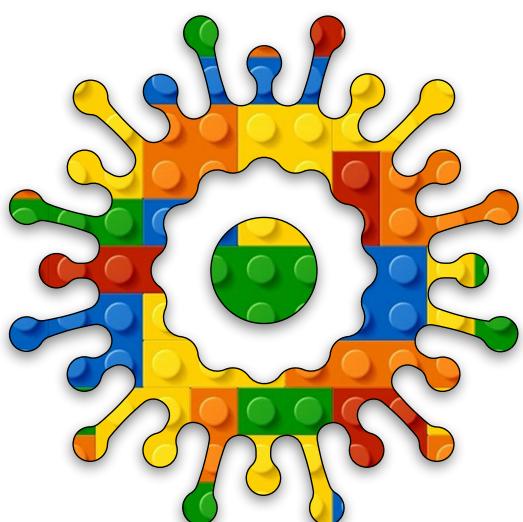


Creative
Coding
Module D
Unit #4

push and
splice





Module D Unit #4 push and splice

- Sketch D4.1 recap
- Sketch D4.2 push()
- Sketch D4.3 splice() up your life
- Sketch D4.4 splicing a bubble
- Sketch D4.5 oversized array
- Sketch D4.6 just roll over
- Sketch D4.7 just a short dist()
- Sketch D4.8 inside the bubble
- Sketch D4.9 true colour
- Sketch D4.10 array of bubbles check
- Sketch D4.11 click of a mouse
- Sketch D4.12 mousePressed()
- Sketch D4.13 mouse delete



Introduction to push and splice

Having a bit of a deeper dive into arrays and then some fun stuff. The first part, we use the `console.log()` to see inside the array, and then we are back on the **bubbles**.

We will also introduce the adding of more functions to a class, in this case, the **Bubble** class.



Sketch D4.1 recap

! A new sketch.

A simple way to add an element to an array is as follows: we are saying that element [5] is 21.

```
let num = [42, 56, 63, 79, 88]

function setup()
{
    num[5] = 21
    console.log(num)
}
```

Notes

You will need to make sure you know how long your array is; this is challenging when you might be adding or removing elements all the time. There is a better way.

Figure D4.1

The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play/pause, stop, refresh, and settings, along with the text "Hello, TheHappyCoder! ▾". The main area has tabs for "sketch.js" and "Preview". The code editor contains the following JavaScript code:

```
1 let num = [42, 56, 63, 79, 88]
2
3 function setup()
4 {
5   num[5] = 21
6   console.log(num)
7 }
```

Below the code editor is a "Console" panel with a "Clear" dropdown menu. It displays the output of the `console.log` statement:

```
▶ (6) [42, 56, 63, 79, 88, 21]
```



Sketch D4.2 push()

Using the **push()** function, it simply adds another element onto the end of the array, as we saw in the previous sketch.

```
let num = [42, 56, 63, 79, 88]

function setup()
{
    num.push(21)
    console.log(num)
}
```

Figure D4.2

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and user information ('Hello, TheHappyCoder!'). Below the toolbar, the title bar says 'sketch.js' and 'Saved: just now'. To the right of the title bar is a 'Preview' button. The main area contains the following code:

```
1 let num = [42, 56, 63, 79, 88]
2
3 function setup()
4{
5  num.push(21)
6  console.log(num)
7}
```

At the bottom left, there's a 'Console' tab and a 'Clear' button. The console output shows the result of the log statement:

```
▶ (6) [42, 56, 63, 79, 88, 21]
```



Sketch D4.3 splice() up your life

But what if we want to delete an element from an array? We use a function called `splice()`. It has two arguments: the first is the index, and the second tells you how many to delete (starting at the index). We want to delete the first one in the array, so it is `splice(0, 1)`.

```
let num = [42, 56, 63, 79, 88]

function setup()
{
    num.splice(0, 1)
    console.log(num)
}
```

