                              changes over time.

createCanvas(100, 50);        //This sets up ta canvas element in your html page, and the two
                              parameters determine the width and height in pixels.
```

DRAWING SHAPES

```
ellipse(56, 46, 55, 55);      //Draws an ellipse. First two parameters indicate the x,y position
                              of the ellipse center, 3rd and 4th parameters determine the width
                              and height. If the width and height are the same size, you will
                              draw a circle.

rect(30, 20, 55, 55);         //Draws a rectangle. First two parameters indicate the x,y position
                              of the rectangle's top left corner, 3rd and 4th parameters
                              determine the width and height. If the width and height are the
                              same size, you will draw a square.

line(30, 20, 85, 75);         //Draws a line between two points. The first two parameters are the
                              x,y positions of the first point, the 3rd and 4th parameters are
                              the x,y positions of the second line.

triangle(30, 75, 58, 20, 86, 75); //Draws a triangle by connecting three points. The parameters
                              are three sets of x,y positions of these three points
```

DRAWING CUSTOM SHAPES

```
//To draw a custom shape, you can connect a series of points using the vertex() function to
indicate the x,y position of each point. The beginShape function indicates the start of the
shape, while endShape(); indicates the end. By including the parameter CLOSE in endShape(), your
final point will connect to your first point to close the shape.

beginShape();          //begin the custom shape
vertex(30, 20);        //establish the x,y positions of all the points to connect into a shape
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);        //end the custom shape, and close it by connecting the last point to the
                        first point
```

STROKE

When you draw a shape, you'll notice a thin, 1 pixel black outline around it. This outline is called a **stroke**. You can style the stroke in a number of ways, changing the width, color, or removing it all together. Here are some functions:

```
noStroke();             // This will remove the outlines of any shapes you draw. If you put it in
                        your setUp function, it will affect all shapes.

strokeWeight(1);        //The default stroke is 1 pixel wide. Increase this value for thicker
                        lines.

stroke(51,100,255);     //This changes the color of your outline. One value between 0-255 will
                        adjust the grayscale value, 3 values between 0-255 each will adjust the
                        red, green, and blue values to create a color. See the next section for a
                        description of color.
```

COLOR

Working with digital color is a little bit different than working with pigments, such as paint, because we are working with **additive color**. Additive color is a method of creating color by mixing light.

The primary colors in additive color are red, green, and blue. You can think of these as three channels of light - a red light, a green light, and a blue light. Each pixel on the screen will have a certain amount of red, green, and blue light (plus a certain amount of transparency or opacity) to create a color.

To create the color white, all three colors are turned on at the highest power - in our case, a value of 255.
To create the color black, all three colors are turned off (like turning off a light bulb) - now, a value of 0.

For any color function, if we indicate only 1 value between 0-255, we will create a color on grayscale, between black (0) to white (255). For example:

```
background(100); //This will color the background of our sketch as a gray color
```

To mix colors, we simply adjust the values of each channel, adjusting how much red, green, or blue to include in our color, using values from 0 - 255. Using the background as the example, again:

```
background(255,0,0); //For a red background (red = 255, green = 0, blue = 0)
background(0,255,0); //A green background
background(0,0,255); //A blue background
```

We can use the function `fill()` to color any shape that follows. For example:

```
fill(255,100,0,); //This will fill any shape that follows the color orange
ellipse(200,400,100,100); //In this case, this circle.
```

Adding a fourth parameter will indicate opacity. So, adding a fourth parameter at a value of 50 will change the opacity of the circle to become much more transparent:

```
fill(255,100,0,50); //The last value, 50, will make the circle more transparent.
ellipse(200,400,100,100);
```

Try drawing a few shapes that overlap, to see transparency in full effect.

COLOR FUNCTIONS

```
background();           //changes the color of the canvas background
fill();                 //changes the color of shapes the follow it
stroke();               //changes the color of lines or shape outlines that follow it
```