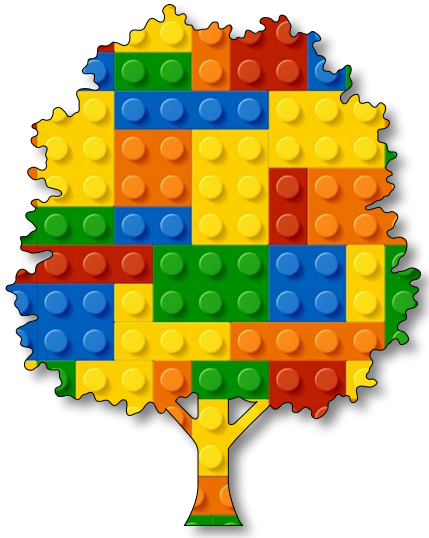
Algorithmic Art

Module B

Unit #7

arrays



Content

Module B Unit #7 Arrays

Introduction to arrays

An array is a key and vital part of coding. It is a way of storing data so that it can be accessed, added to, or altered at a later date. One way to think of it is as a series of boxes that can hold bits of data (whether numbers or words). An array has a numbering system for each box, called an `index`. In coding, counting starts with `zero`, not `one`. So the first box is `index 0`, the second box is `index 1`, the third is `index 2`, and so on (see Fig.1). You can imagine that it can be confusing that the third box is `index 2`.

The format to identify an array is square brackets such as `[23, 15, 37, 42, ...]` so we can describe this array as follows (see fig.2):
`index[0]` is `23`,
`index[1]` is `15`,
`index[2]` is `37`, and
`index[3]` is `42`, etc.

You may be wondering what arrays may have to do with creative coding; a lot is the quick answer, for there will be instances where you will want to store information, for instance, colour names or other values for accessing later.

Figure: 1 index numbering system

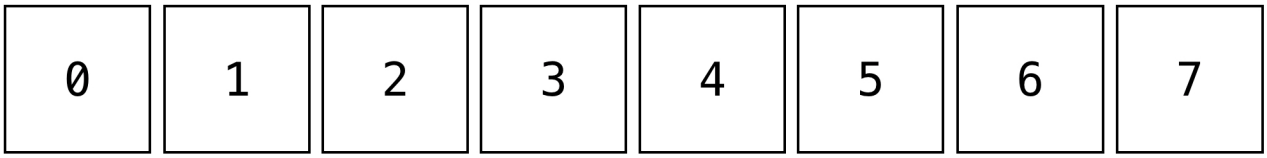
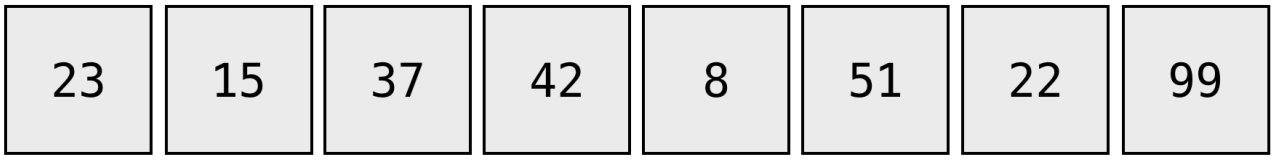


Figure 2: the values (elements) inside each box



On the top row are the `index[]` references and on the bottom row are the actual values at those reference points. We need to give the array a name. We can use let to define the array, such as:

`let numbers = []`	This is an empty array.

We can initialise it with some initial values if we want.

`let numbers = [23, 15, 37, 42, 8, 51, 22, 99]`	The array has now got some values in it

Sketch B7.1 starting sketch

❗ new sketch
A simple sketch with a circle at (100, 100) with a diameter of 46.

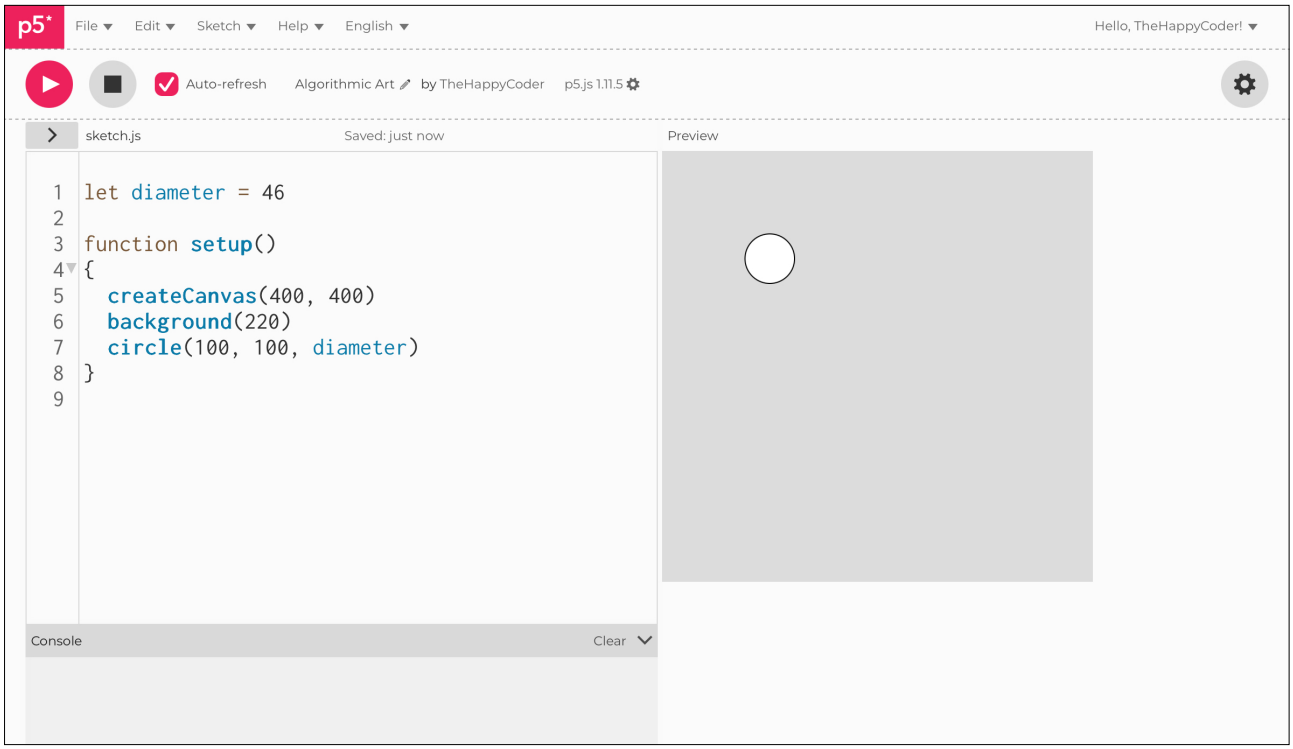
```
let diameter = 46


function setup()
{
  createCanvas(400, 400)
  background(220)
  circle(100, 100, diameter)
}
```

Notes

We are doing all of this in the setup() function because later on we don't want to loop in the draw() function.

Figure B7.1



Sketch B7.2 an array

We will put the `diameter` in an array. To get at the diameter, we will need to access the array specifying the index number, which in this case will be index `0`. The array is called `diameter` and it is identified as an array by the square brackets `[]`.

```
let diameter = [46]


function setup()
{
  createCanvas(400, 400)
  background(220)
  circle(100, 100, diameter[0])
}
```

🗒️ Notes

Well done, you have created your first array.

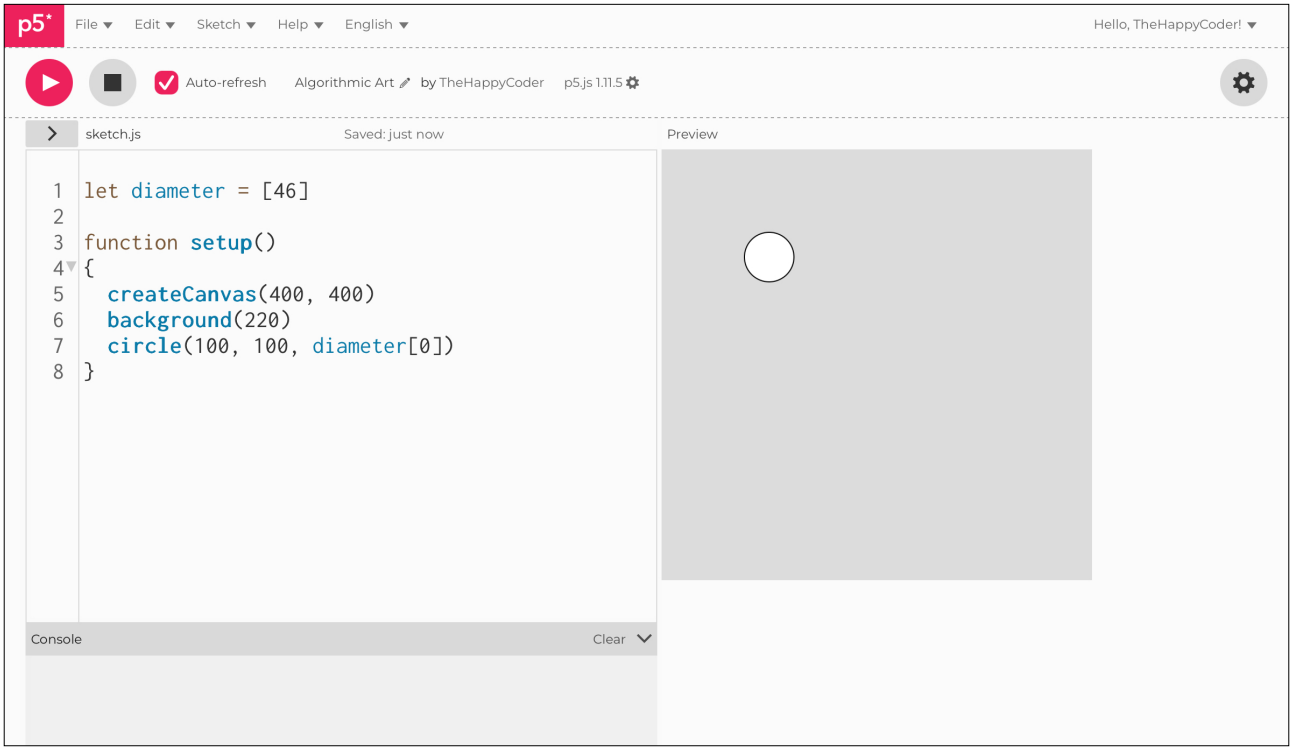
🌻 Challenge

What happens if you put a `1` instead of `0` in the index (`diameter[1]`)?