Figure D4.3

The screenshot shows the p5.js code editor interface. At the top, there are navigation menus: File, Edit, Sketch, Help, and English. On the right side, it says "Hello, TheHappyCoder!" with a gear icon. The main workspace is titled "sketch.js" and shows the following code:

```
1 let num = [42, 56, 63, 79, 88]
2
3 function setup()
4{
5  num.splice(0, 1)
6  console.log(num)
7}
```

Below the code editor is a "Console" panel with a "Clear" button. It displays the output of the `console.log` statement:

```
▼ (4) [56, 63, 79, 88]
  0: 56
  1: 63
  2: 79
  3: 88
```



Sketch D4.4 splicing a bubble

! new bubble sketch, I have highlighted the major changes.
To delete a bubble, we create a function called **keyPressed()** which will activate when you press a key on your keyboard. Also, I have reduced the random wobble so that you can see how they are deleted every time you hit a key.

! remember to click on the canvas first.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function keyPressed()
{
    bubbles.splice(0, 1)
    console.log(bubbles.length)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
```

```

        bubbles[i].show()
        bubbles[i].move()
    }

}

class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        noFill()
        circle(this.x, this.y, this.r)
    }
}

```

Notes

I have included a `console.log()` so that you can see how many are in the array and see the number go down. Every time you press a key, you should see a bubble disappear, the oldest or first in the array.

Figure D4.4

The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and other options like 'Algorithmic Art' by TheHappyCoder. The version p5.js 1.11.7 is also visible. On the right, a 'Preview' window displays a cluster of small, semi-transparent circles of varying sizes, centered around a larger, darker circle. The left side of the screen shows the code in a text editor:

```
> sketch.js          Saved: 25 seconds ago
35   this.x = x
36   this.y = y
37   this.r = r
38 }
39
40 move()
41 {
42   this.x = this.x + random(-1, 1)
43   this.y = this.y + random(-1, 1)
44 }
45
46 show()
47 {
48   noFill()
49   circle(this.x, this.y, this.r)
50 }
51 }
```

Below the code editor is a 'Console' window which lists the line numbers 48, 47, 46, 45, and 44.



Sketch D4.5 oversized array

Something else we can do is delete them if the array is over a certain size. For this, we remove the `keyPressed()` function completely. Here, the array is never bigger than **50** bubbles.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        bubbles[i].show()
        bubbles[i].move()
    }
    if (bubbles.length > 50)
    {
        bubbles.splice(0, 1)
    }
}
```

```
class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        noFill()
        circle(this.x, this.y, this.r)
    }
}
```

