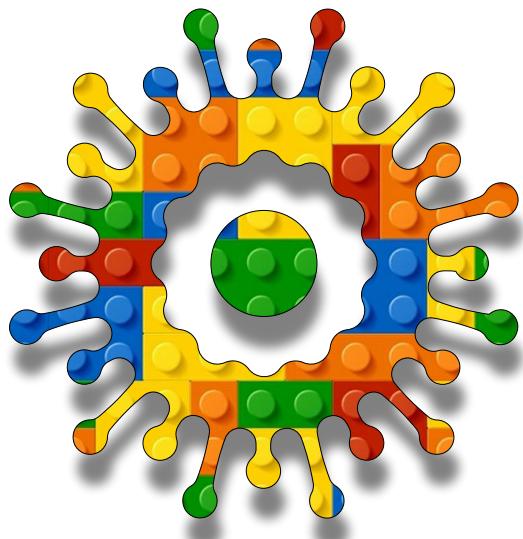


Artificial
Intelligence
Module B
Unit #1

p5.js code
snippets 3





Module B Unit #1 p5.js code snippets 3

Introduction to code snippets part 3

The index.html file

Sketch B1.1 drawing a rectangle

Sketch B1.2 drawing an ellipse

Sketch B1.3 drawing triangles

Sketch B1.4 measuring the distance

Sketch B1.5 in the name of the colour



Introduction to code snippets part 3

We are going to cover some essential coding snippets relevant to the next section on pre-trained models.



Sketch B1.1 drawing a rectangle

We have covered most of the shapes you will use or need except for this one, the **rectangle**. A rectangle has four arguments. The first two are similar to those of the square, and they are the co-ordinates of the rectangle; the third is the width of the rectangle, and the fourth is the height of the rectangle.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  rect(100, 100, 200, 50)
  rect(100, 200, 20, 100)
  rect(200, 200, 100, 150)
}
```

Notes

Drawing three rectangles, the arguments are:

-  x position