🛠️ Code Explanation

`let diameter = [46]`	To make an array we use square brackets `[]`
`circle(100, 100, diameter[0])`	The array has one element at index `[0]`

Figure B7.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let diameter = [46]
function setup()
{
createCanvas(400, 400)
background(220)
circle(100, 100, diameter[0])
}
Console
Clear

Sketch B7.3 a bigger array

We can put more numbers (elements) in this array and then draw each circle, spacing them out.

```
let diameter = [46, 12, 33, 18, 27]


function setup()
{
  createCanvas(400, 400)
  background(220)
  circle(100, 100, diameter[0])
  circle(100, 150, diameter[1])
  circle(100, 200, diameter[2])
  circle(100, 250, diameter[3])
  circle(100, 300, diameter[4])
}
```

🗒️ Notes

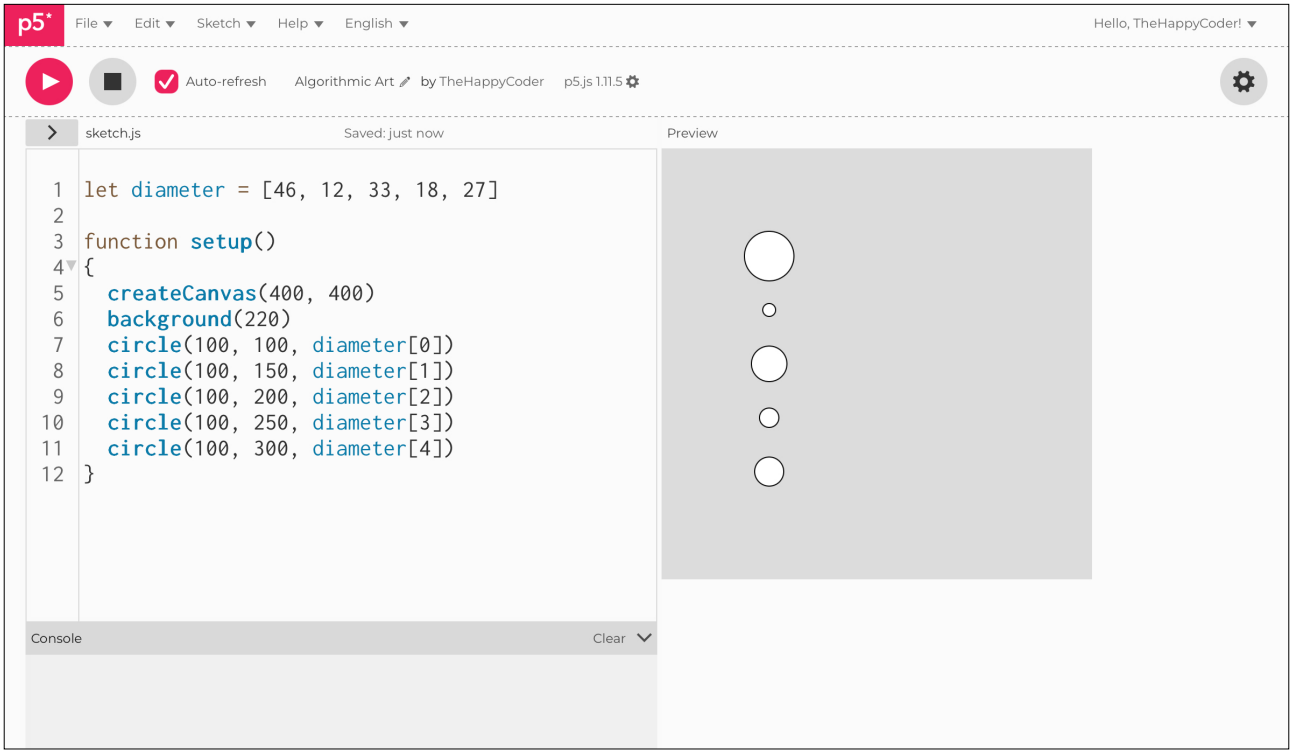
Notice we have to give an index number for each element of the array:

```
index [0] is 46
index [1] is 12
index [2] is 33
index [3] is 18
index [4] is 27
```

🛠️ Code Explanation

`let diameter = [46, 12, 33, 18, 27]`	An array holding five elements

Figure B7.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let diameter = [46, 12, 33, 18, 27]
function setup()
{
createCanvas(400, 400)
background(220)
circle(100, 100, diameter[0])
circle(100, 150, diameter[1])
circle(100, 200, diameter[2])
circle(100, 250, diameter[3])
circle(100, 300, diameter[4])
}
Console
Clear

Sketch B7.4 looping through the array

This isn't very efficient, so let's use a `for()` loop.

```
let diameter = [46, 12, 33, 18, 27]

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 5; i++)
  {
    circle(100, 100, diameter[i])
  }
}
```

🗒️ Notes

We have them all on top of each other.

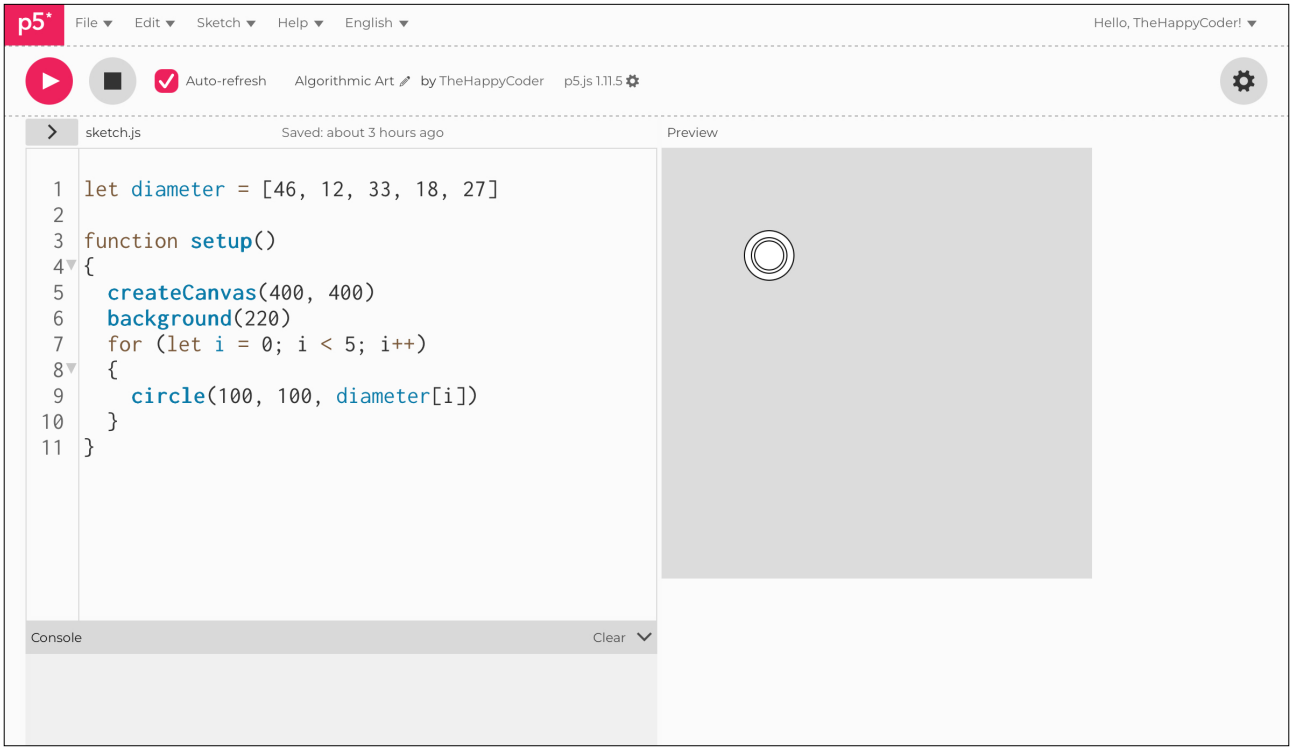
🌻 Challenge

How could we separate them?

🛠️ Code Explanation

`circle(100, 100, diameter[i])`	We loop through each element one at a time incrementing i by 1 each iteration

Figure B7.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: about 3 hours ago
Preview
let diameter = [46, 12, 33, 18, 27]
function setup()
{
createCanvas(400, 400)
background(220)
for (let i = 0; i < 5; i++)
{
circle(100, 100, diameter[i])
}
}
Console
Clear

Sketch B7.5 another array

We add another array for the y position of the circles. This allows us to space them back out again. We use the same principle for the y array.

```
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 5; i++)
  {
    circle(100, y[i], diameter[i])
  }
}
```

🗒️ Notes

We are back where we were.