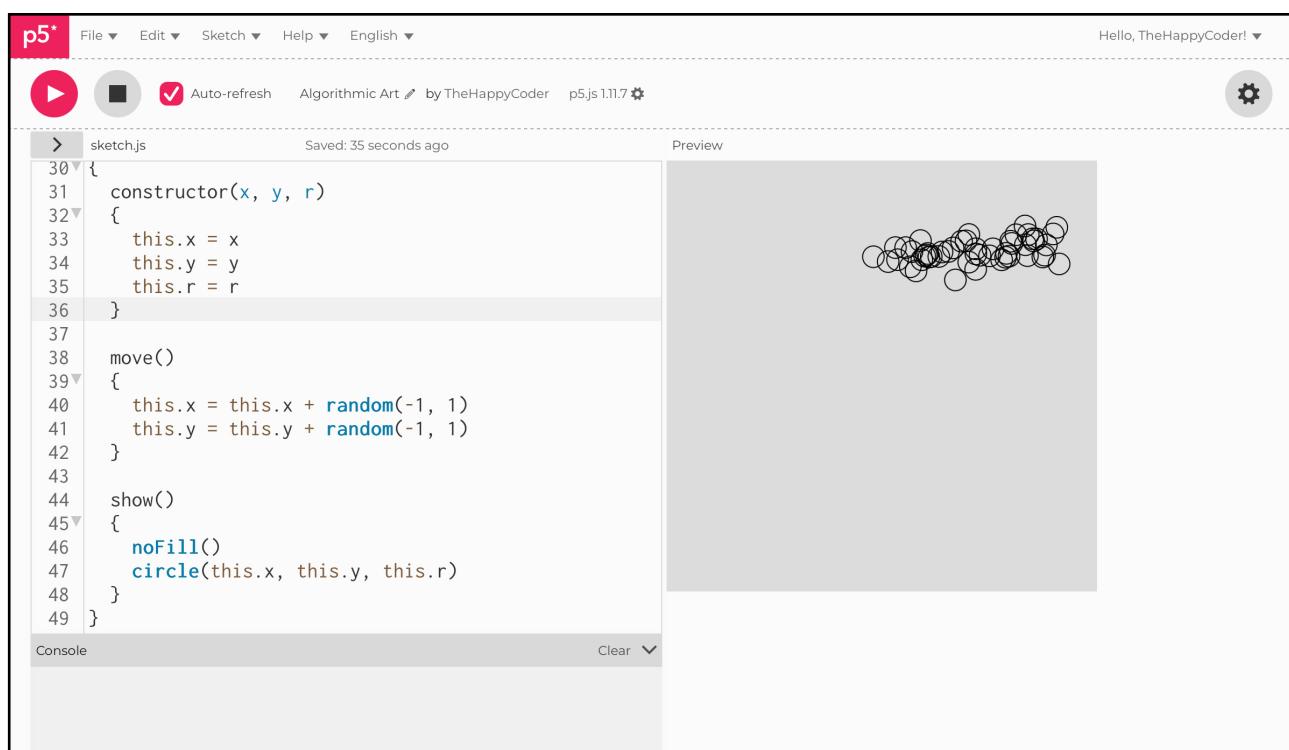
Notes

What you get is what looks like a trail following you, which never gets too long.

Challenge

Have them delete themselves if they wander off the canvas.

Figure D4.5



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

```
> sketch.js          Saved: 35 seconds ago
30> {
31   constructor(x, y, r)
32   {
33     this.x = x
34     this.y = y
35     this.r = r
36   }
37
38   move()
39   {
40     this.x = this.x + random(-1, 1)
41     this.y = this.y + random(-1, 1)
42   }
43
44   show()
45   {
46     noFill()
47     circle(this.x, this.y, this.r)
48   }
49 }
```

The preview window shows a cluster of black-outlined circles of varying sizes scattered across a light gray background, representing the output of the p5.js code.



Sketch D4.6 just roll over

Changing the colour of the bubble when the mouse rolls over the bubble. First, we will create a function called `changeColour()` in the bubble class. It will have an argument called `bright`, which is set to `0` by default.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
  bubble = new Bubble(mouseX, mouseY, 20)
  bubbles.push(bubble)
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
  if (bubbles.length > 50)
  {
    bubbles.splice(0, 1)
  }
}
```

```
}

class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        noFill()
        circle(this.x, this.y, this.r)
    }

    changeColour(bright)
    {
        this.brightness = bright
    }
}
```

***** Notes

Nothing is yet changing, just building the sketch.



Sketch D4.7 just a short dist()

Next, we want a function that measures when the mouse is over the bubble. We create a function in the bubble class called `rollover()`. It receives two arguments, `px` and `py`, which are the `mouse X` and `mouseY` co-ordinates (which we will get to in a moment).

The function `dist()` measures the distance between two sets of co-ordinates; in this case, it is the distance between the centre of the bubble, `this.x`, `this.y`, and the `mouseX` and `mouseY` positions.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        bubbles[i].show()
        bubbles[i].move()
    }
    if (bubbles.length > 50)
```

```
{  
    bubbles.splice(0, 1)  
}  
}  
  
class Bubble  
{  
    constructor(x, y, r)  
    {  
        this.x = x  
        this.y = y  
        this.r = r  
        this.brightness = 0  
    }  
  
    move()  
    {  
        this.x = this.x + random(-1, 1)  
        this.y = this.y + random(-1, 1)  
    }  
  
    show()  
    {  
        noFill()  
        circle(this.x, this.y, this.r)  
    }  
  
    changeColour(bright)  
    {  
        this.brightness = bright  
    }  
  
    rollover(px, py)
```

```
{  
    let d = dist(px, py, this.x, this.y)  
}  
}
```

Notes

Still nothing to see.



