

Introduction to Creative Coding

The p5.js Library

p5.js is a JavaScript library for creative coding. A collection of pre-written code, it provides us with tools that simplify the process of creating interactive visuals with code in the web browser.

p5.js Project Structure

A p5.js project is like any other web project—it utilizes HTML, CSS, and JavaScript. A typical p5.js project includes the p5.js library, **index.html**, **style.css**, and **sketch.js**.

Including the p5.js Library

The p5.js library must be included using a `<script>` tag in the `<head>` section of an HTML document. Only then, the p5.js library can be used in a JavaScript file.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Include p5.js library -->
    <script src="p5.js"></script>
    <link rel="stylesheet"
type="text/css" href="style.css">
    <meta charset="utf-8" />
  </head>
  <body>
    <script src="sketch.js"></script>
  </body>
</html>
```

The p5.js Canvas

The `<canvas>` element is an HTML element that renders graphics created with JavaScript's Canvas API. Utilizing the Canvas API behind the scenes, the p5.js library provides many built-in drawing functions that simplify drawing to the HTML `<canvas>` on the fly using JavaScript.

The `createCanvas()` Function

The `createCanvas()` function creates an HTML canvas on the web page, taking the desired canvas width and height as arguments. Typically, it is one of the first functions called in the `setup()` function. The `createCanvas()` function can only be called once within a p5.js sketch.

```
function setup() {  
  // Creates a canvas with 800px width  
  and 600px height  
  createCanvas(800, 600);  
}
```

The Default Canvas

The p5.js library will create a default canvas with a width of 100 pixels and a height of 100 pixels when no arguments are provided for the `createCanvas()` function.

The `noCanvas()` Function

If, for some reason, your p5.js sketch does not require a canvas, explicitly call the `noCanvas()` function to stop the p5.js library from creating a canvas at the start of the program.

```
function setup() {  
  // No canvas element will be created  
  for this p5.js sketch  
  noCanvas();  
}
```

The `background()` Function

The `background()` function sets the background color of the p5.js canvas. The background of a p5.js canvas is transparent by default.

```
function setup() {  
  // Sets background to a gray color  
  background(127);  
}
```

Using `background()` with One Argument

When the `background()` function is called with a numeric argument between 0 and 255, the background color will be set to a grayscale value, with 0 being pure black and 255 being pure white.

```
function setup() {  
  // Sets the background color to white  
  background(255);  
}
```

The `setup()` Function

At the beginning of a p5.js program, the p5.js library automatically executes the `setup()` function. The `setup()` function should not be explicitly called in the sketch.

```
function setup() {  
  // Runs once at the beginning of  
  the p5.js sketch  
}
```

Uses of the `setup()` Function

The `setup()` function typically contains code that defines the initial state of the sketch, such as the canvas size, background color, and initial values of global variables.

```
let beginSize;  
  
// Initializing the canvas size,  
background color and beginlSize value  
function setup() {  
  createCanvas(800, 600);  
  background(220);  
  beginSize = 5;  
}
```

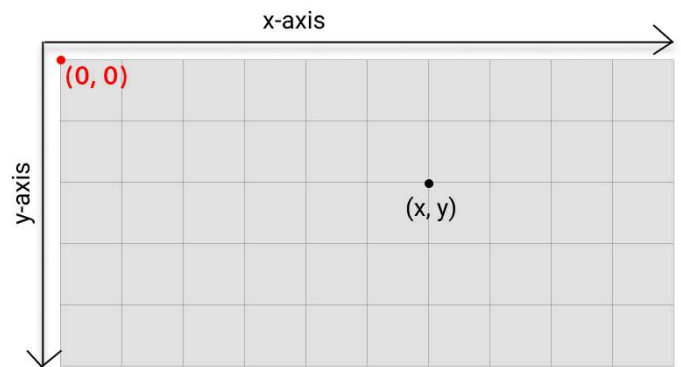
The `draw()` Function

The `draw()` function is automatically called after the `setup()` function, which runs once at the program's start. The `draw()` loop infinitely runs the code block inside the function from top to bottom.