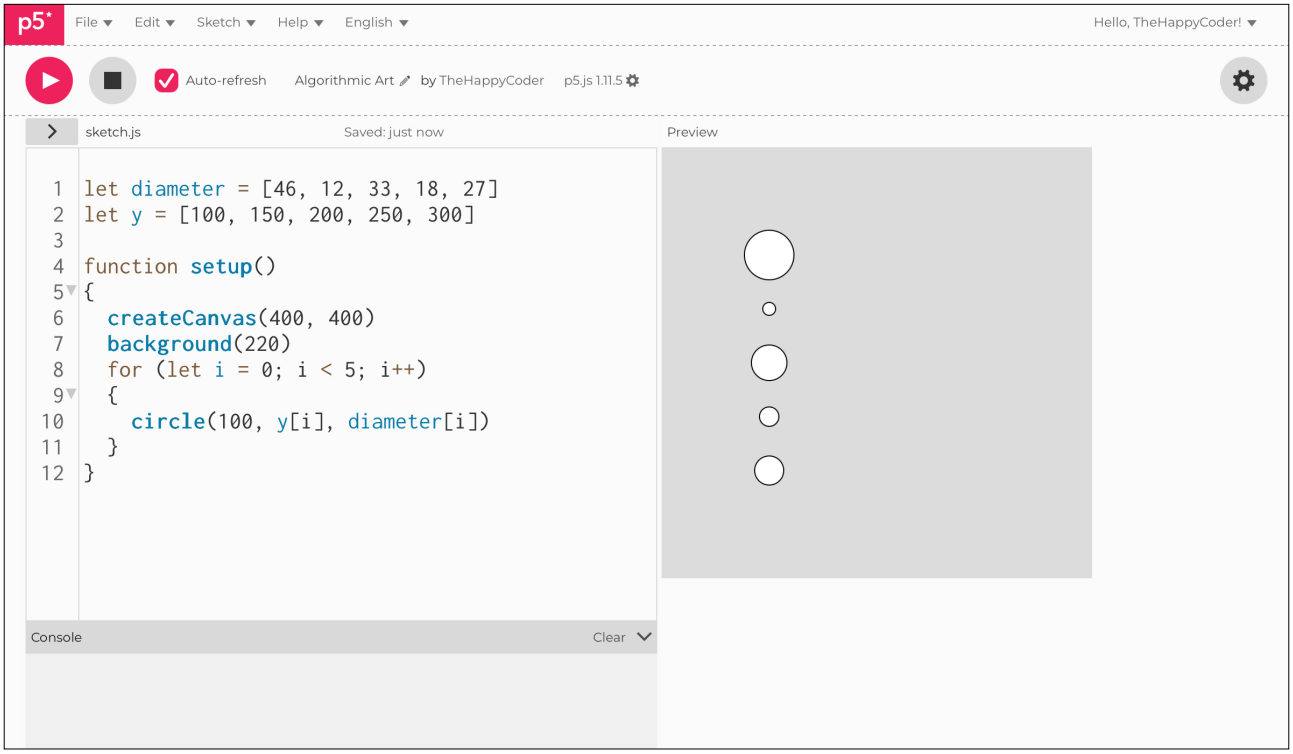
🌻 Challenge

Think of a better name for the array than y.

🛠️ Code Explanation

`circle(100, y[i], diameter[i])`	We loop through two arrays, the diameter and the y position

Figure B7.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 let diameter = [46, 12, 33, 18, 27]
2 let y = [100, 150, 200, 250, 300]
3
4 function setup()
5 {
6 createCanvas(400, 400)
7 background(220)
8 for (let i = 0; i < 5; i++)
9 {
10 circle(100, y[i], diameter[i])
11 }
12 }
Console
Clear

Sketch B7.6 an array of strings

Rather than numbers, we can also have strings of letters, words, or a combination of all three. Here, we will colour each circle with a corresponding colour from an array of named colours.

```
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 5; i++)
  {
    fill(colours[i])
    circle(100, y[i], diameter[i])
  }
}
```

🗒️ Notes

A string is denoted by having speech (singular or double) marks.

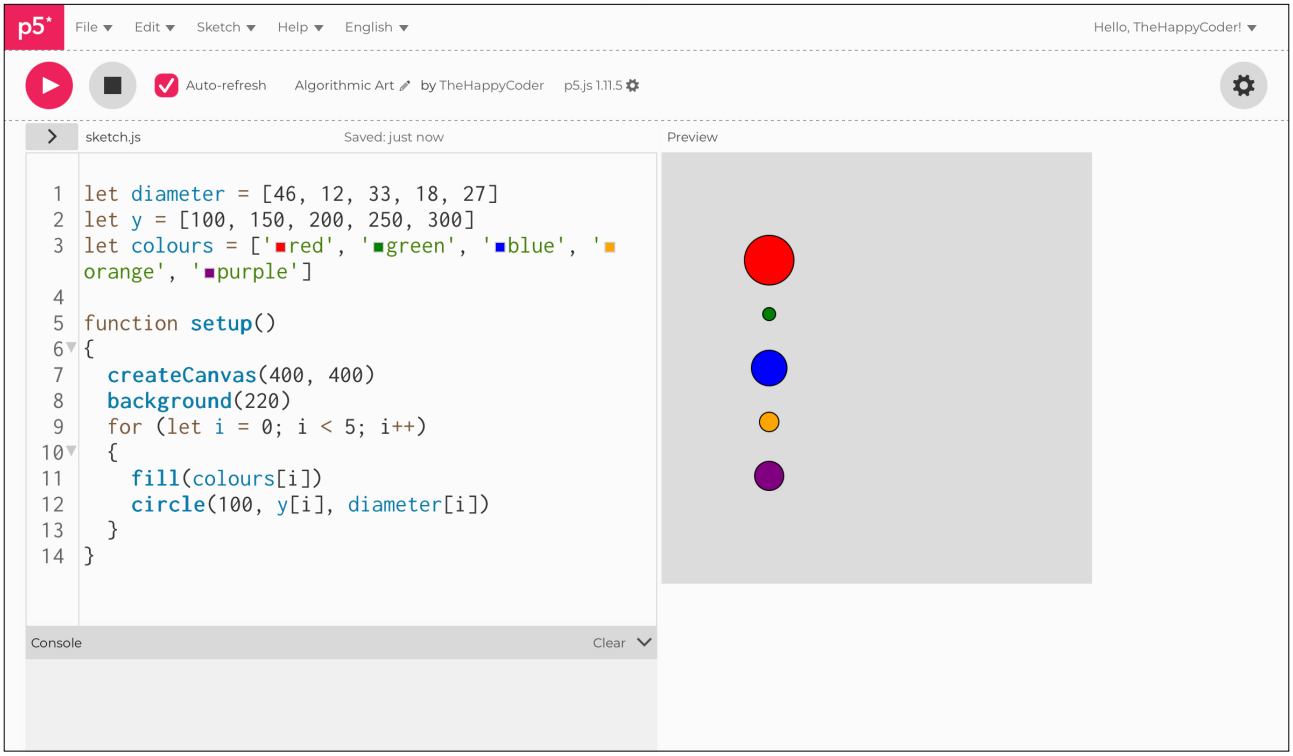
🌻 Challenge

Add your own colours.

🛠️ Code Explanation

`fill(colours[i])`	Each element is the named colour

Figure B7.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']
function setup()
{
createCanvas(400, 400)
background(220)
for (let i = 0; i < 5; i++)
{
fill(colours[i])
circle(100, y[i], diameter[i])
}
}
Console
Clear

Sketch B7.7 random selection

Another useful trick is random selection from an array; here, we will randomly select the colour of the circle.

```
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 5; i++)
  {
    fill(random(colours))
    circle(100, y[i], diameter[i])
  }
}
```

🗒️ Notes

You may get the same colour chosen more than once.

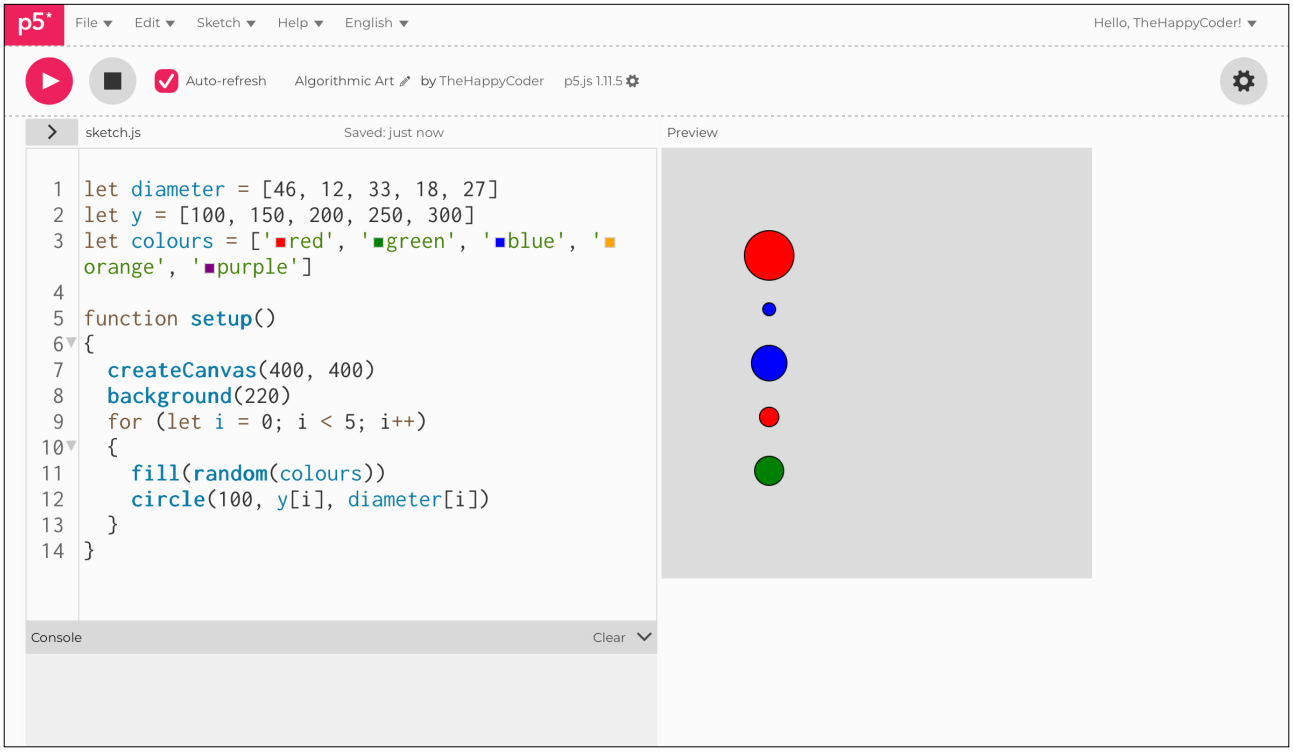
🌻 Challenges

1. Add more colours to the colour array.
2. Randomise the `y` position and the `diameter`.

🛠️ Code Explanation

`fill(random(colours))`	The colours are randomly chosen for the array

Figure B7.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']
function setup()
{
createCanvas(400, 400)
background(220)
for (let i = 0; i < 5; i++)
{
fill(random(colours))
circle(100, y[i], diameter[i])
}
}
Console
Clear

Sketch B7.8 array length

Rather than hardcoding the length of the array, we can use a function that already knows the length of the array.

```
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < diameter.length; i++)
  {
    fill(random(colours))
    circle(100, y[i], diameter[i])
  }
}
```

🗒️ Notes

Use `diameter.length` rather than hard-coding the value. This is useful if you have arrays that change in size because it is possible to run code that adds to the array (and removes elements).