-  y position
-  horizontal dimension
-  vertical dimension



Challenge

Add the `rectMode()` to put the co-ordinates in the centre of the rectangle.

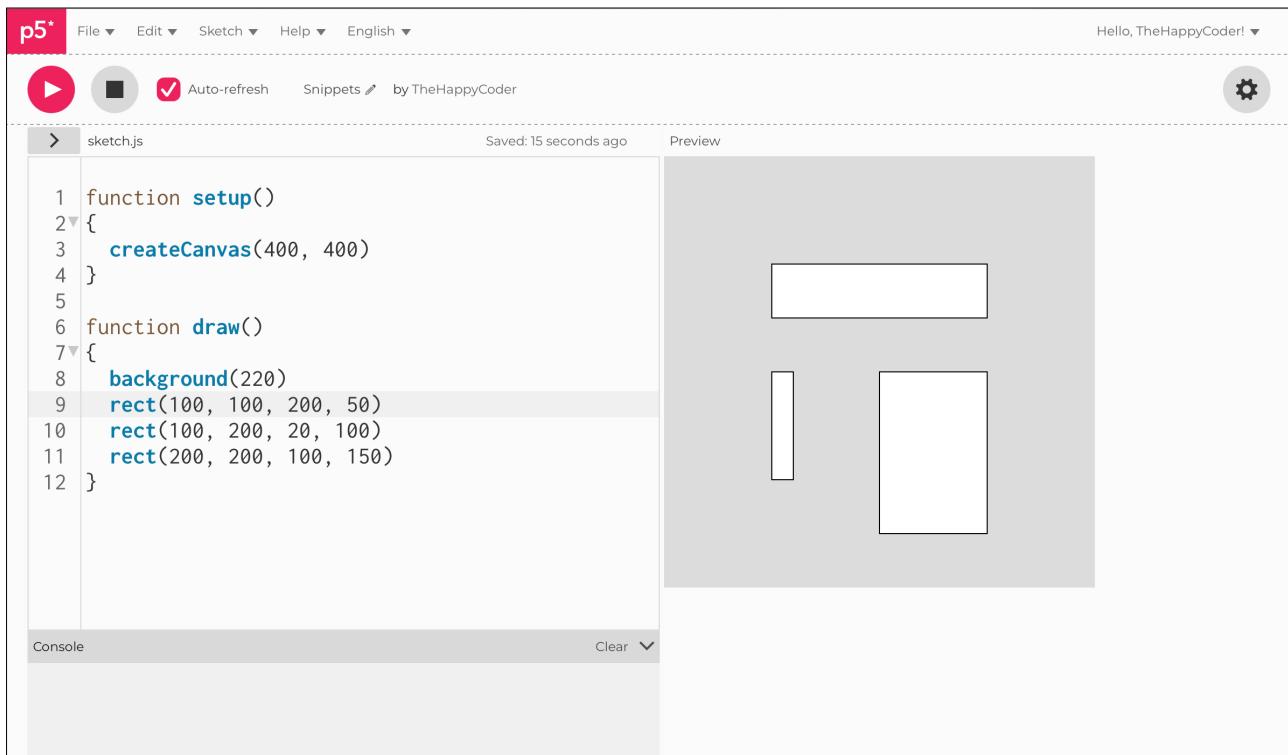


Code Explanation

<code>rect(100, 100, 200, 50)</code>	Draws a rectangle at top left hand corner 100 from the left, 100 down and with a width of 200 and height of 50
--------------------------------------	--

Draws a rectangle at top left hand corner 100 from the left, 100 down and with a width of 200 and height of 50

Figure B1.1 drawing rectangles



The screenshot shows the p5.js web editor interface. At the top, there are buttons for File, Edit, Sketch, Help, and English, along with a user greeting 'Hello, TheHappyCoder!'. Below the menu is a toolbar with a play button, a square, and an auto-refresh checkbox (which is checked). The code editor on the left contains 'sketch.js' with the following code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background(220)
9     rect(100, 100, 200, 50)
10    rect(100, 200, 20, 100)
11    rect(200, 200, 100, 150)
12 }
```