Sketch D4.8 inside the bubble

The radius of the bubble is `r(this.r)`, so if `px` and `py` are less than the radius, we know that the mouse is inside the bubble. We use a simple boolean, `true` or `false`.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        bubbles[i].show()
        bubbles[i].move()
    }
    if (bubbles.length > 50)
    {
        bubbles.splice(0, 1)
    }
}
```

```
class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        noFill()
        circle(this.x, this.y, this.r)
    }

    changeColour(bright)
    {
        this.brightness = bright
    }

    rollover(px, py)
    {
        let d = dist(px, py, this.x, this.y)
        if (d < this.r)
        {
    }
```

```
        return true
    }
else
{
    return false
}
}
```

Notes

Still the same old...



Sketch D4.9 true colour

We want to fill the bubble with whatever colour is true (**255**) or false (**0**), the default, so we use the **fill()** function with a little bit of alpha.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        bubbles[i].show()
        bubbles[i].move()
    }
    if (bubbles.length > 50)
    {
        bubbles.splice(0, 1)
    }
}
```

```
class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        fill(this.brightness, 150)
        circle(this.x, this.y, this.r)
    }

    changeColour(bright)
    {
        this.brightness = bright
    }

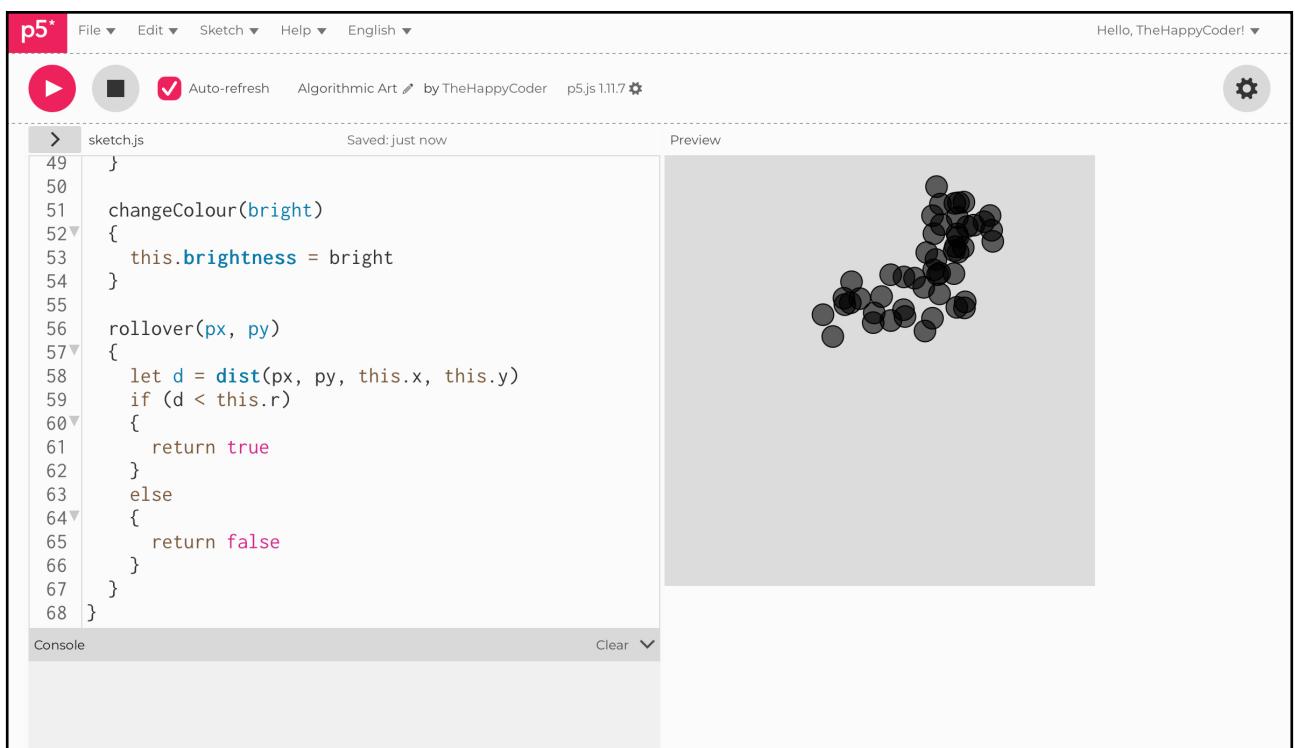
    rollover(px, py)
    {
        let d = dist(px, py, this.x, this.y)
        if (d < this.r)
        {
            return true
        }
    }
}
```

```
    }
} else
{
    return false
}
}
```