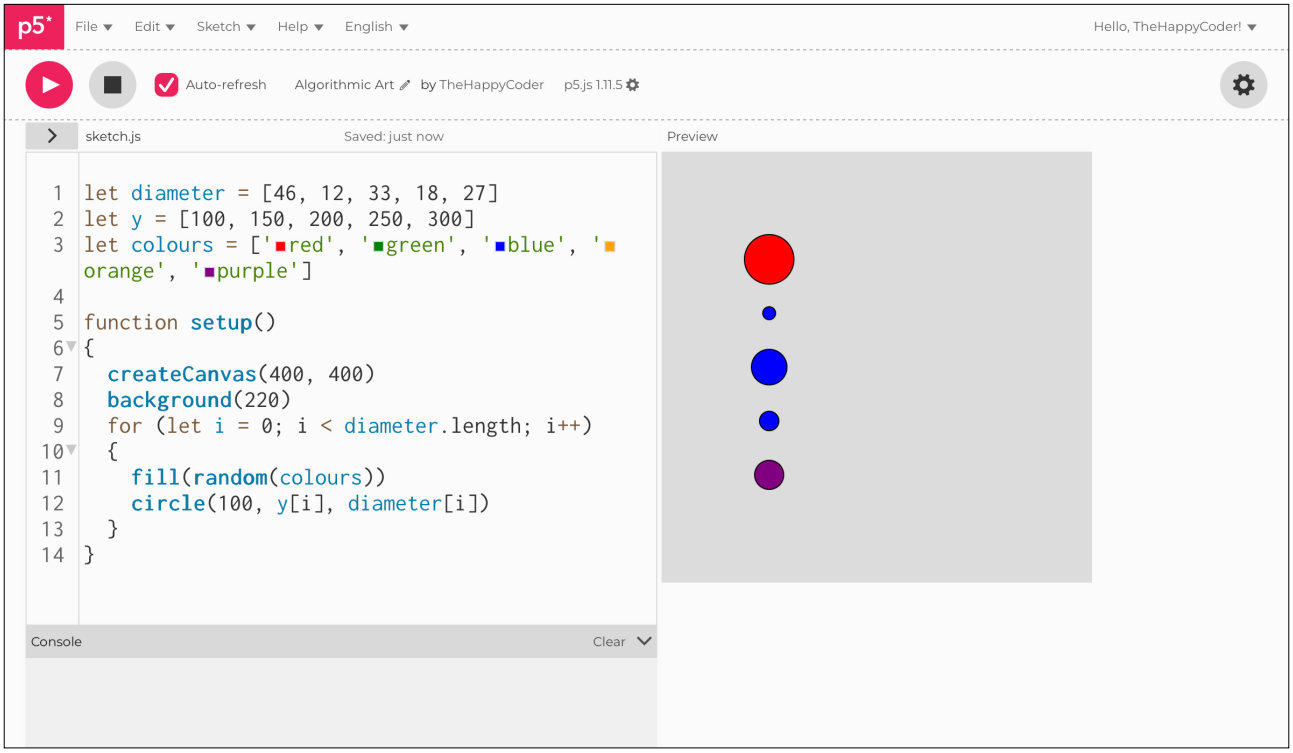
🌻 Challenge

What would happen if there were `six` elements in the `diameter` array but still only `five` in the `y` array?

🛠️ Code Explanation

`for (let i = 0; i < diameter.length; i++)`	The diameter.length looks at the array and works out how many elements it has

Figure B7.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let diameter = [46, 12, 33, 18, 27]
let y = [100, 150, 200, 250, 300]
let colours = ['red', 'green', 'blue', 'orange', 'purple']
function setup()
{
createCanvas(400, 400)
background(220)
for (let i = 0; i < diameter.length; i++)
{
fill(random(colours))
circle(100, y[i], diameter[i])
}
}
Console
Clear

Sketch B7.9 empty array

❗ start a new sketch
We have created two empty arrays, one for x and one for y.

```
let x = []
let y = []

function setup()
{
  createCanvas(400, 400)
  background(220)
}
```

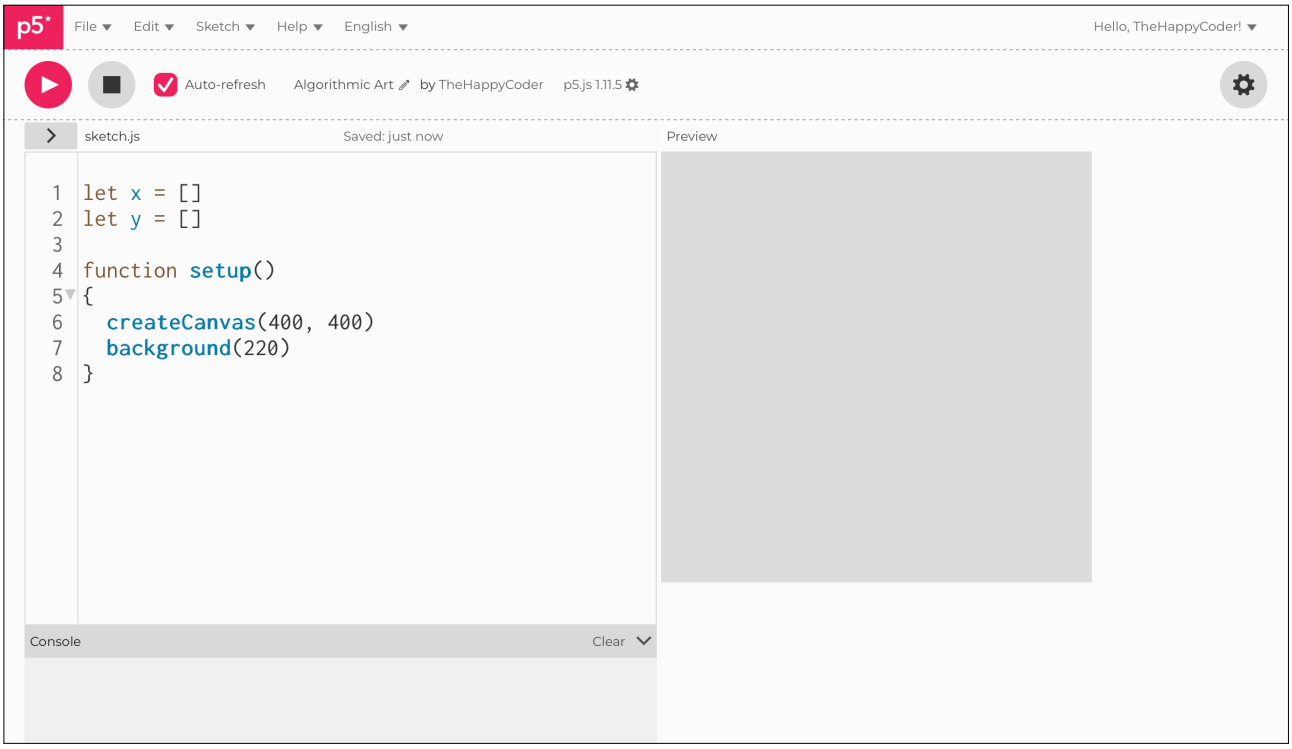
🗒️ Notes

Empty arrays created.

🛠️ Code Explanation

`let x = []`	Empty x array
`let y = []`	Empty y array

Figure B7.9



Sketch B7.10 filling the array

We can fill the array with random values; we will give each array ten numbers.

```
let x = []
let y = []

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 10; i++)
  {
    x[i] = random(width)
    y[i] = random(height)
  }
}
```

🗒️ Notes

This fills each array with ten random numbers.

🛠️ Code Explanation

`x[i] = random(width)`	Ten random numbers between 0-400 stored in the x array at each index i
`y[i] = random(height)`	Ten random numbers between 0-400 stored in the y array at each index i

Sketch B7.11 drawing the circles

We can now draw these circles. Every time you run the sketch, it generates a new set of ten circles.

```
let x = []
let y = []

function setup()
{
  createCanvas(400, 400)
  background(220)
  for (let i = 0; i < 10; i++)
  {
    x[i] = random(width)
    y[i] = random(height)
  }
  for (let i = 0; i < x.length; i++)
  {
    circle(x[i], y[i], 20)
  }
}
```

🗒️ Notes

This uses the length of the x array, x.length, so it is critical that the y array is either the same length or longer. If you have two arrays that you need to check, then you would need to build in some code to check and compare lengths and always use the smaller of the two.