The preview window on the right shows the output of the code. It features a light gray background with three white rectangles: a large horizontal rectangle at the top, a tall narrow rectangle on the left, and a medium-sized square on the right.



Sketch B1.2 drawing an ellipse

Another shape that we can draw is an **ellipse**, which is similar to a circle with an extra argument.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  ellipse(200, 100, 150, 50)
  ellipse(100, 200, 20, 100)
  ellipse(200, 250, 100, 200)
  ellipse(300, 200, 20, 100)
}
```

Notes

We draw four ellipses, the arguments are:

-  x position
-  y position
-  horizontal dimension
-  vertical dimension

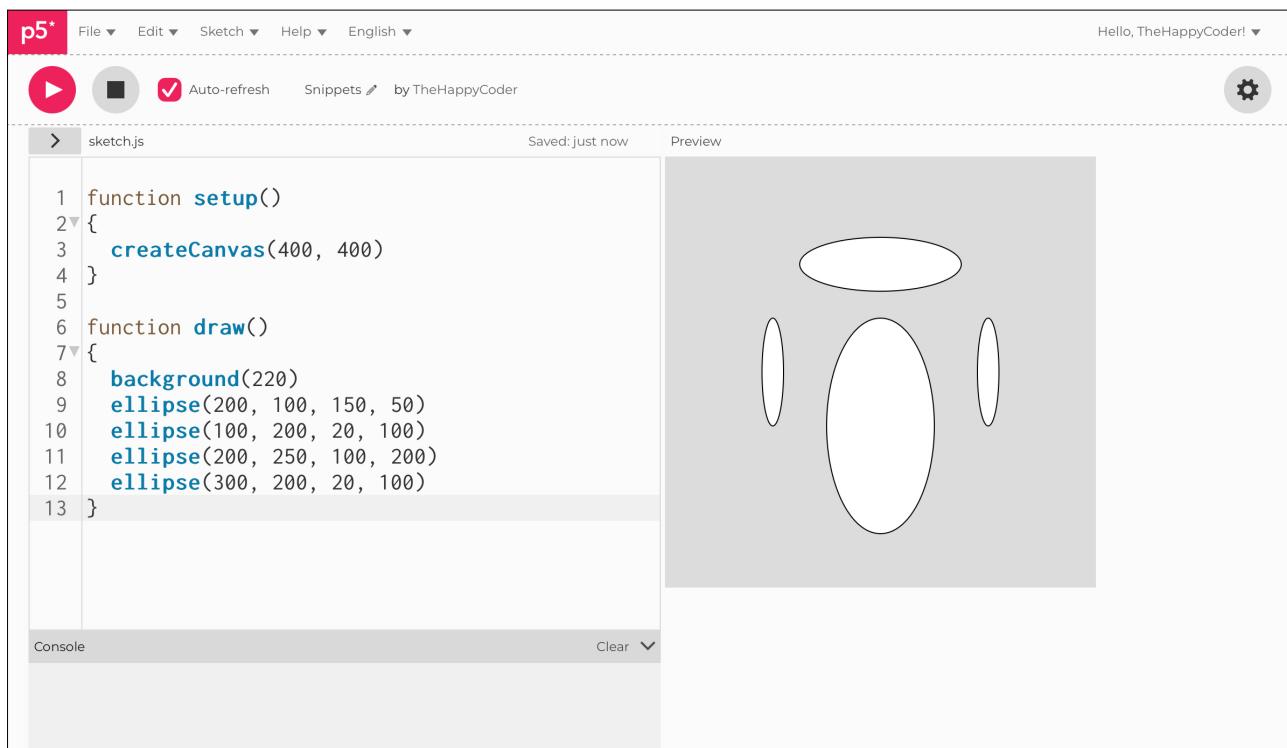
Challenge

Make a pattern with them

Code Explanation

<code>ellipse(200, 100, 150, 50)</code>	First two arguments are the x and y co-ordinates, third is the width of ellipse and the fourth is the height of the ellipse
---	---

Figure B1.2



The screenshot shows the p5.js IDE interface. The top bar includes the p5 logo, File, Edit, Sketch, Help, English, and a user profile. The main area has tabs for sketch.js and Preview. The code editor on the left contains the following JavaScript code:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8   background(220)
9   ellipse(200, 100, 150, 50)
10  ellipse(100, 200, 20, 100)
11  ellipse(200, 250, 100, 200)
12  ellipse(300, 200, 20, 100)
13 }
```