```
function setup() {  
  // Runs once at the start of the  
  program  
}  
  
function draw() {  
  // Loops infinitely after setup() is  
  run  
}
```

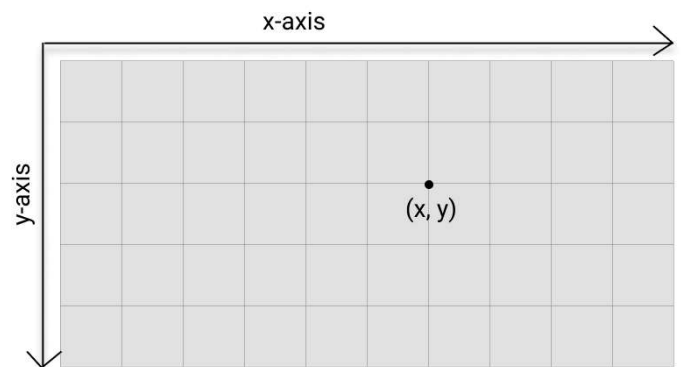
The Canvas Origin

The p5.js canvas uses a coordinate system to describe space. The origin, (0, 0), of the canvas is the top-left corner of the canvas.



The Coordinate System


The canvas coordinate system is described using ordered pairs, `(x, y)`, where the `x` coordinate is the distance from the left edge of the canvas and the `y` coordinate is the distance from the top edge of the canvas.



Colors

A color value can be represented in various ways with p5.js. It can be given as:

- Gray value as one numeric value between 0 and 255.
- RGB (Red, Green, Blue) value as three numeric values between 0 and 255.
- RGBA (Red, Green, Blue, Alpha) value as four numeric values between 0 and 255.
- Hexadecimal value as a string.
- HSB (Hue, Saturation, Brightness) value as three numeric values between 0 and 360 for hue and between 0 and 100 for saturation and brightness.



```
// Grayscale Integer Value
background(50);

// R, G, B Integer Value
background(249, 102, 255);

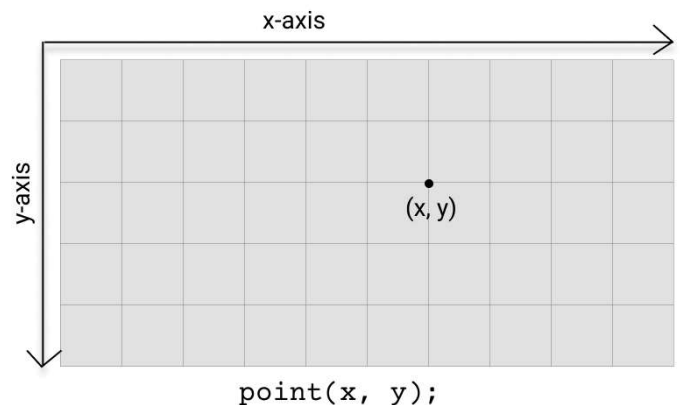
// Named SVG/CSS color string
background('DodgerBlue');

// Hexadecimal Integer Value
background('#FF8C00');
```

The point() Function

The `point()` function draws a single pixel at specified coordinates. It takes two arguments where the first argument is the x coordinate, and the second argument is the y coordinate.

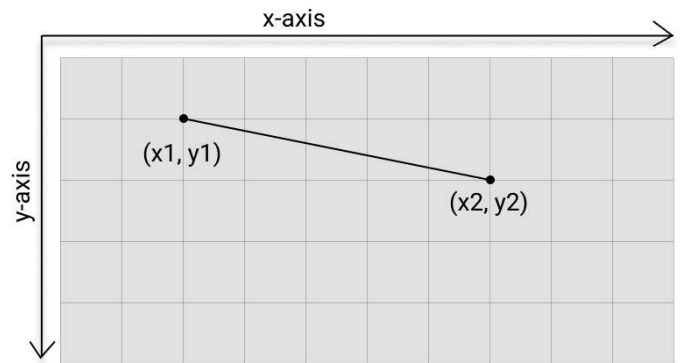
The color of the point can be changed with the `stroke()` function. The size of the point can be changed with the `strokeWeight()` function.



The `line()` Function

The `line()` function draws a line between two points and requires four arguments: the x and y positions for each endpoint.

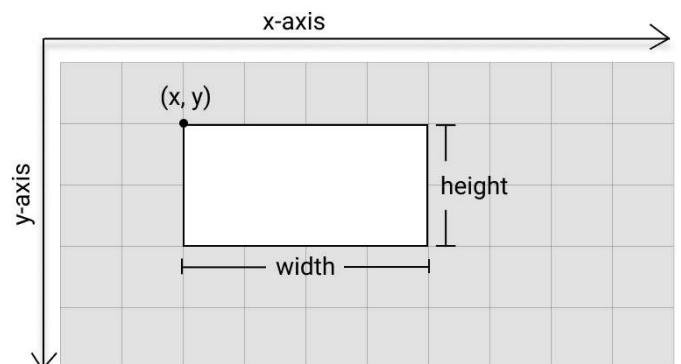
The width of the line can be set with the `strokeWeight()` function and the color of the line can be set with the `stroke()` function.