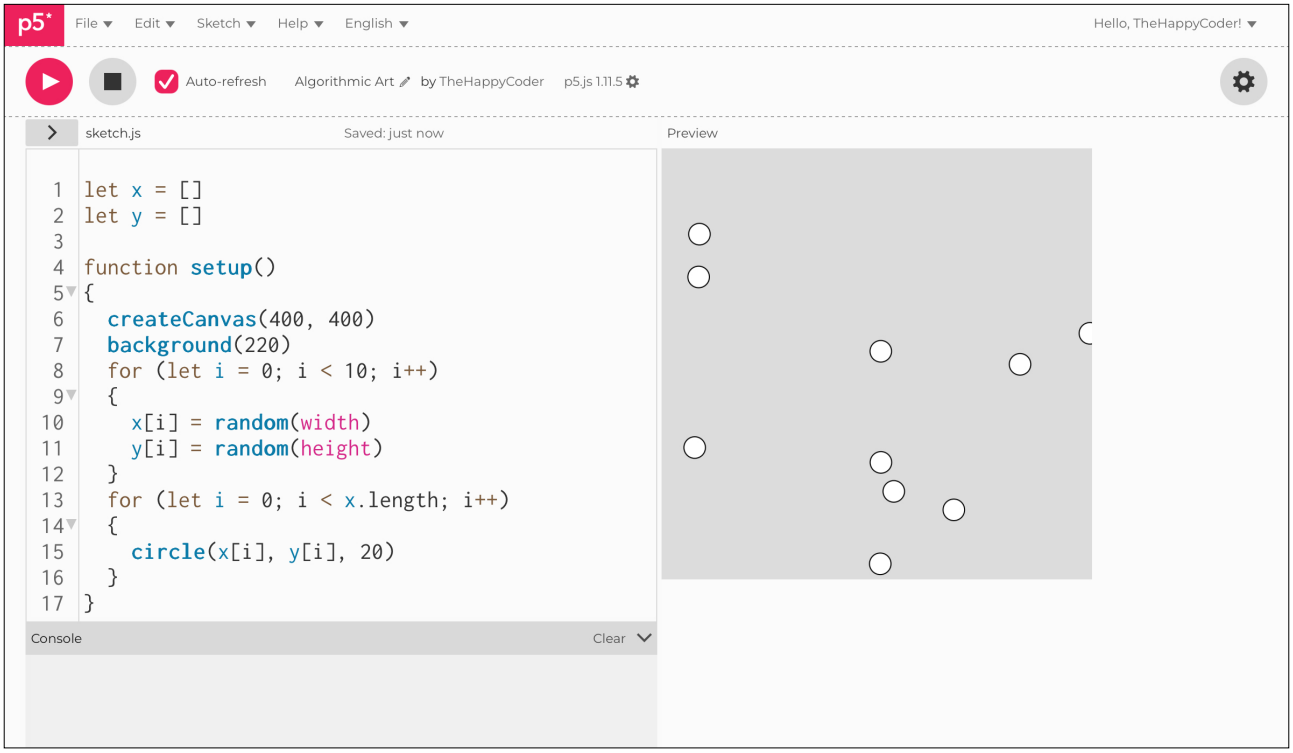
🌻 Challenge

Could you devise a way of checking the length of each array and comparing them?

🛠️ Code Explanation

`for (let i = 0; i < x.length; i++)`	The x.length means that it stops when it gets to the end of the x array
`circle(x[i], y[i], 20)`	Looks at each array (x and y) and pulls each value in turn to collect the coordinates of the circles

Figure B7.11


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let x = []
let y = []
function setup()
{
createCanvas(400, 400)
background(220)
for (let i = 0; i < 10; i++)
{
x[i] = random(width)
y[i] = random(height)
}
for (let i = 0; i < x.length; i++)
{
circle(x[i], y[i], 20)
}
}
Console
Clear

Sketch B7.12 clicking the mouse

❗ start a new sketch

We can use a function called `mousePressed()`. It is a function that waits for the mouse to be clicked (pressed) and will execute any code you give it. Here we simply draw a circle every time we click on the canvas.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
}
```

```
function mousePressed()
{
  let x = mouseX
  let y = mouseY
  circle(x, y, 20)
}
```

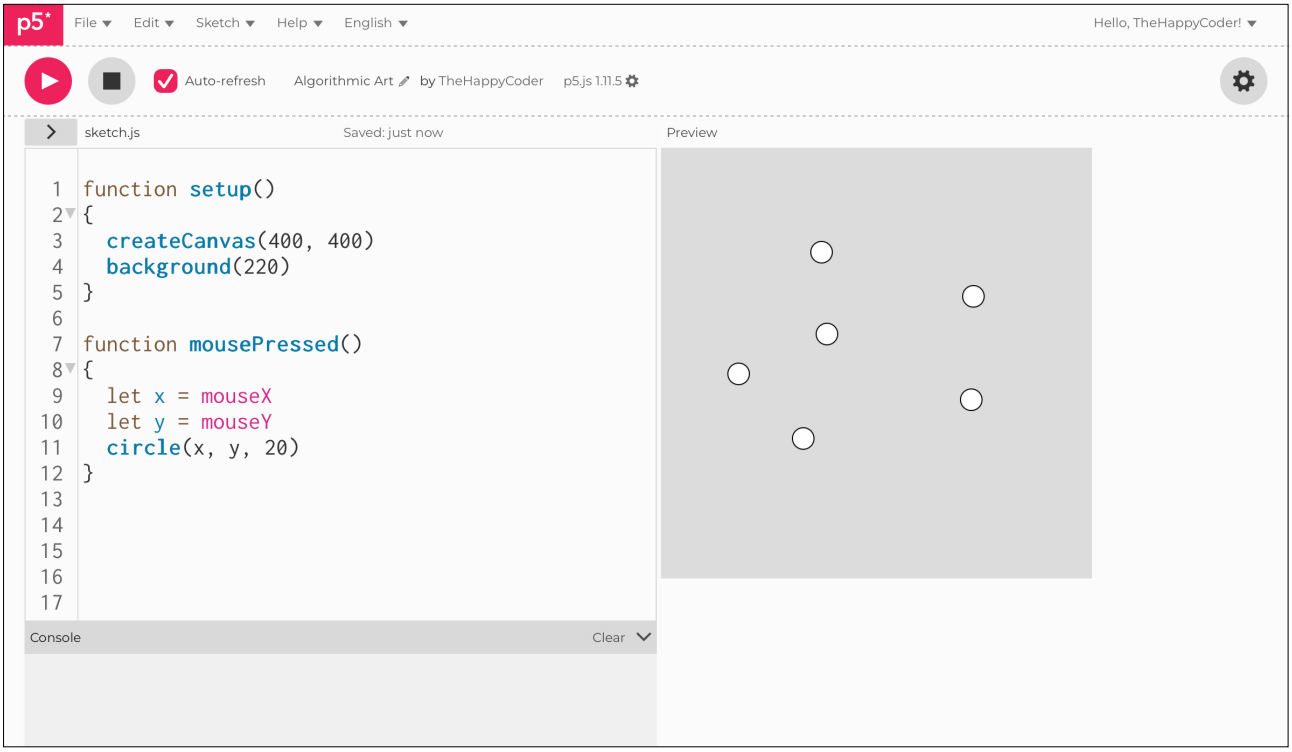
🗒️ Notes

You get a circle every time you click on the canvas.

🛠️ Code Explanation

`function mousePressed()`	This function is only executed when the mouse is pressed

Figure B7.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
background(220)
}
function mousePressed()
{
let x = mouseX
let y = mouseY
circle(x, y, 20)
}
Console
Clear

Sketch B7.13 pushing the array

Instead, we can store the positions of the circles (their x and y coordinates) when we click on the canvas. The function now pushes the x and y coordinates into an array called bubbles using the push() function.

```
let bubbles = []

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function mousePressed()
{
  let x = mouseX
  let y = mouseY
  circle(x, y, 20)
  bubbles.push(x, y)
}
```

🗒️ Notes

The push() function does just that, pushing elements into an array.

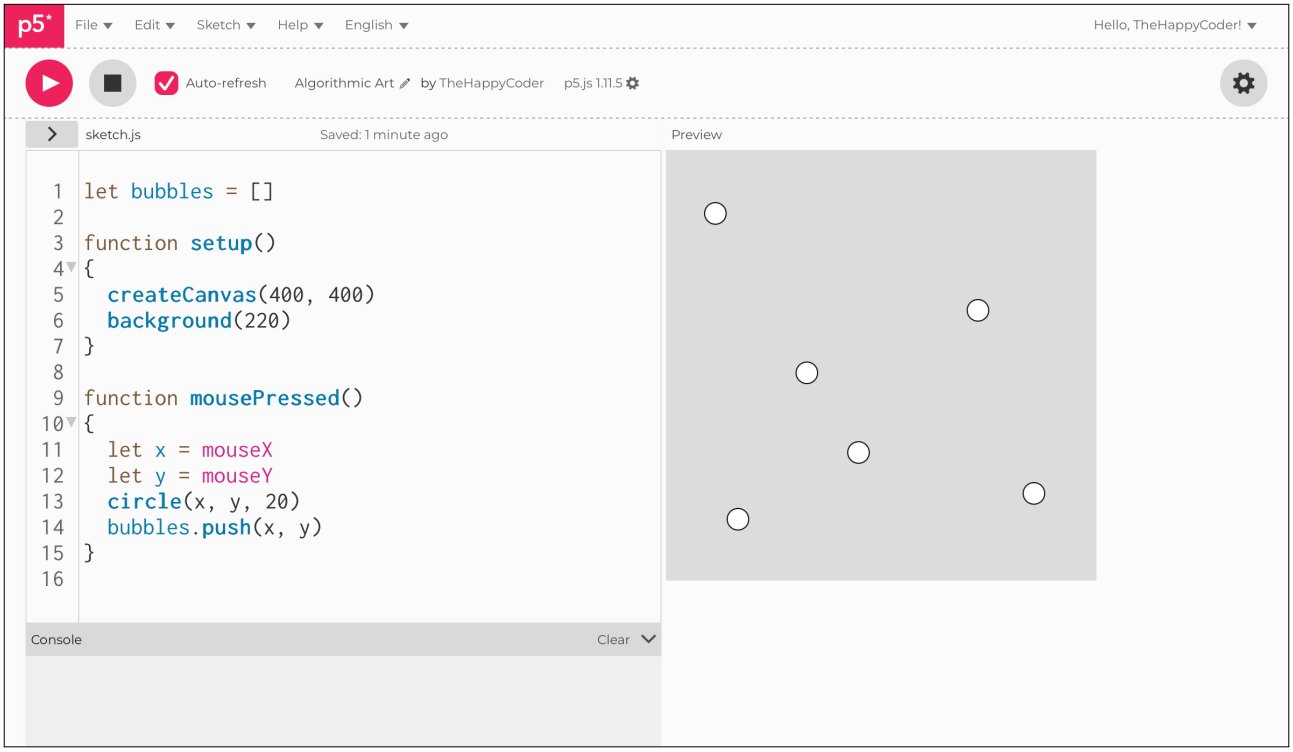
🌻 Challenge

Could we push a third element, for instance, the diameter?

🛠️ Code Explanation

bubbles.push(x, y)	This pushes two elements into a single array called bubbles

Figure B7.13


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 1 minute ago
Preview
let bubbles = []
function setup()
{
createCanvas(400, 400)
background(220)
}
function mousePressed()
{
let x = mouseX
let y = mouseY
circle(x, y, 20)
bubbles.push(x, y)
}
Console
Clear

Sketch B7.14 console log

We can see inside the array by using something called `console.log()`. This sends information to the console.