The Preview window on the right displays a gray square containing five white ellipses. One large ellipse is centered at (200, 100) with a bounding box of 150x50. A small ellipse is at (100, 200) with a bounding box of 20x100. A medium ellipse is at (200, 250) with a bounding box of 100x200. Two very small ellipses are at (300, 200) with a bounding box of 20x100 each.



Sketch B1.3 drawing triangles

The triangle has six arguments for the three co-ordinates at each corner.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  triangle(200, 100, 100, 190, 300, 190)
  triangle(200, 300, 100, 210, 300, 210)
}
```

Notes

The triangle requires a bit more thought organising the co-ordinates. I often sketch them out first on a piece of paper to help plan them out. The arguments are:

- x1 position
- y1 position
- x2 position
- y2 position
- x3 position
- y3 position

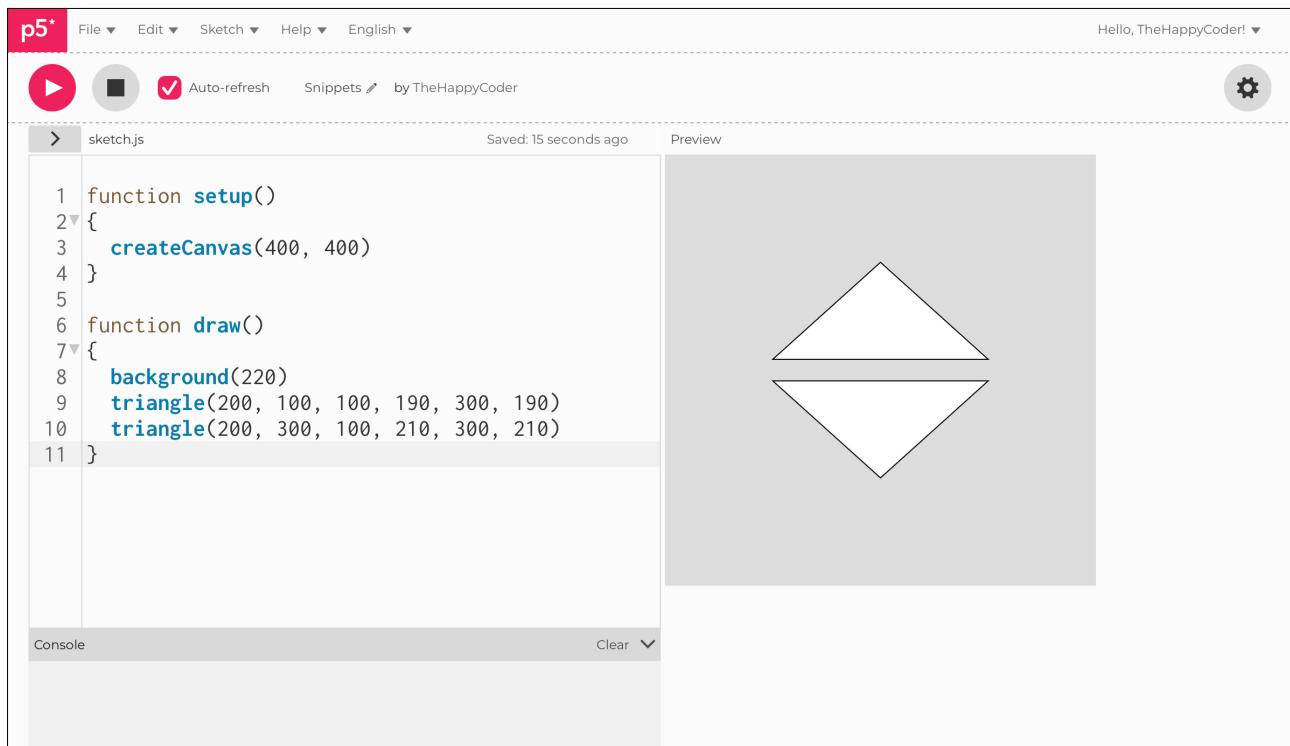
Challenge

Make some more

Code Explanation

<code>triangle(200, 100, 100, 190, 300, 190)</code>	The three co-ordinates for the three corners of the triangle
---	---

Figure B1.3



The screenshot shows the p5.js code editor interface. At the top, there are navigation buttons for File, Edit, Sketch, Help, and English, along with an Auto-refresh checkbox and a Snippets button. The user is identified as "Hello, TheHappyCoder!". Below the header, the code editor shows a file named "sketch.js" with the following content:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8   background(220)
9   triangle(200, 100, 100, 190, 300, 190)
10  triangle(200, 300, 100, 210, 300, 210)
11 }
```

The preview window on the right displays a gray canvas with two white triangles. The top triangle is positioned in the upper right quadrant, and the bottom triangle is positioned in the lower right quadrant. Both triangles have vertices at (200, 100), (100, 190), and (300, 190) for the top one, and (200, 300), (100, 210), and (300, 210) for the bottom one.



Sketch B1.4 measuring the distance

! Starting a new sketch

If we want to know the distance between two points, for instance, the centre of two circles, we can use a `dist()` function that calculates the distance between those two points. This is useful when looking at collisions or in our `ml5.js` example later using it to draw a circle. Here we are measuring the distance between two circles and writing the value on the canvas.

```
let x1
let y1
let x2
let y2

function setup()
{
  createCanvas(400, 400)
  textSize(32)
}

function draw()
{
  background(200)
  x1 = width/2
  y1 = height/2
  x2 = mouseX
  y2 = mouseY
  let d = dist(x1, y1, x2, y2)
  text(d, 50, 50)
  circle(x1, y1, 25)
  circle(x2, y2, 25)
}
```

Notes

Calculates the distance between two points. The version of `dist()` with four parameters calculates distance in two dimensions. The version of `dist()` with six parameters calculates distance in three dimensions; this is handy if you are using 3D shapes.