Notes

You get something, what you get is the default dark grey (black with alpha) bubbles. We are not finished yet.

Figure D4.9



The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and help. The title bar says "Hello, TheHappyCoder! ▾". The main area has tabs for "sketch.js" and "Preview". The code editor on the left contains the following JavaScript code:

```
sketch.js
Saved: just now
Preview

49 }
50
51 changeColour(bright)
52 {
53   this.brightness = bright
54 }
55
56 rollover(px, py)
57 {
58   let d = dist(px, py, this.x, this.y)
59   if (d < this.r)
60   {
61     return true
62   }
63   else
64   {
65     return false
66   }
67 }
68 }
```

The preview window on the right shows a cluster of dark gray circles of varying sizes, representing a data visualization or algorithmic art piece.



Sketch D4.10 array of bubbles check

We need to check through the array of bubbles to see if any of the bubbles contain the `mouseX` or `mouseY`. We use an `if()` statement for this job.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        if (bubbles[i].rollover(mouseX, mouseY))
        {
            bubbles[i].changeColour(255)
        }
        else
        {
            bubbles[i].changeColour(0)
        }
    }
}
```

```
bubbles[i].show()
bubbles[i].move()
}

if (bubbles.length > 50)
{
    bubbles.splice(0, 1)
}
}

class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }

    show()
    {
        fill(this.brightness, 150)
        circle(this.x, this.y, this.r)
    }

    changeColour(bright)
    {

```

```
    this.brightness = bright
}

rollover(px, py)
{
    let d = dist(px, py, this.x, this.y)
    if (d < this.r)
    {
        return true
    }
    else
    {
        return false
    }
}
}
```

Notes

Now the bubble changes to white when you hover or roll over it.

Figure D4.10

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, refresh, and settings, along with the text "Hello, TheHappyCoder! ▾". The main area has tabs for "sketch.js" and "Preview". The "sketch.js" tab contains the following code:

```
57 }
58
59 changeColour(bright)
60 {
61   this.brightness = bright
62 }
63
64 rollover(px, py)
65 {
66   let d = dist(px, py, this.x, this.y)
67   if (d < this.r)
68   {
69     return true
70   }
71   else
72   {
73     return false
74   }
75 }
76 }
```

The "Preview" tab shows a gray canvas with several dark gray circles scattered across it. One circle near the center is white with a black outline, indicating it is being controlled by the mouse.



Sketch D4.11 click of a mouse

! remove `mouseDragged()` and all its components.

We want to delete a bubble with the click of a mouse. We use the `splice()` function for this. It removes one or more elements from an array and returns those spliced values. First, let us create **20** randomly distributed bubbles and change the colour when we roll over them.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 20; i++)
    {
        let x = random(width)
        let y = random(height)
        bubble = new Bubble(x, y, 20)
        bubbles.push(bubble)
    }
}

// function mouseDragged()
// {
//     bubble = new Bubble(mouseX, mouseY, 20)
//     bubbles.push(bubble)
// }

function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
```

```

    if (bubbles[i].rollover(mouseX, mouseY))
    {
        bubbles[i].changeColour(255)
    }
    else
    {
        bubbles[i].changeColour(0)
    }
    bubbles[i].show()
    bubbles[i].move()
}

if (bubbles.length > 50)
{
    bubbles.splice(0, 1)
}
}

class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }

    move()
    {
        this.x = this.x + random(-1, 1)
        this.y = this.y + random(-1, 1)
    }
}

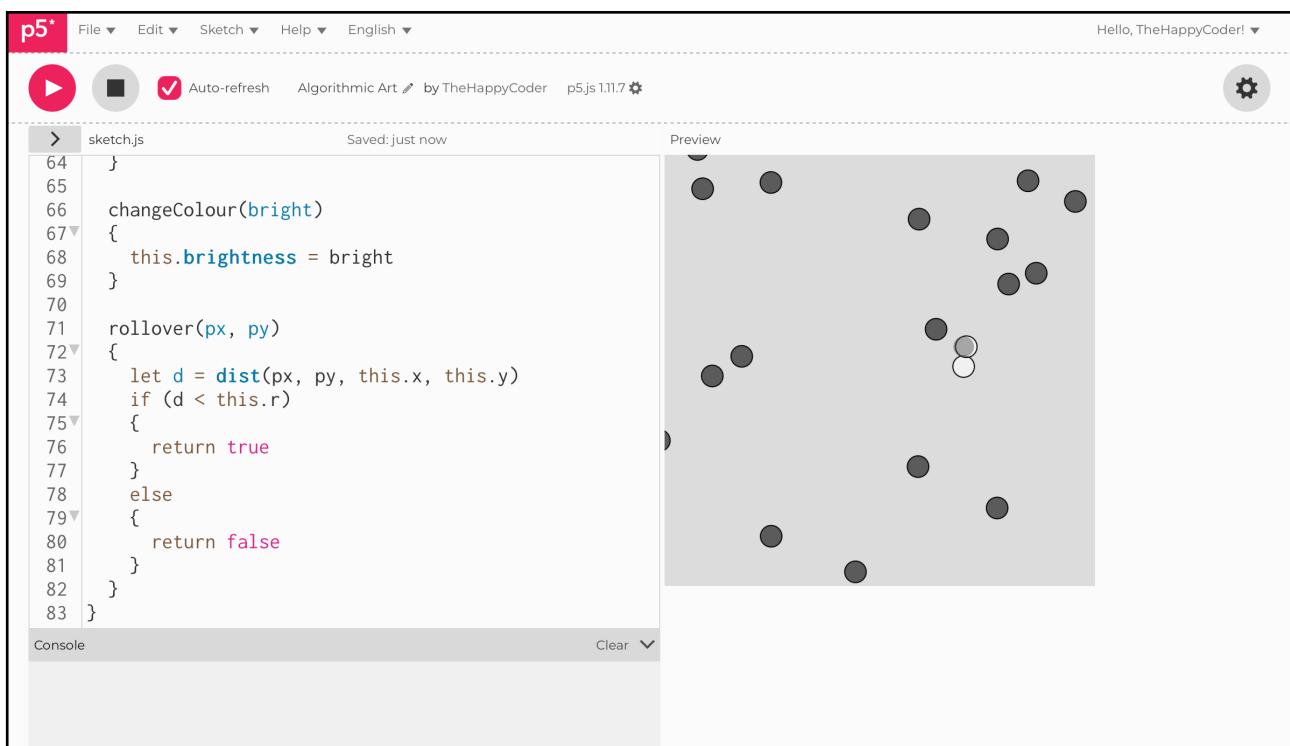
```

```
show()
{
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
}

changeColour(bright)
{
    this.brightness = bright
}

rollover(px, py)
{
    let d = dist(px, py, this.x, this.y)
    if (d < this.r)
    {
        return true
    }
    else
    {
        return false
    }
}
```

Figure D4.11



The screenshot shows the p5.js code editor interface. At the top, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The title bar says "Hello, TheHappyCoder! ▾". The main area has tabs for "sketch.js" and "Preview". The code editor displays the following JavaScript code:

```
sketch.js
Saved: just now
64 }
65
66 changeColour(bright)
67 {
68   this.brightness = bright
69 }
70
71 rollover(px, py)
72 {
73   let d = dist(px, py, this.x, this.y)
74   if (d < this.r)
75   {
76     return true
77   }
78   else
79   {
80     return false
81   }
82 }
83 }
```

The preview window shows a light gray square containing several dark gray circles of varying sizes, scattered across the surface.