```
let bubbles = []

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function mousePressed()
{
  let x = mouseX
  let y = mouseY
  circle(x, y, 20)
  bubbles.push(x, y)
  console.log(bubbles)
}
```

🗒️ Notes

You can see the results in the console below the code panel. When you click once, you get two elements in the array: the first circle's (`bubble`) `x` and `y` coordinates, then when you click a second time, you get another pair of coordinates for the second circle, and so on.

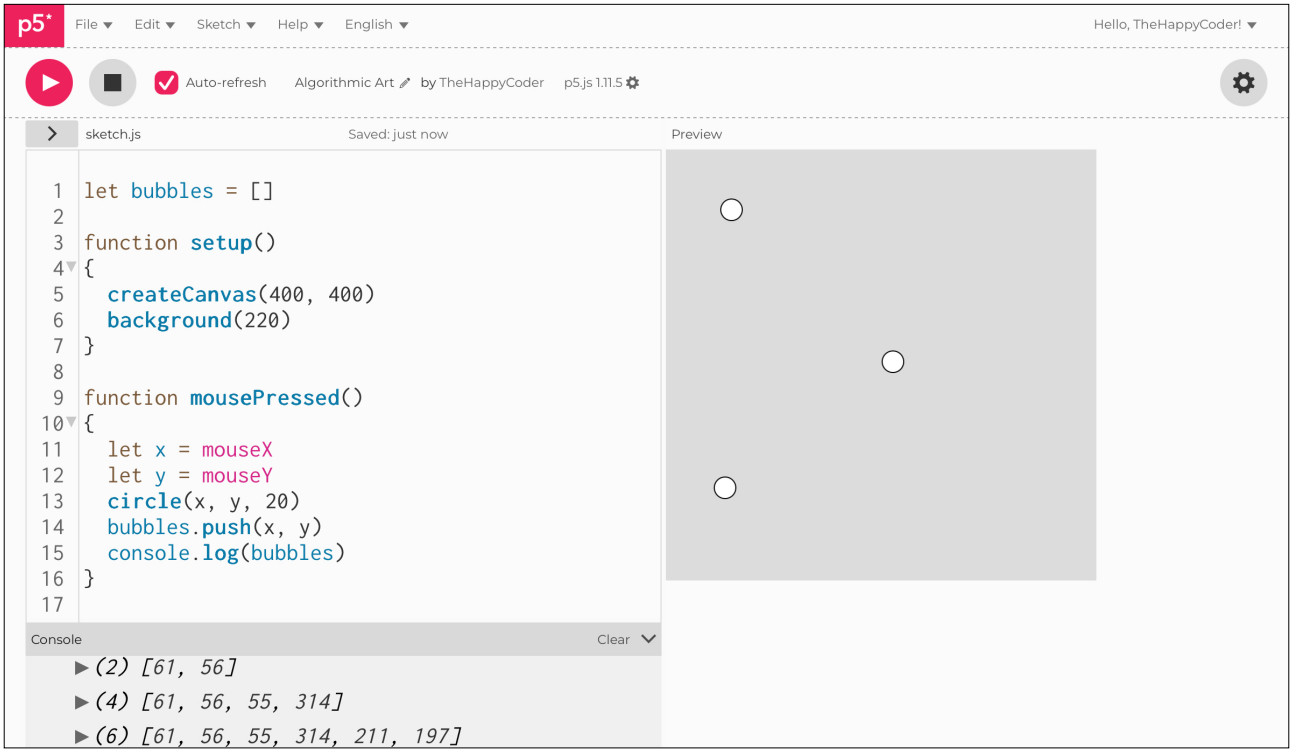
🌻 Challenge

Try `console.log(bubbles.length)`.

🛠️ Code Explanation

`let bubbles = []`	We define an empty array
`let x = mouseX`	Our x value is the mouseX position when we click
`let y = mouseY`	Our y value is the mouseY position when we click
`bubbles.push(x, y)`	The x and y values are pushed into the bubbles array
`console.log(bubbles)`	We can see inside the bubbles array

Figure B7.14


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let bubbles = []
function setup()
{
createCanvas(400, 400)
background(220)
}
function mousePressed()
{
let x = mouseX
let y = mouseY
circle(x, y, 20)
bubbles.push(x, y)
console.log(bubbles)
}
Console
Clear
(2) [61, 56]
(4) [61, 56, 55, 314]
(6) [61, 56, 55, 314, 211, 197]

Adding and splicing

Arrays can often be pre-populated and are therefore quite static. But you can also add elements to them with new strings or integers, or even change an element. Also, you are able to remove an element or add a whole new database to the array. This makes it a powerful tool if you want to use data that is dynamic.

You can also start with an empty array and fill it with newly created data. Useful when creating an array or database with new data.

Sketch B7.15 appending using concat

❗ new sketch

The `concat()` function allows you to join two arrays together to make one. We can see inside the array to check if they have been combined.

```
let colours1 = ['red', 'green', 'blue']
let colours2 = ['yellow', 'orange', 'purple']
let colours3 = []

function setup()
{
  createCanvas(400, 400)
  background(220)
  colours3 = colours1.concat(colours2)
  console.log(colours3)
}
```

🗒️ Notes

This works with numbers and with objects.

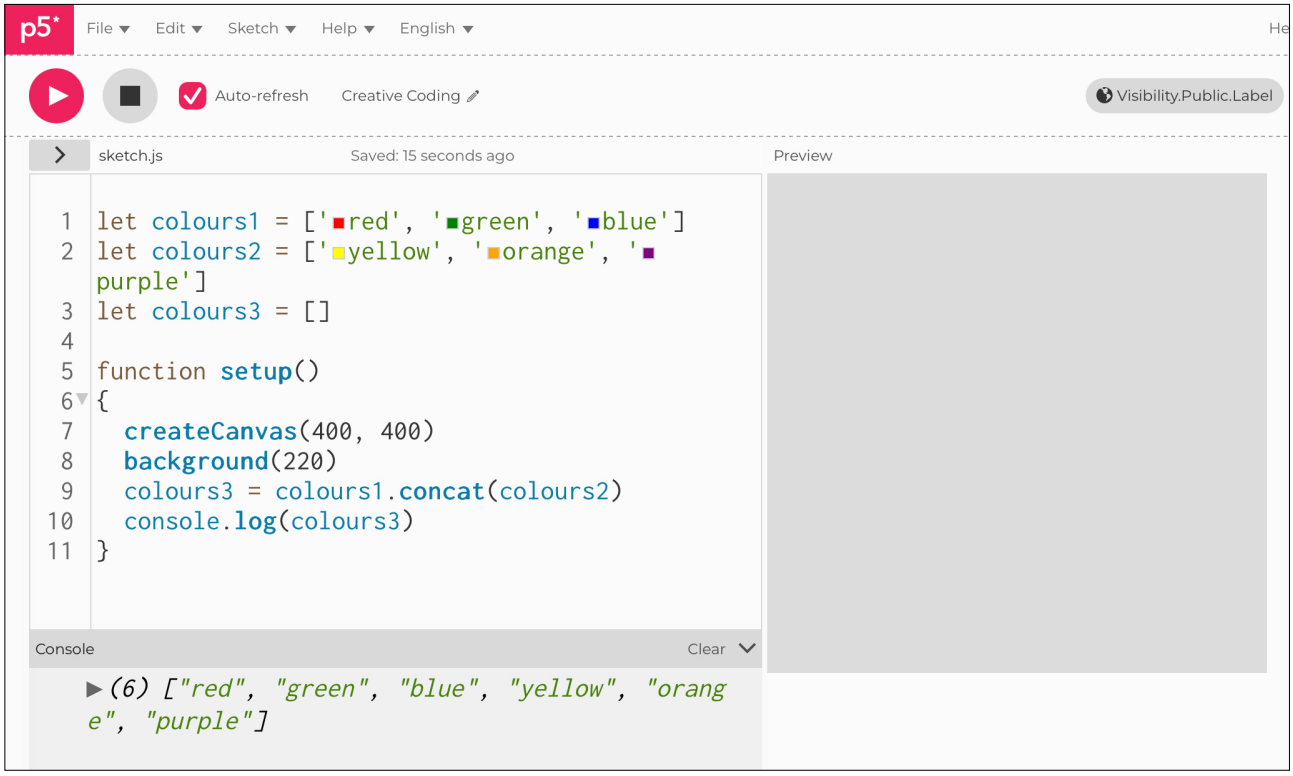
🌻 Challenge

Try with three arrays. Does it work?

🛠️ Code Explanation

`colours3 = colours1.concat(colours2)`	Adds two (colours1 and colours2) arrays together to form one new one (colours3)

Figure B7.15


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Creative Coding
Visibility.Public.Label
sketch.js
Saved: 15 seconds ago
Preview
let colours1 = ['red', 'green', 'blue']
let colours2 = ['yellow', 'orange', 'purple']
let colours3 = []
function setup()
{
createCanvas(400, 400)
background(220)
colours3 = colours1.concat(colours2)
console.log(colours3)
}
Console
Clear
(6) ["red", "green", "blue", "yellow", "orange", "purple"]

Sketch B7.16 adding by splicing

❗ newish sketch

You can add new elements to the array using `splice()`. Here we add an extra colour (yellow) at the end. There are three arguments to the `splice()` function. The first argument (`3`) tells you where you want to add the extra element, the second argument (`0`) means you are adding it, and the third argument (`'yellow'`) is what you are adding.