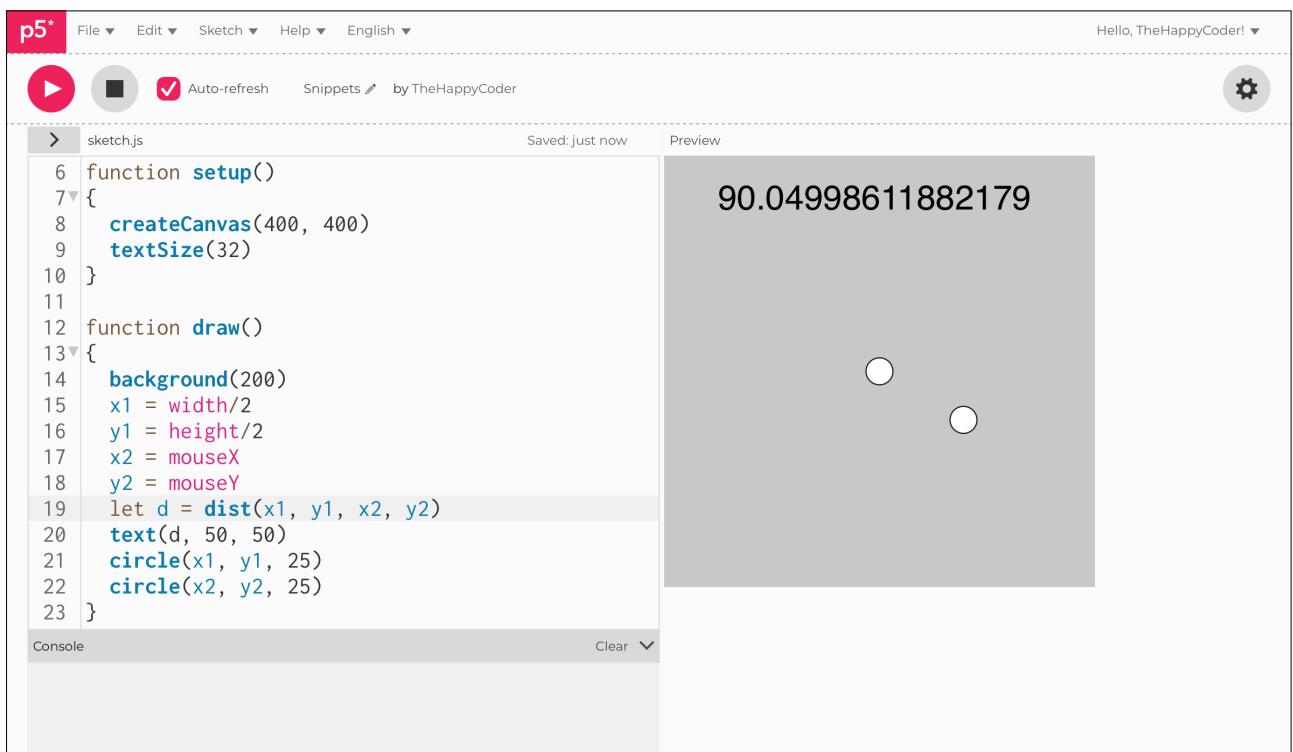
-  x1 position of point 1
-  y1 position of point 1
-  x2 position of point 2
-  y2 position of point 2

To make it just an integer replace with: `text(int(d), 50, 50)`

Code Explanation

let d = dist(x1, y1, x2, y2)	Calculating the distance (d) between two pairs of co-ordinates (x1, y1) and (x2, y2)
------------------------------	--

Figure B1.4



The screenshot shows the p5.js code editor interface. At the top, there is a menu bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right of the menu is a greeting 'Hello, TheHappyCoder! ▾'. Below the menu is a toolbar with icons for play, stop, and refresh, followed by 'Auto-refresh' and 'Snippets' (with a checkmark). On the far right of the toolbar is a gear icon. The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. The 'sketch.js' section contains the following code:

```
function setup() {
  createCanvas(400, 400)
  textSize(32)
}

function draw() {
  background(200)
  x1 = width/2
  y1 = height/2
  x2 = mouseX
  y2 = mouseY
  let d = dist(x1, y1, x2, y2)
  text(d, 50, 50)
  circle(x1, y1, 25)
  circle(x2, y2, 25)
}
```

The 'Preview' section shows a gray canvas with a white background. It displays the value '90.04998611882179' in black text at the top center. Two small white circles are drawn on the canvas: one at the center (x1, y1) and another at the mouse position (x2, y2).