Sketch D4.12 mousePressed()

Now we introduce `mousePressed()` and when there is a rollover and the mouse is pressed, we delete that `bubble` with `splice()`. The `mousePressed()` function has a `for()` loop that checks the array from the end and works backwards to the beginning to see if any of the bubbles have been clicked on.

The reason for doing it backwards is because of the index numbering. If you delete through starting at the beginning of the array, you will then change the indexing of the whole array, even though you are still working through the array one element at a time. It is possible you will miss an element otherwise.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 20; i++)
    {
        let x = random(width)
        let y = random(height)
        bubble = new Bubble(x, y, 20)
        bubbles.push(bubble)
    }
}

function mousePressed()
{
    for (let i = bubbles.length - 1; i >= 0; i--)
    {
    }
}
```

```
}
```

```
function draw()
{
    background(220)
    for (let i = 0; i < bubbles.length; i++)
    {
        if (bubbles[i].rollover(mouseX, mouseY))
        {
            bubbles[i].changeColour(255)
        }
        else
        {
            bubbles[i].changeColour(0)
        }
        bubbles[i].show()
        bubbles[i].move()
    }
    if (bubbles.length > 50)
    {
        bubbles.splice(0, 1)
    }
}
```

```
class Bubble
{
    constructor(x, y, r)
    {
        this.x = x
        this.y = y
        this.r = r
        this.brightness = 0
    }
}
```

```
move()
{
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
}

show()
{
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
}

changeColour(bright)
{
    this.brightness = bright
}

rollover(px, py)
{
    let d = dist(px, py, this.x, this.y)
    if (d < this.r)
    {
        return true
    }
    else
    {
        return false
    }
}
```



Notes

Nothing new is happening just yet. We are just checking; we have an empty `for()` loop at the moment.

Figure D4.12

The screenshot shows the p5.js IDE interface. At the top, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The title bar says "Hello, TheHappyCoder! ▾". The left panel contains the code for "sketch.js", which includes methods for changing color and checking if the mouse is over an object. The right panel, labeled "Preview", shows a gray canvas with several dark gray circles scattered across it. One circle near the bottom right is highlighted with a thin white border, indicating it is being rolled over by the mouse.

```
sketch.js
66 }
67
68 changeColour(bright)
69 {
70   this.brightness = bright
71 }
72
73 rollover(px, py)
74 {
75   let d = dist(px, py, this.x, this.y)
76   if (d < this.r)
77   {
78     return true
79   }
80   else
81   {
82     return false
83   }
84 }
85 }
```



Sketch D4.13 mouse delete

Now we can delete a bubble when it is clicked.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 20; i++)
    {
        let x = random(width)
        let y = random(height)
        bubble = new Bubble(x, y, 20)
        bubbles.push(bubble)
    }
}

function mousePressed()
{
    for (let i = bubbles.length - 1; i >= 0; i--)
    {
        if (bubbles[i].rollover(mouseX, mouseY))
        {
            bubbles.splice(i, 1)
        }
    }
}

function draw()
{
    background(220)
```

```

for (let i = 0; i < bubbles.length; i++)
{
  if (bubbles[i].rollover(mouseX, mouseY))
  {
    bubbles[i].changeColour(255)
  }
  else
  {
    bubbles[i].changeColour(0)
  }
  bubbles[i].show()
  bubbles[i].move()
}
if (bubbles.length > 50)
{
  bubbles.splice(0, 1)
}
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }
}