```
let colours = ['red', 'green', 'blue']


function setup()
{
  createCanvas(400, 400)
  background(220)
  colours.splice(3, 0, 'yellow')
  console.log(colours)
}
```

🗒️ Notes

You get an extra element added to the array. If you want to add something to the end of the array, giving it a hardcoded index position is not going to work if the array keeps growing. A better way is shown in the next sketch.

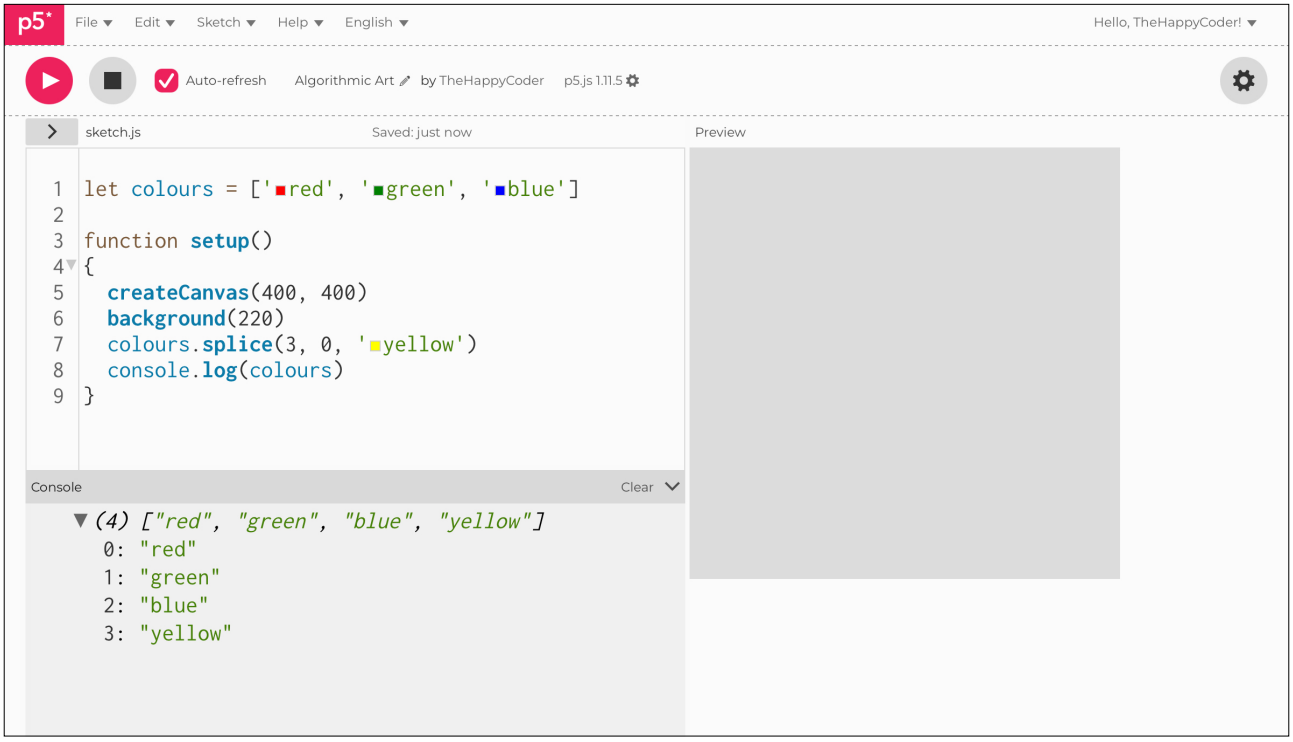
🌻 Challenges

1. Put `'yellow'` in other positions in the array.
2. What happens if you give it a position, say, `13`?

🛠️ Code Explanation

`colours.splice(3, 0, 'yellow')`	Putting yellow at index[3] which is the fourth (end) element in the array

Figure B7.16


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 let colours = ['red', 'green', 'blue']
2
3 function setup()
4 {
5 createCanvas(400, 400)
6 background(220)
7 colours.splice(3, 0, 'yellow')
8 console.log(colours)
9 }
Console
Clear
(4) ["red", "green", "blue", "yellow"]
0: "red"
1: "green"
2: "blue"
3: "yellow"

Sketch B7.17 a better way

We can use the length of the array as our way of keeping track of the number of elements in the array.

```
let colours = ['red', 'green', 'blue']


function setup()
{
  createCanvas(400, 400)
  background(220)
  colours.splice(colours.length, 0, 'yellow')
  console.log(colours)
}
```

🗒️ Notes

This is a stronger way of adding to an array.

🛠️ Code Explanation

`colours.splice(colours.length, 0, 'yellow')`	Adds to the end of the array whatever the length

Sketch B7.18 in a different place

This time we place the yellow in spot `index [1]`, just in case you haven't already tried.

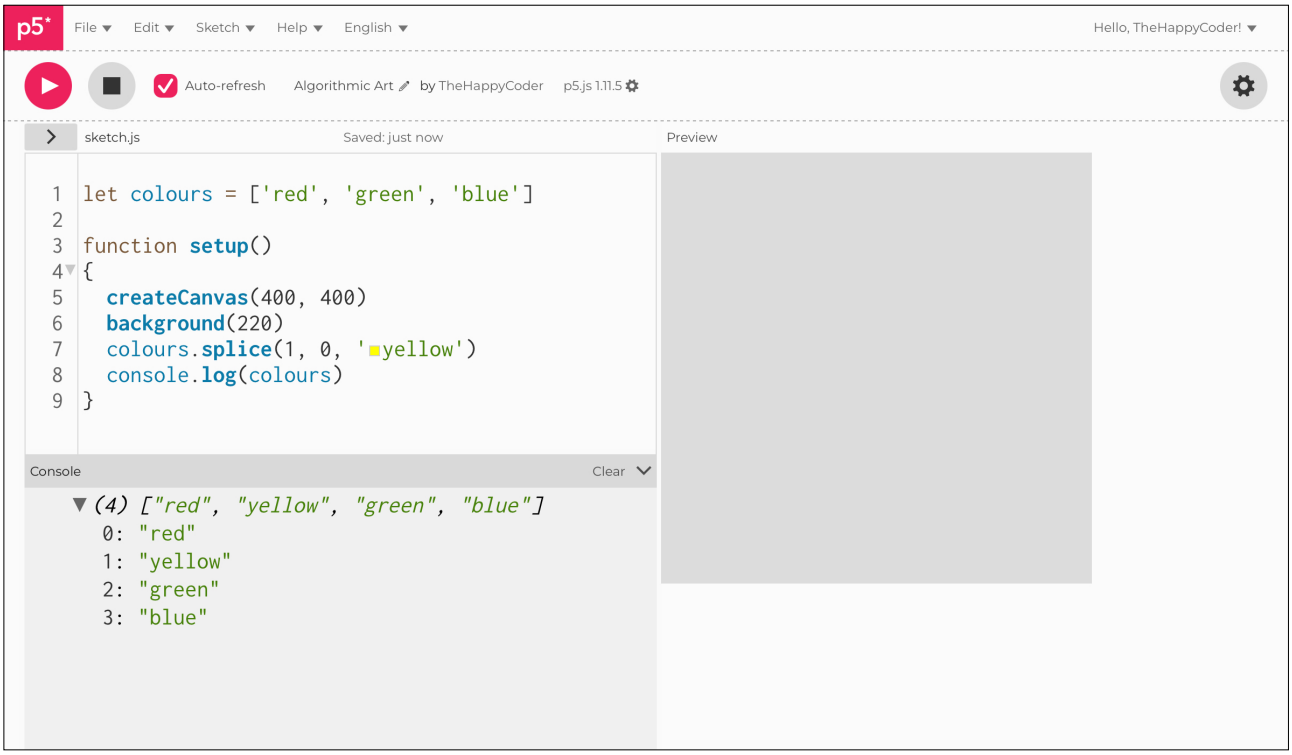
```
let colours = ['red', 'green', 'blue']


function setup()
{
  createCanvas(400, 400)
  background(220)
  colours.splice(1, 0, 'yellow')
  console.log(colours)
}
```

Notes

It is now sandwiched between the red and the green.

Figure B7.18



Sketch B7.19 replacing an element using splice

As well as adding, we can replace an element in an array. We do this by replacing the 0 with a 1 as the second argument.

```
let colours = ['red', 'green', 'blue']


function setup()
{
  createCanvas(400, 400)
  background(220)
  colours.splice(1, 1, 'yellow')
  console.log(colours)
}
```

🗒️ Notes

The first argument tells you where to put it. It replaces the green with yellow.

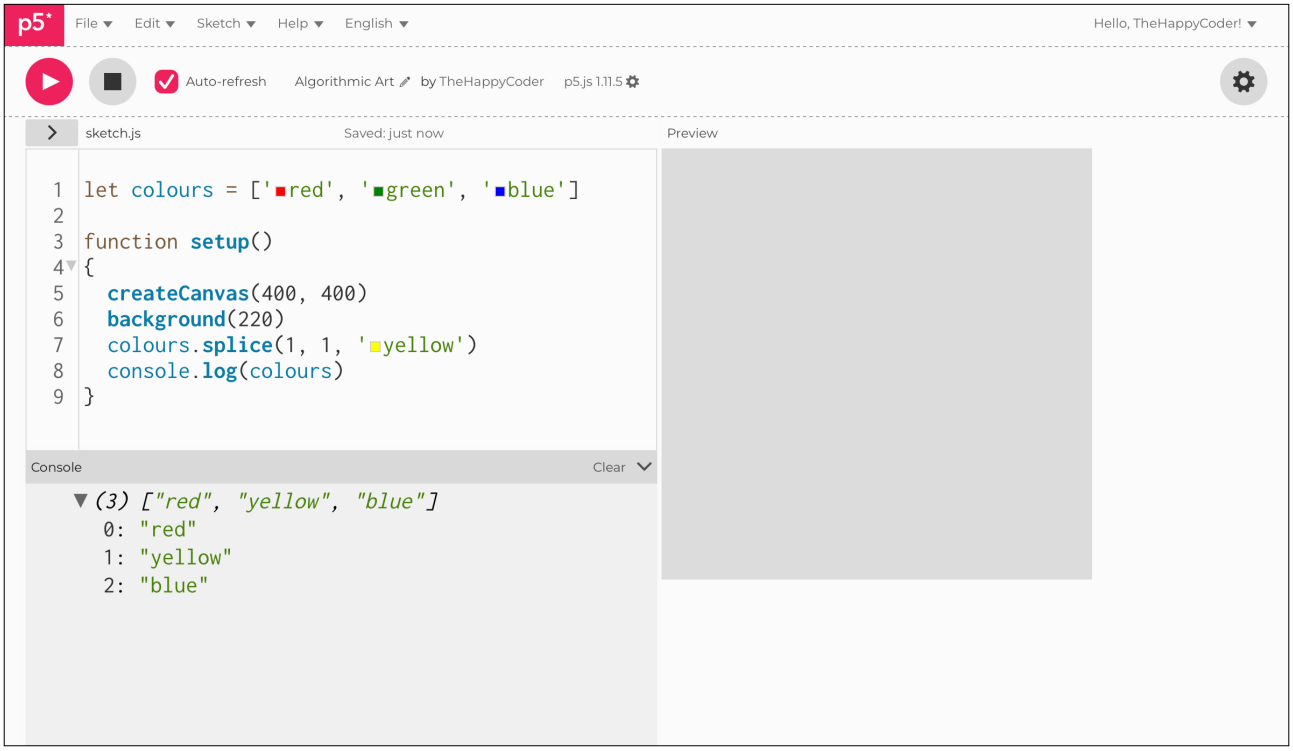
🌻 Challenge

Replace other colours.

🛠️ Code Explanation

`colours.splice(1, 1, 'yellow')`	Replaces the 'green' with 'yellow'

Figure B7.19


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 let colours = ['red', 'green', 'blue']
2
3 function setup()
4 {
5 createCanvas(400, 400)
6 background(220)
7 colours.splice(1, 1, 'yellow')
8 console.log(colours)
9 }
Console
Clear
(3) ["red", "yellow", "blue"]
0: "red"
1: "yellow"
2: "blue"

Sketch B7.20 deleting an element using splice

You can delete an element from an array; here we delete two elements starting at index 1.