```
line(x1, y1, x2, y2);
```

The `rect()` Function

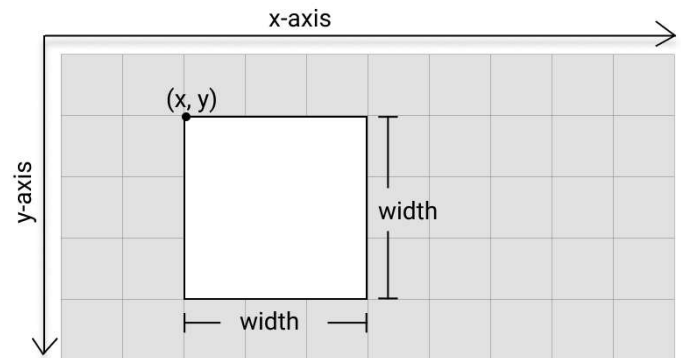
The `rect()` function draws a rectangle to the canvas. It takes four arguments: the first two arguments are the x and y positions of the top left corner of the rectangle. The third argument is the width, and the fourth argument is the height.



```
rect(x, y, width, height);
```

The `square()` Function

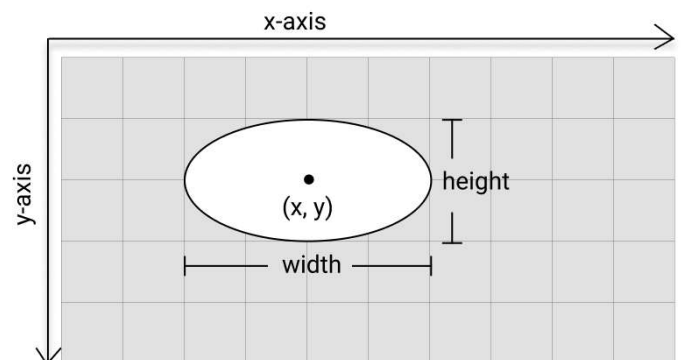
The `square()` function draws a square to the canvas. It has three required arguments: the first two arguments are the `x` and `y` positions of the top left corner of the square. The third argument is the width (and height) of the square.



```
square(x, y, width);
```

The `ellipse()` Function

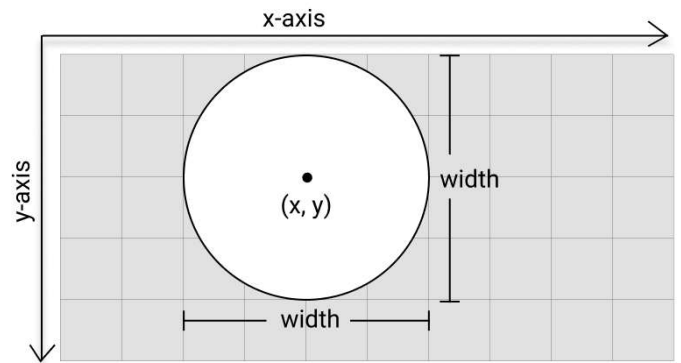
The `ellipse()` function draws an ellipse to the canvas. It requires four arguments where the first and second arguments are the `x` and `y` positions of the center of the ellipse. The third and fourth arguments are the width and height of the ellipse.



```
ellipse(x, y, width, height);
```

The `circle()` Function

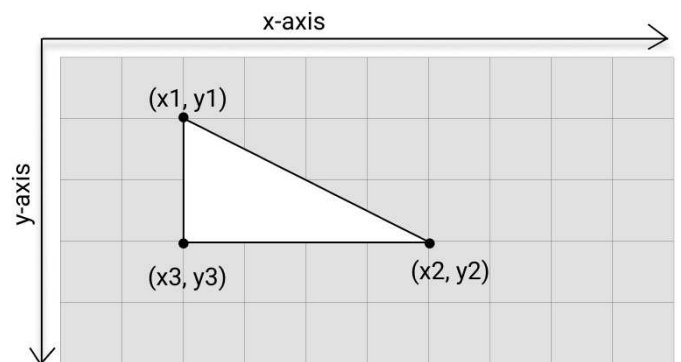
The `circle()` function draws a circle to the canvas. It has three required arguments where the first and second arguments are the x and y positions of the center of the circle. The third argument is the width (and height) of the circle.