```

```
}

show()
{
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
}

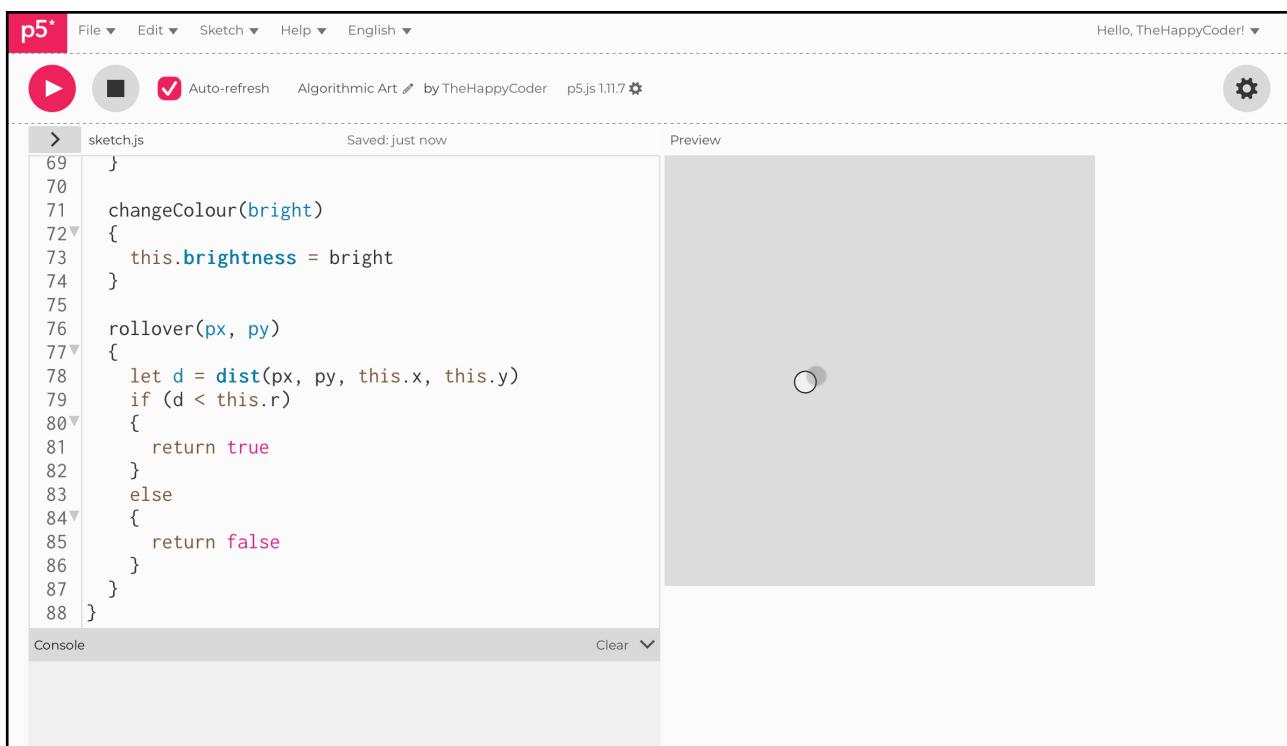
changeColour(bright)
{
    this.brightness = bright
}

rollover(px, py)
{
    let d = dist(px, py, this.x, this.y)
    if (d < this.r)
    {
        return true
    }
    else
    {
        return false
    }
}
```

Notes

You should see the bubble change colour still when you hover over it, but then when you click on it, it should disappear.

Figure D4.13



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right of the toolbar, it says "Hello, TheHappyCoder!" with a dropdown arrow. The main area has tabs for "sketch.js" and "Preview". The "sketch.js" tab contains the following code:

```
69 }
70
71 changeColour(bright)
72 {
73   this.brightness = bright
74 }
75
76 rollover(px, py)
77 {
78   let d = dist(px, py, this.x, this.y)
79   if (d < this.r)
80   {
81     return true
82   }
83   else
84   {
85     return false
86   }
87 }
88 }
```

The "Preview" tab shows a gray canvas with a small white circle in the center. Below the code editor is a "Console" section with a "Clear" button.