```
let colours = ['red', 'green', 'blue']


function setup()
{
  createCanvas(400, 400)
  background(220)
  colours.splice(1, 2)
  console.log(colours)
}
```

🗒️ Notes

We have removed both the green and blue colour elements from the array.

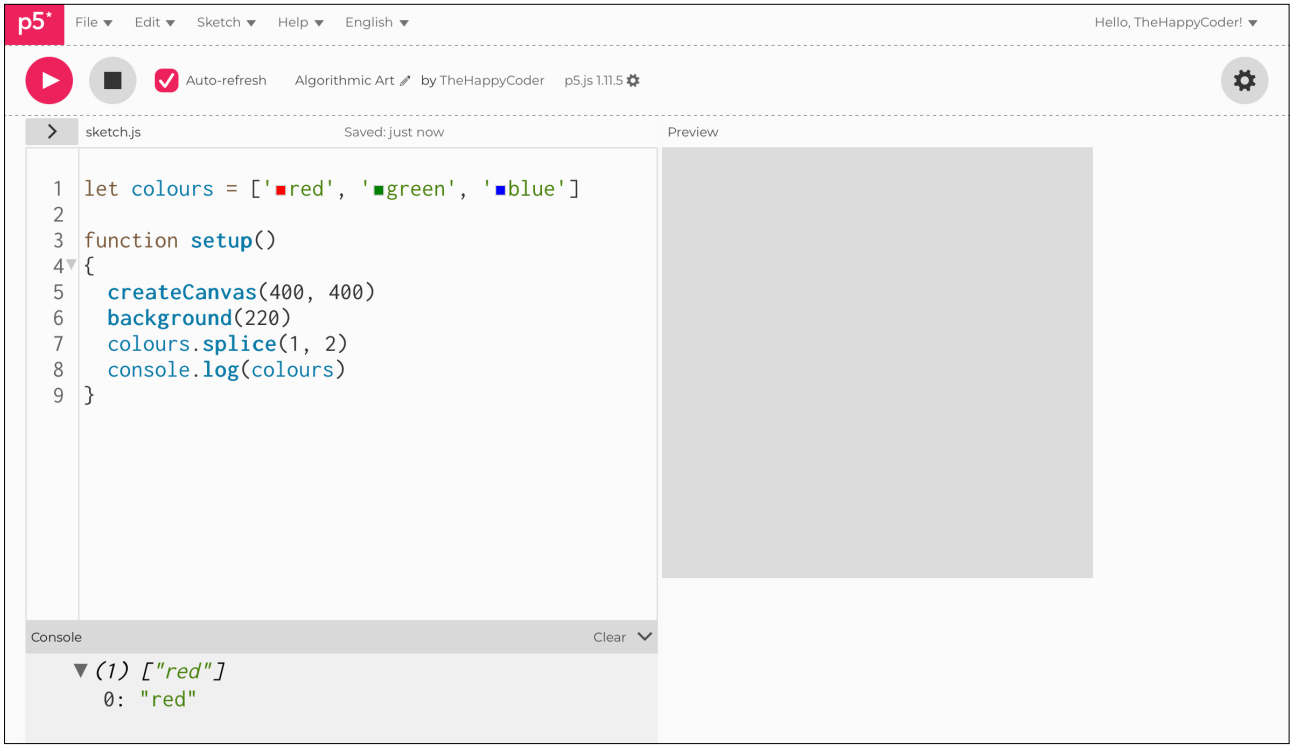
🌻 Challenge

Try: `colours.splice(0, 3).` You should get an empty array.

🛠️ Code Explanation

`colours.splice(1, 2)`	Deleting two elements starting at index[1]

Figure B7.20


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let colours = ['■red', '■green', '■blue']
function setup()
{
createCanvas(400, 400)
background(220)
colours.splice(1, 2)
console.log(colours)
}
Console
Clear
(1) ["red"]
0: "red"

Sketch B7.21 an array of objects

❗ start a new sketch and call the array `data` for a change (mousePressed not draw!)
Another way to use an array is to have objects. In this example, an object is one circle. Each object has an `x` and `y` value; they are recognisable as objects because of the colons (`:`). We will use `console.log(data)` as before to see the difference. Each object is called `inputs` to collect the `x` and `y` co-ordinates when you click on the canvas.

```
let data = []

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function mousePressed()
{
  let inputs = {
    x: mouseX,
    y: mouseY
  }
  circle(inputs.x, inputs.y, 30)
  data.push(inputs)
  console.log(data)
}
```

Notes

We use `console.log(data)` to see inside the array. It gives you a list of objects, each one representing a circle. Inside each object are the `x` and `y` inputs. Click on the arrow to reveal the contents of the array (fig. 1.21b). Please note that in version 2 you will get a float not an integer.

Challenge

Change the variable names of `x` and `y` to, say, `flower` and `animal`. See where else you have to change them.

Code Explanation

`let inputs = {...}`	We give the collective name for the x and y values as inputs
`x: mouseX`	For the x value we use a colon then the input value or variable
`y: mouseY`	For the y value we use a colon then the input value or variable
`circle(inputs.x, inputs.y, 50)`	The x co-ordinate of the object (inputs) is inputs.x, repeated for the y component

Figure B7.21a

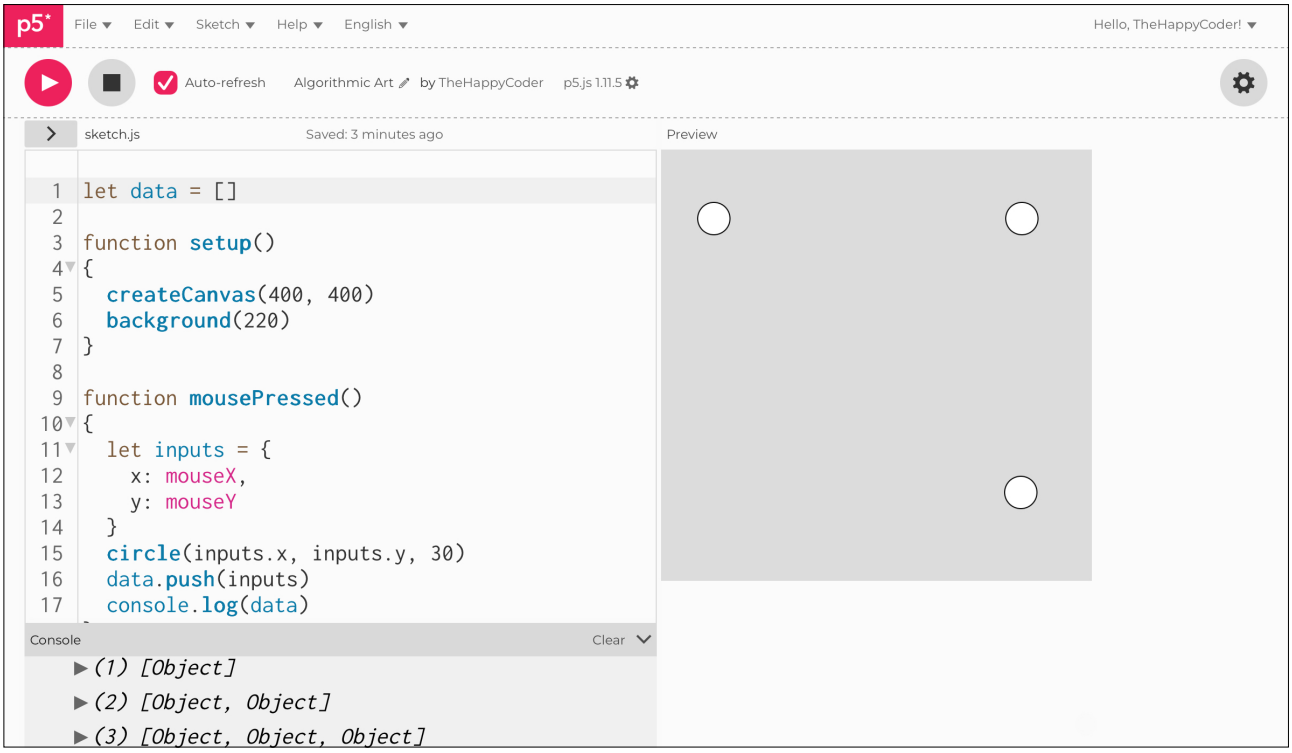


Figure B7.21b: opening the object in the console