```
circle(x, y, width);
```

The `triangle()` Function

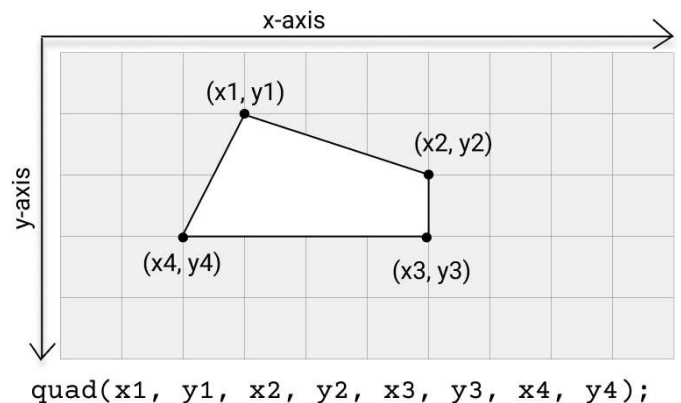
The `triangle()` function draws a triangle to the canvas. It has six required arguments: the x and y positions for each of the triangle's three vertices.



```
triangle(x1, y1, x2, y2, x3, y3);
```


The `quad()` Function

The `quad()` function draws a quadrilateral to the canvas. It has eight required arguments: the x and y positions for each of the four vertices.



width and height

`width` is a built-in variable that returns the width of the canvas, and the `height` variable returns the height of the canvas.

```
function setup() {  
  createCanvas(400, 800);  
  console.log(width); // Logs 400 to  
  console  
  console.log(height); // Logs 800 to  
  console  
}
```

The `background()` Function

The `background()` function sets the color used for the background of the p5.js canvas. The default background is transparent. This function is typically used within the `draw()` function to clear the canvas at the beginning of each frame, but it can be used inside the `setup()` if the background needs to be only set once.

```
function draw() {  
  // Sets a red background color  
  background(255, 0, 0);  
}
```

The fill() Function

The `fill()` function sets the color used to fill a shape with the specified color. It must be called prior to drawing the shape. The default fill color is white.

```
function draw() {  
  // Sets the fill color of the circle to  
  blue  
  fill(0, 0, 255);  
  circle(100, 100, 25);  
}
```

The noFill() Function

The `noFill()` function sets the fill color of a shape as transparent. It must be called before drawing the shape.

```
function draw() {  
  // Sets the square to have transparent  
  fill  
  noFill();  
  square(50, 50, 25);  
}
```

The stroke() Function

The `stroke()` function sets the stroke color used for a shape to the specified color. It must be called before drawing the shape. The default stroke color is black.

```
function draw() {  
  // Sets stroke color of the square to  
  green  
  stroke(0, 255, 0);  
  square(50, 50, 25);  
}
```

The strokeWeight() Function

The `strokeWeight()` function sets the width of a shape's stroke to specified pixels. It must be called before drawing the shape. The default stroke weight is 1 pixel.

```
function draw() {  
  // Draws a line of 5px thickness  
  strokeWeight(5);  
  line(50, 25, 50, 75);  
}
```

The `noStroke()` Function

The `noStroke()` function disables the stroke of a shape. It must be called before drawing the shape.

```
function draw() {  
  // Draws a circle with blue fill color  
  and no stroke/outline  
  noStroke();  
  fill(0, 0, 255);  
  circle(50, 50, 25);  
}
```

Order of Shapes

The order in which shape functions are called is important, as the shape function called **last** will be rendered on top of previously drawn shapes.

```
function draw() {  
  // The square will appear on top of the  
  circle  
  circle(100, 100, 100);  
  square(50, 50, 100);  
}
```

for Loops

A `for` loop can be used to draw multiple shapes at once which is useful to create patterns of shapes. The iterator variable of the `for` loop can be used for the `x` and `y` positions to draw shapes next to each other.

```
function draw() {  
  // Draw a 5x5 grid of circles that are  
  25px apart horizontally and vertically  
  for(let posX = 0; posX < 5; posX++) {  
    for(let posY = 0; posY < 5; posY++) {  
      circle(posX * 25, posY * 25, 10);  
    }  
  }  
}
```

 **Print**  **Share** ▼