Sketch B1.5 in the name of the colour

! Starting another new sketch

We can, if we wish, just use the name of the colours. There is a wide range of colours, too numerous to mention them here. The name must be in speech marks.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background('orange')
  fill('green')
  circle(100, 100, 100)
}
```

Notes

When you put the name of the colour in speech marks, it gives you a square indicator to the colour. There are quite a few colour names. If you use 'lightgreen' with no gap, you get light green. Some colours will take the word dark, for instance, **darkred**. There are other names such as **teal**, **magenta**, and so on. You can use single or double quotes. It is useful if you just want a simple colour rather than trying to remember the **RGB** values.

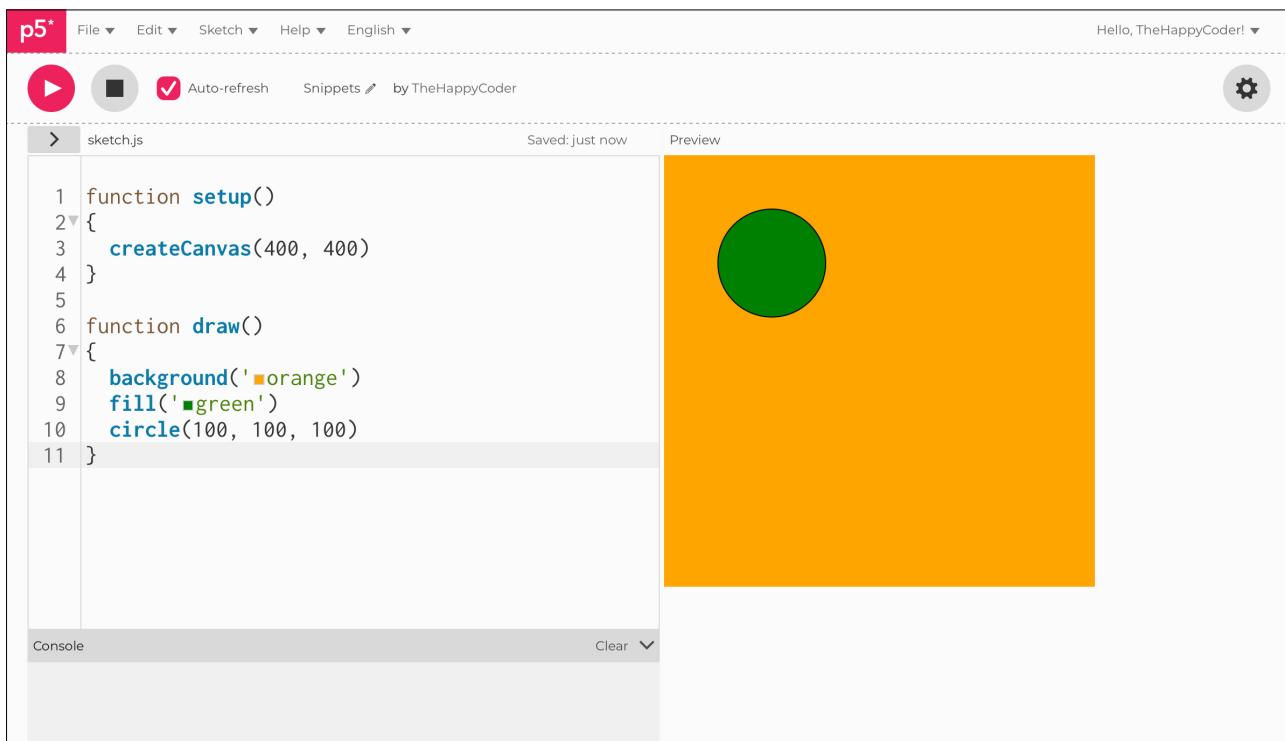
Challenges

1. Just experiment.
2. Look at the **Creative Coding module B unit #6 colour charts and pickers** in the resources section of www.elegantAI.org.

🛠️ Code Explanation

background('orange')	Gives you an orange background
fill('green')	Fills the circle green

Figure B1.6



The screenshot shows the p5.js code editor interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), a user profile (Hello, TheHappyCoder!), and a gear icon for settings. The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background('orange')
9     fill('green')
10    circle(100, 100, 100)
11 }
```

The 'Preview' tab shows a yellow square canvas with a single green circle centered at (100, 100) with a radius of 100 pixels. The bottom bar has a 'Console' tab and a 'Clear' dropdown.